Principles of Software Construction: Objects, Design, and Concurrency

DevOps (part 1)

Jonathan Aldrich **Bogdan Vasilescu**

Carnegie Mellon University
School of Computer Science

S3D
Software and Societal
Systems Department

Lecture 23 Quiz

On Canvas

Administrative

- Frameworks to extend have been selected
    - We'll distribute the picks by tomorrow
    - If you are a maintainer, take some time to improve docs now, then wait and prepare to field Issues & PRs (quickly).
    - If not, pick one to extend when they come online
        - See the handout: add *n* new data plugins and *n - 1* new visualization plugins; make them reasonably different from the existing ones, and use at least one 3rd party API
    - Deadline: next week Friday

Where we are

Design for

understanding

change/ext.

reuse

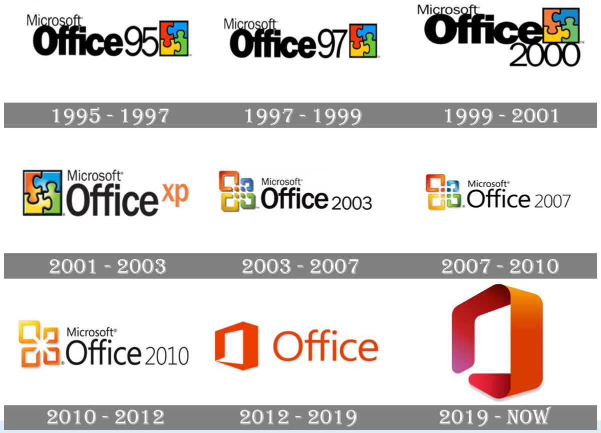
robustness

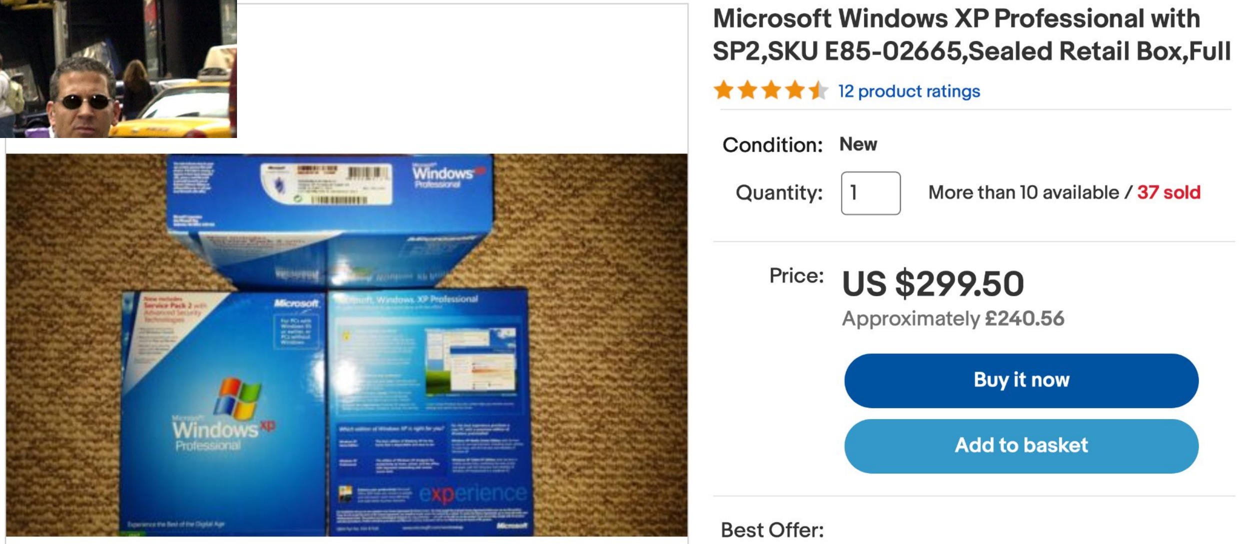
...

Small scale: One/few objects	*Mid scale:* Many objects	*Large scale:* Subsystems
Subtype Polymorphism ✓ Information Hiding, Contracts ✓ Immutability ✓ Types ✓ Static Analysis ✓ Unit Testing ✓	Domain Analysis ✓ Inheritance & Del. ✓ Responsibility Assignment, Design Patterns, Antipattern ✓ Promises/Reactive P. ✓ Static Analysis ✓	GUI vs Core ✓ Frameworks and Libraries ✓, APIs ✓ Distributed systems, microservices ✓ Testing for Robustness ✓ CI ✓, **DevOps**, Teams

DevOps

Early days: Boxed software, infrequent releases





Microsoft Windows XP Professional with SP2,SKU E85-02665,Sealed Retail Box,Full

★★★★½ 12 product ratings

Condition: New

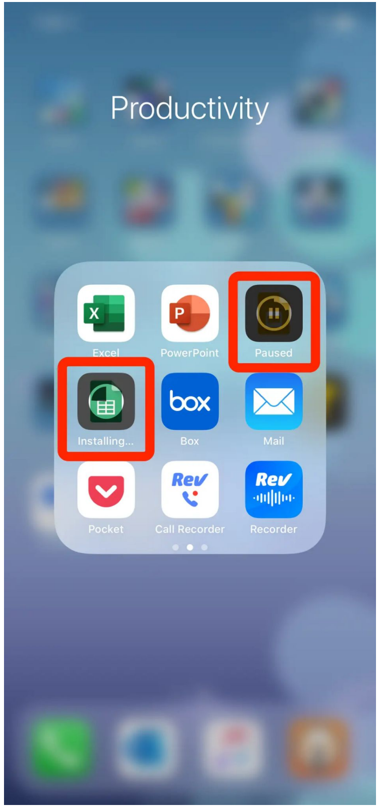
Quantity: 1 More than 10 available / 37 sold

Price: US $299.50
Approximately £240.56

Buy it now

Add to basket

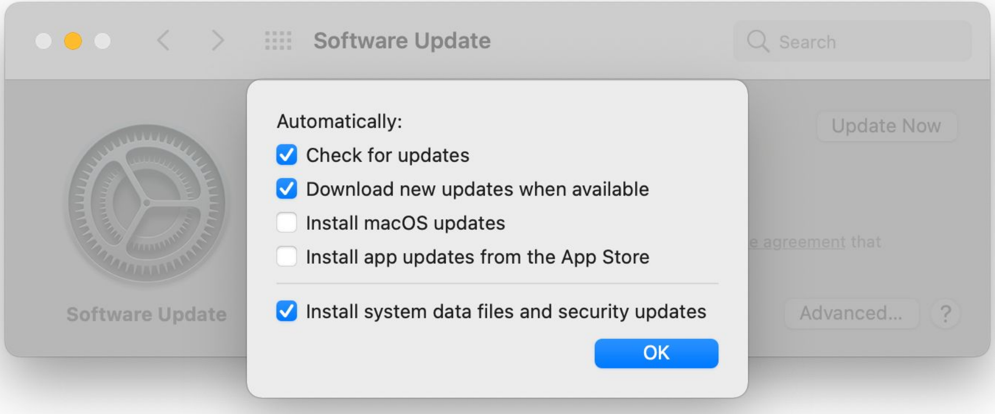
Best Offer:



These days:

Hosted software, frequent releases

Customer may not even notice update



Automatically:

☑ Check for updates

☑ Download new updates when available

□ Install macOS updates

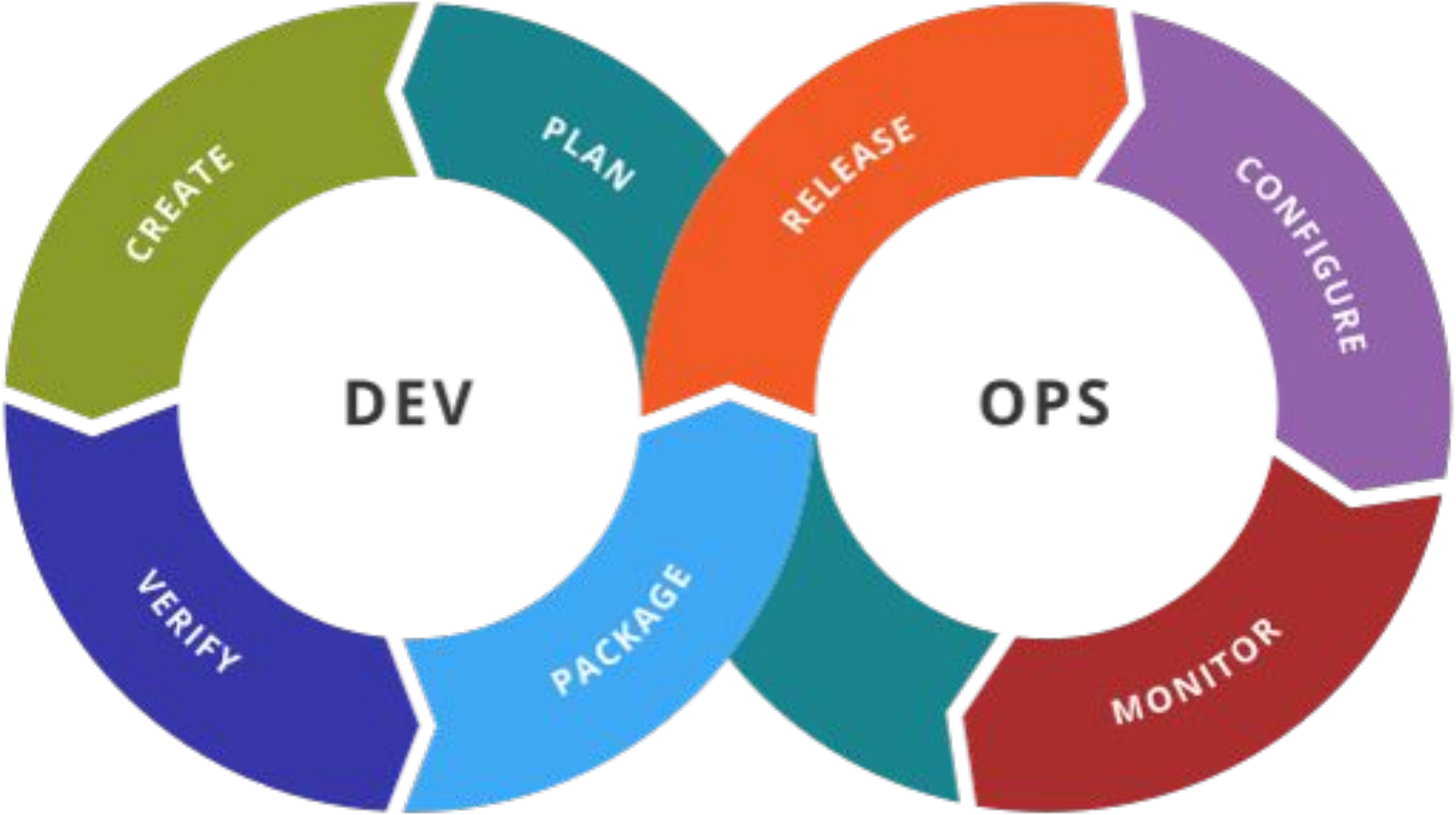
□ Install app updates from the App Store

☑ Install system data files and security updates

OK

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


DEV
OPS
PLAN
CREATE
VERIFY
PACKAGE
RELEASE
CONFIGURE
MONITOR

Dev resp. vs Ops resp.

- 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
- …

Dev resp. vs Ops resp.

- Coding
- Testing, st
- Continuou
- Bug tracki
- Running lo scalability
- …

- dware resources
- updates
- rformance
- ashes
- d spikes, …
- ase performance
- buted at scale
- eleases
- …

DevOps buzz word:
Shortening / Blending of Dev-Ops cycle

Key Ideas and Principles

Better coordinate between developers and operations (collaborative)

Reduce friction bringing changes from development into production

Consider the entire tool chain into production (holistic)

Document and version all dependencies and configurations ("configuration as code")

Small iterations, incremental and continuous releases

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

Common Practices

All configurations in version control

Test and deploy in containers

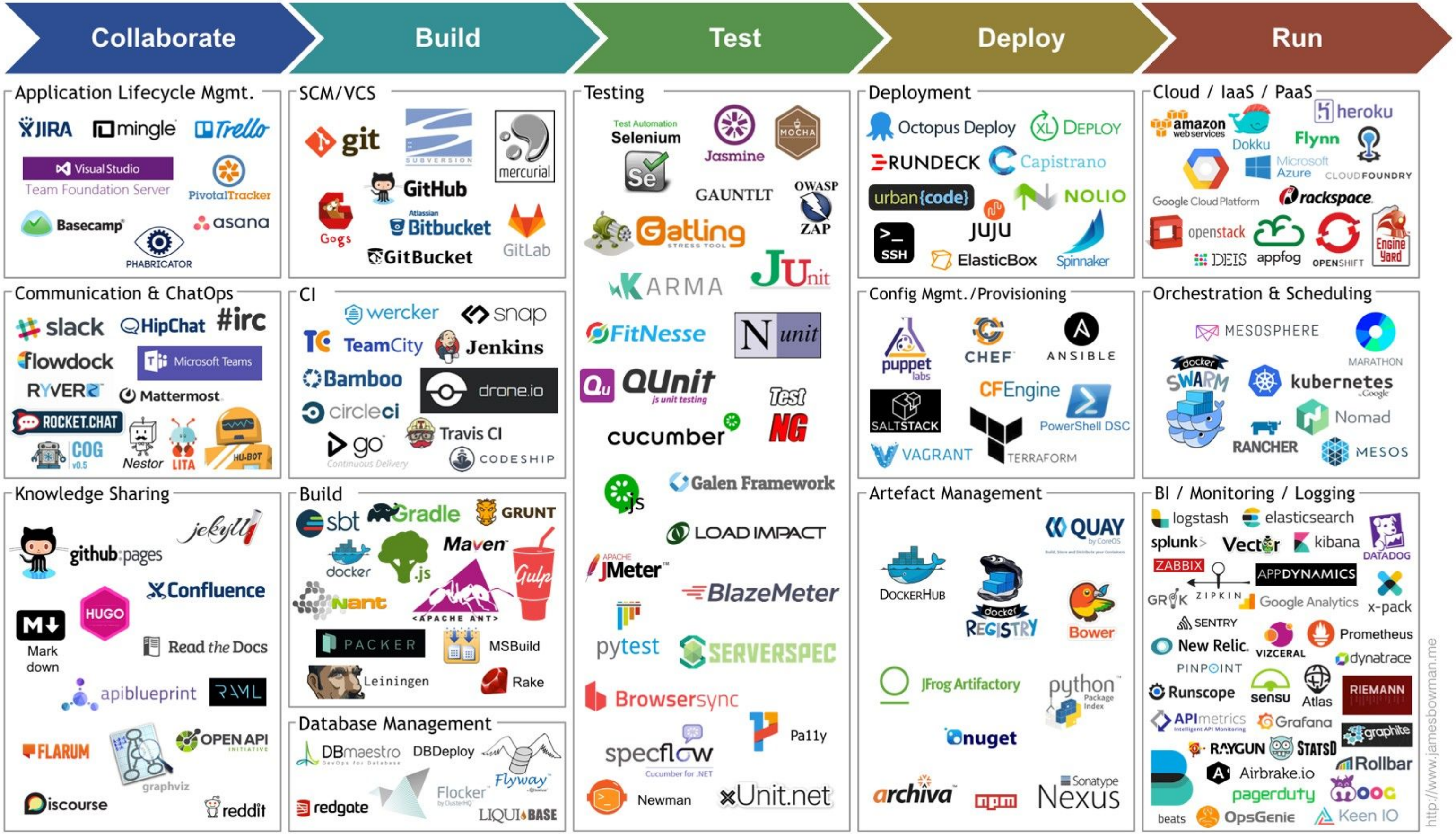
Automated testing, testing, testing, ...

Monitoring, orchestration, and automated actions in practice

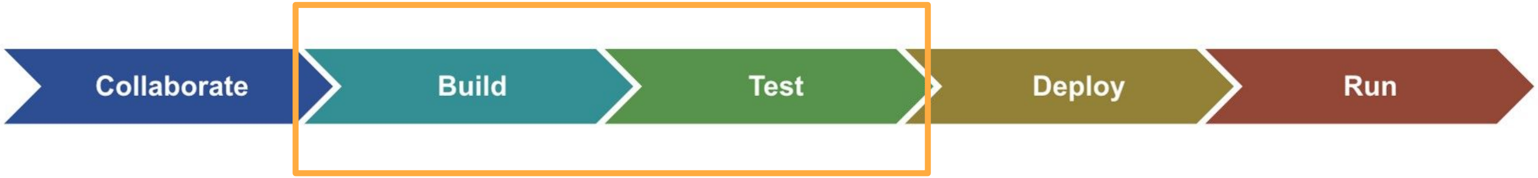
Microservice architectures

Release frequently

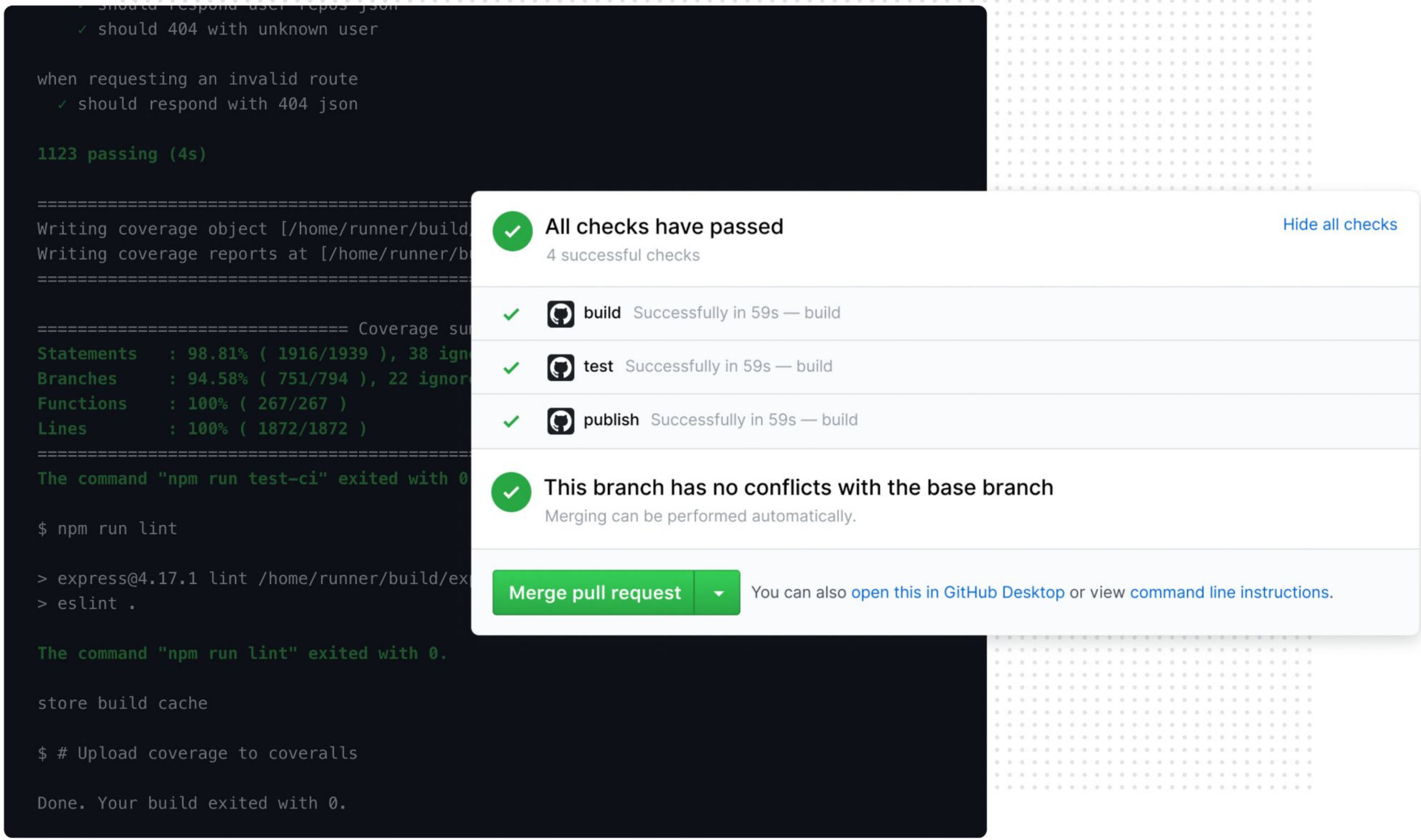
Heavy Automation, Lots of Tooling



Let’s zoom in on the different stages



Recall: Continuous Integration



```
    ✓ should respond user repos json
    ✓ should 404 with unknown user

when requesting an invalid route
  ✓ should respond with 404 json

1123 passing (4s)

=============================================
Writing coverage object [/home/runner/build/
Writing coverage reports at [/home/runner/bu
=============================================

=============================== Coverage su
Statements   : 98.81% ( 1916/1939 ), 38 igno
Branches     : 94.58% ( 751/794 ), 22 ignore
Functions    : 100% ( 267/267 )
Lines        : 100% ( 1872/1872 )
=============================================
The command "npm run test-ci" exited with 0

$ npm run lint

> express@4.17.1 lint /home/runner/build/ex
> eslint .

The command "npm run lint" exited with 0.

store build cache

$ # Upload coverage to coveralls

Done. Your build exited with 0.
```

✓ All checks have passed

4 successful checks

Hide all checks

✓ build Successfully in 59s — build

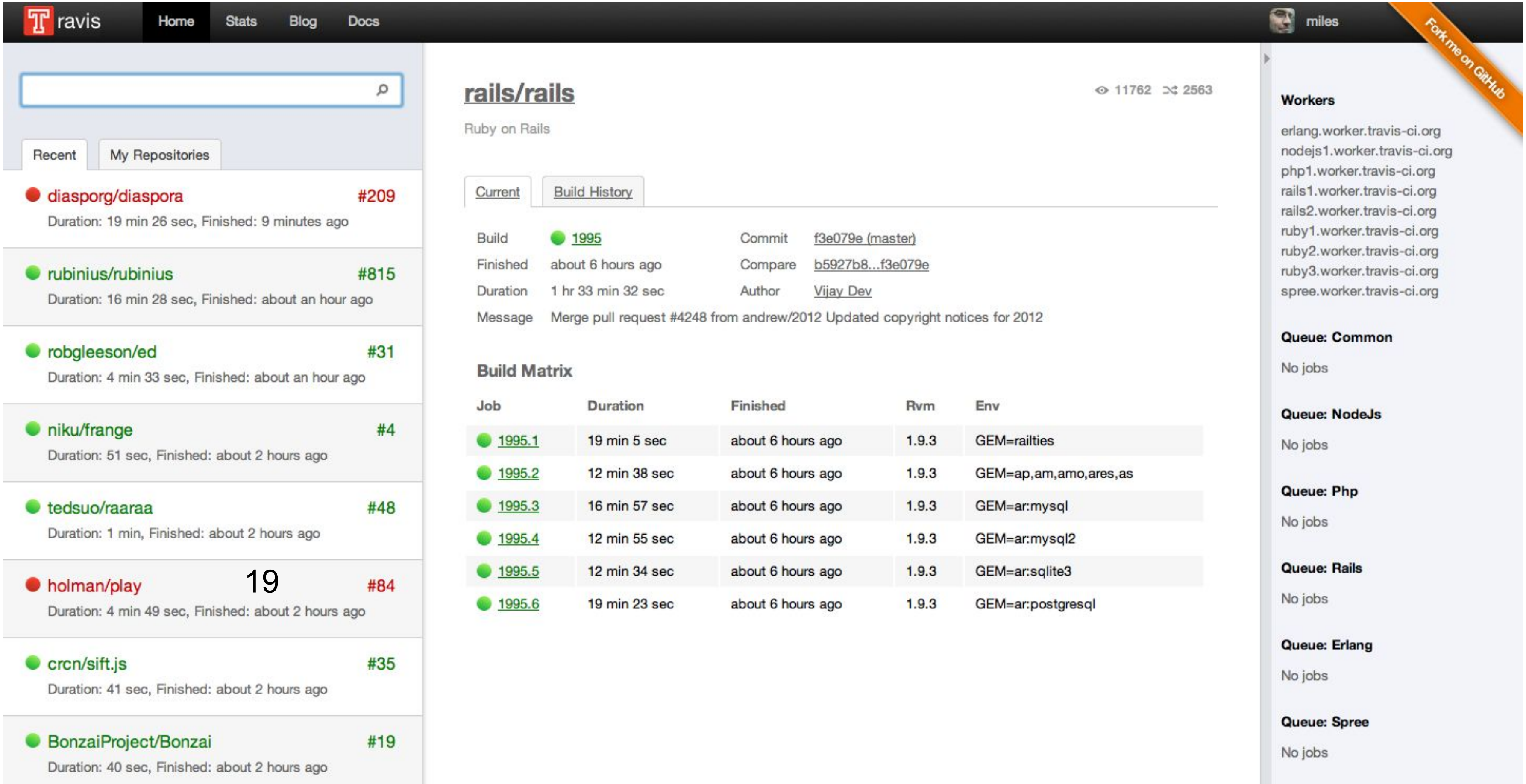
✓ test Successfully in 59s — build

✓ publish Successfully in 59s — build

✓ This branch has no conflicts with the base branch

Merging can be performed automatically.

Merge pull request ▾ You can also open this in GitHub Desktop or view command line instructions.



- ● 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
- ● robgleeson/ed #31
  Duration: 4 min 33 sec, Finished: about an hour ago
- ● niku/frange #4
  Duration: 51 sec, Finished: about 2 hours ago
- ● tedsuo/raaraa #48
  Duration: 1 min, Finished: about 2 hours ago
- ● holman/play #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

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

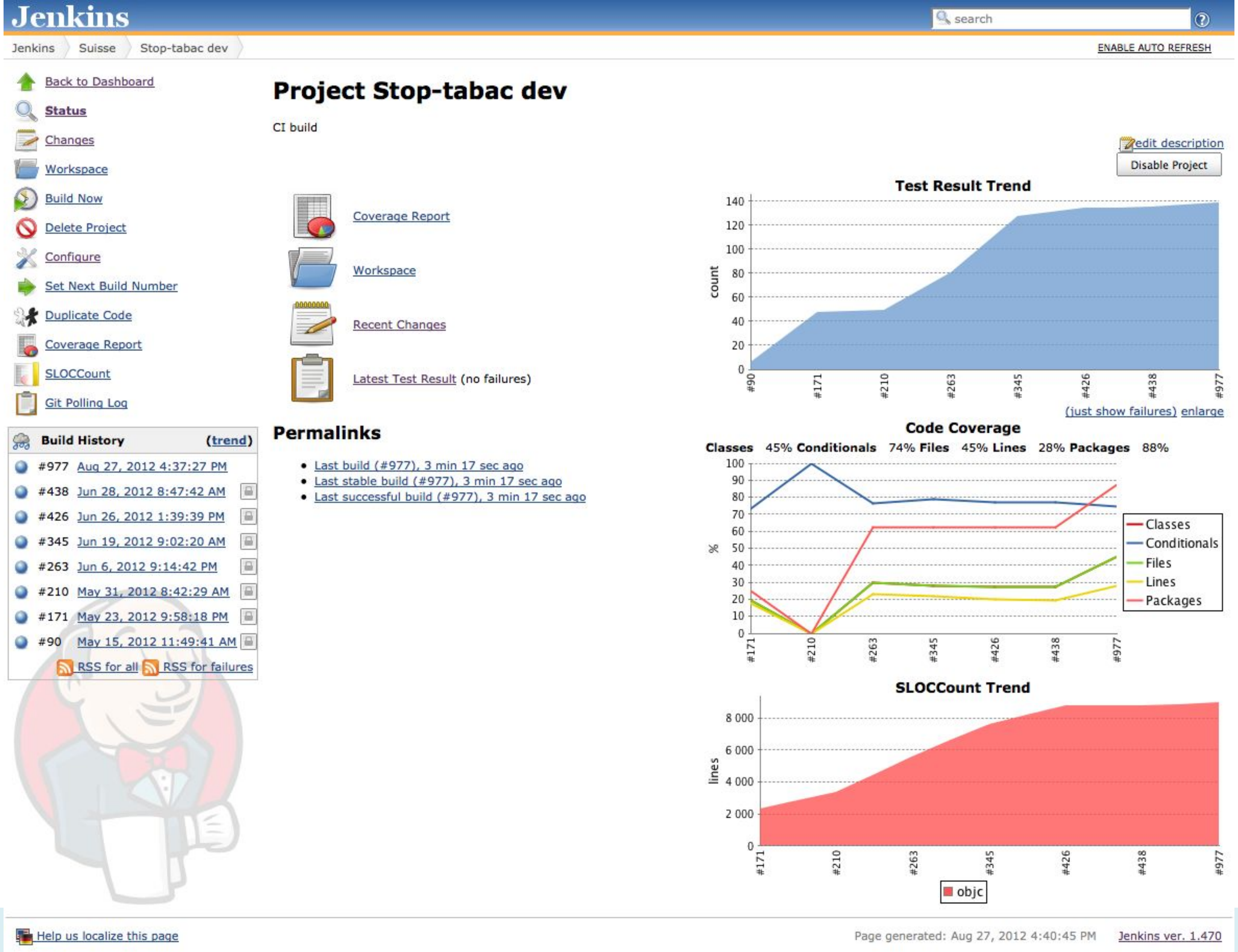
No jobs

Queue: Erlang

No jobs

Queue: Spree

No jobs



Jenkins

search

Jenkins › Suisse › Stop-tabac dev

ENABLE AUTO REFRESH

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

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

Project Stop-tabac dev

CI build

edit description

Disable Project

- Coverage Report
- Workspace
- Recent Changes
- Latest Test Result (no failures)

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

Code Coverage

Classes	Conditionals	Files	Lines	Packages
45%	74%	45%	28%	88%

(just show failures) enlarge

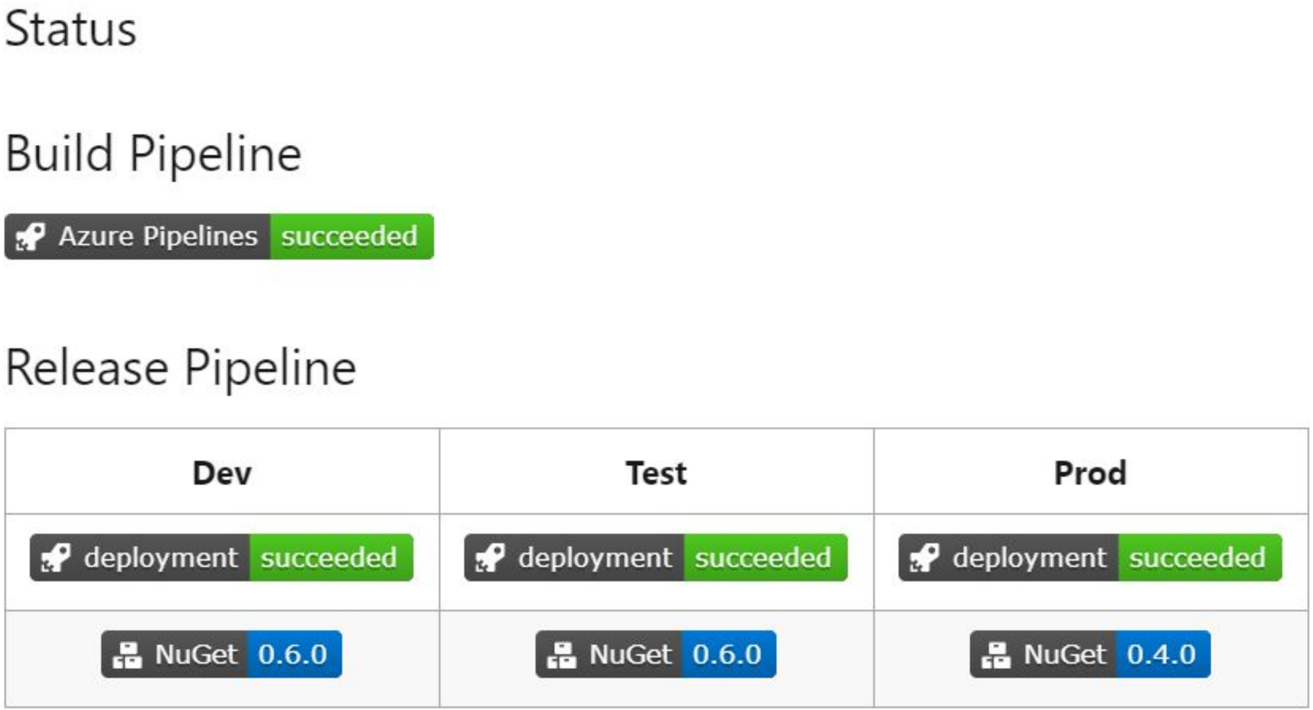
Help us localize this page

Page generated: Aug 27, 2012 4:40:45 PM Jenkins ver. 1.470

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

Aside: The role of signaling



Status

Build Pipeline

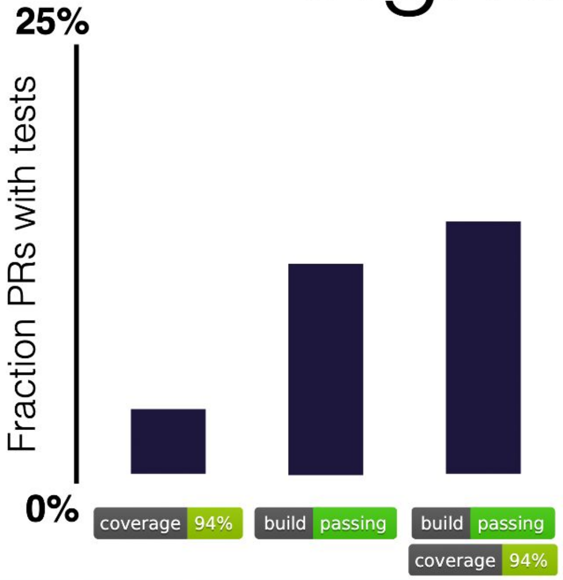
Azure Pipelines succeeded

Release Pipeline

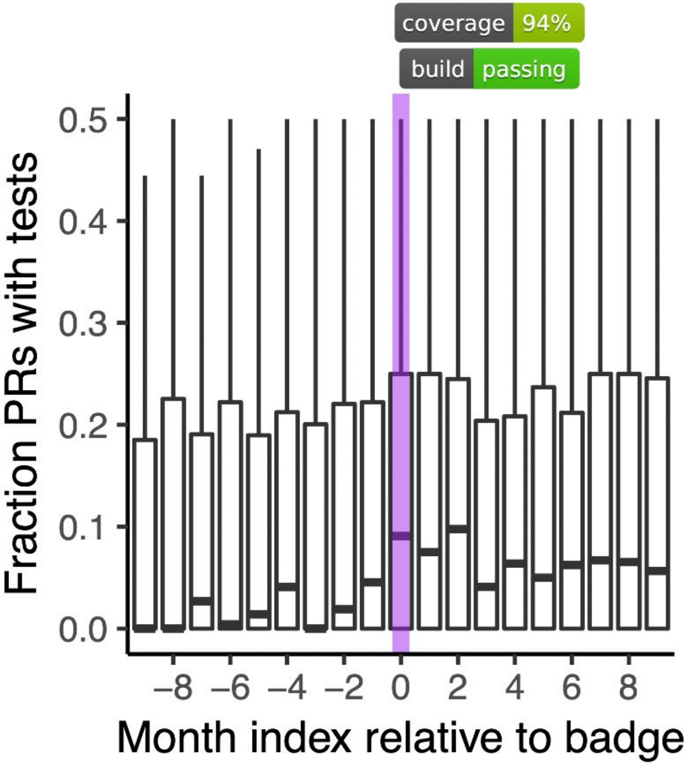
Dev	Test	Prod
deployment succeeded	deployment succeeded	deployment succeeded
NuGet 0.6.0	NuGet 0.6.0	NuGet 0.4.0

https://blog.devops4me.com/status-badges-in-azure-devops-pipelines/

Signals of PR quality



Result: Build status+code coverage badges indicate *more tests in PRs*



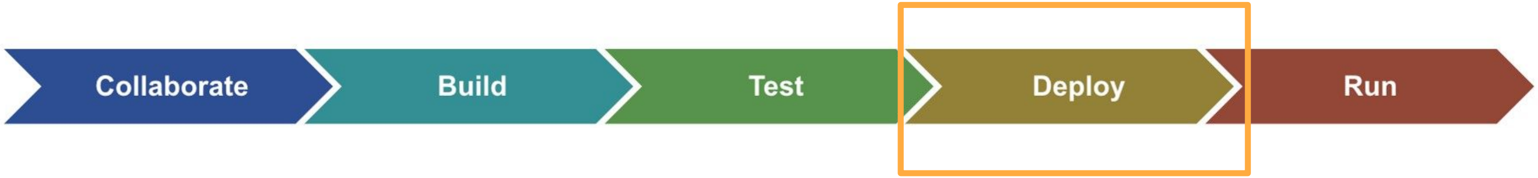
STRUDEL
Carnegie Mellon University

Trockman, A., Zhou, S., Kästner, C., & Vasilescu, B. (2018). Adding sparkle to social coding: An empirical study of repository badges in the npm ecosystem. *International Conference on Software Engineering* (pp.

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

Releasing Software

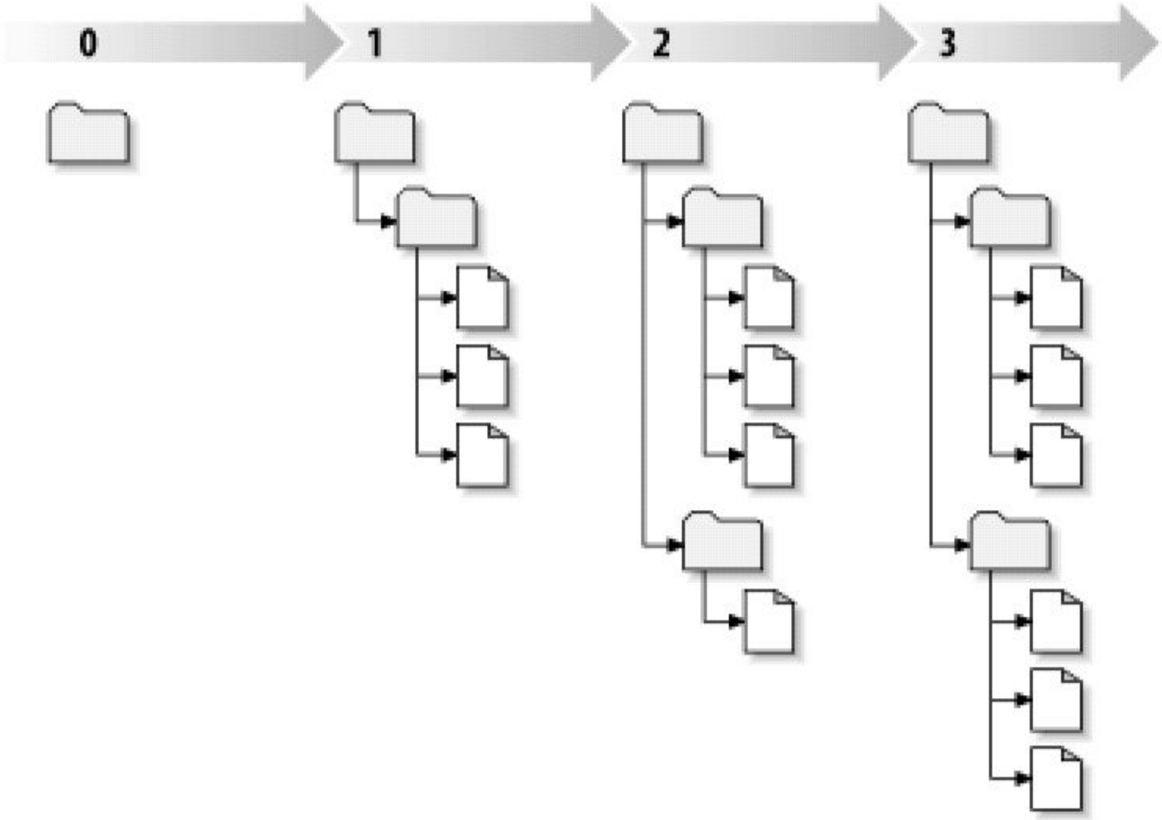


Semantic Versioning for Releases

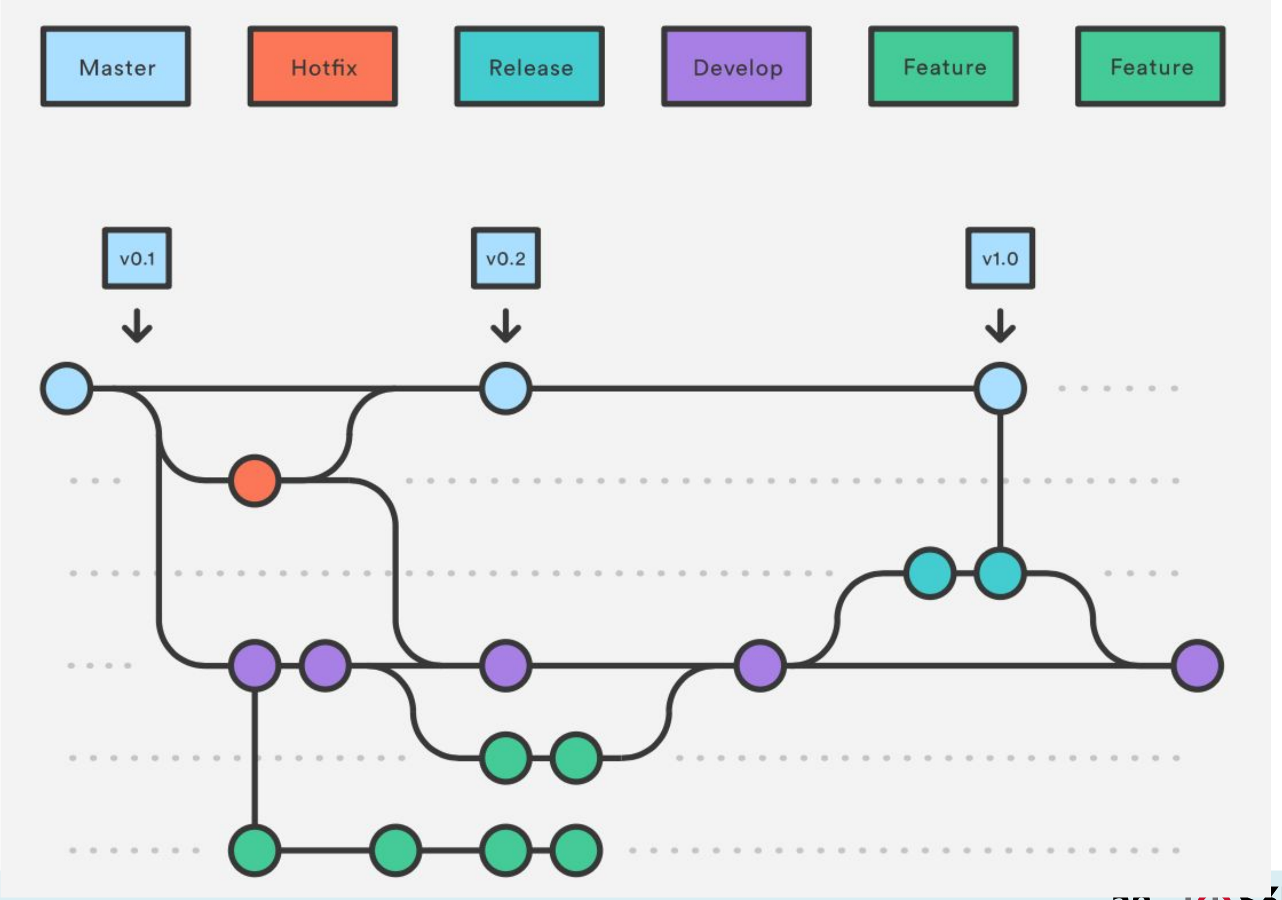
- Given a version number MAJOR.MINOR.PATCH, increment the:
  - MAJOR version when you make incompatible API changes,
  - MINOR version when you add functionality in a backwards-compatible manner, and
  - PATCH version when you make backwards-compatible bug fixes.
- Additional labels for pre-release and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

http://semver.org/

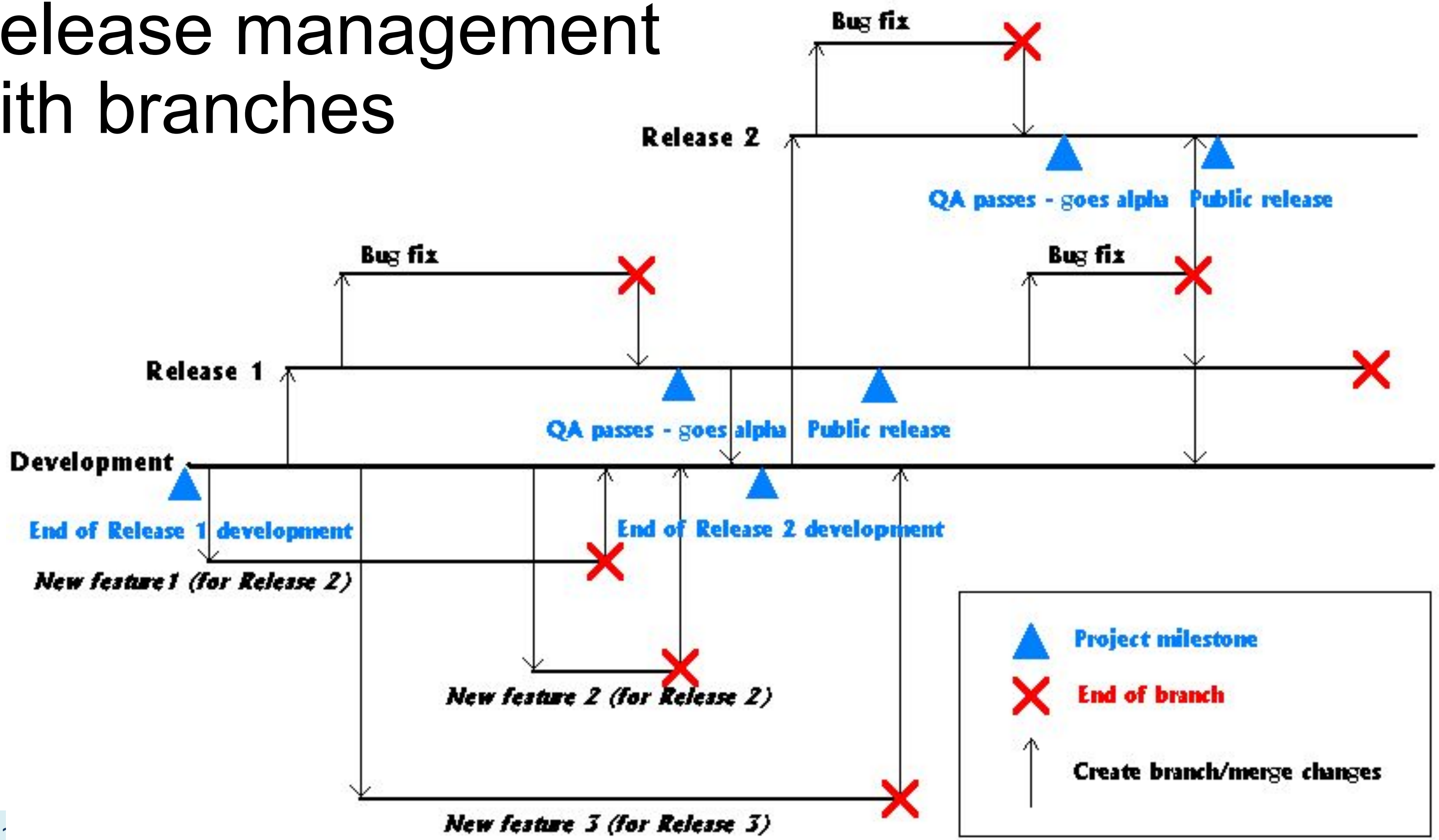
Versioning entire projects

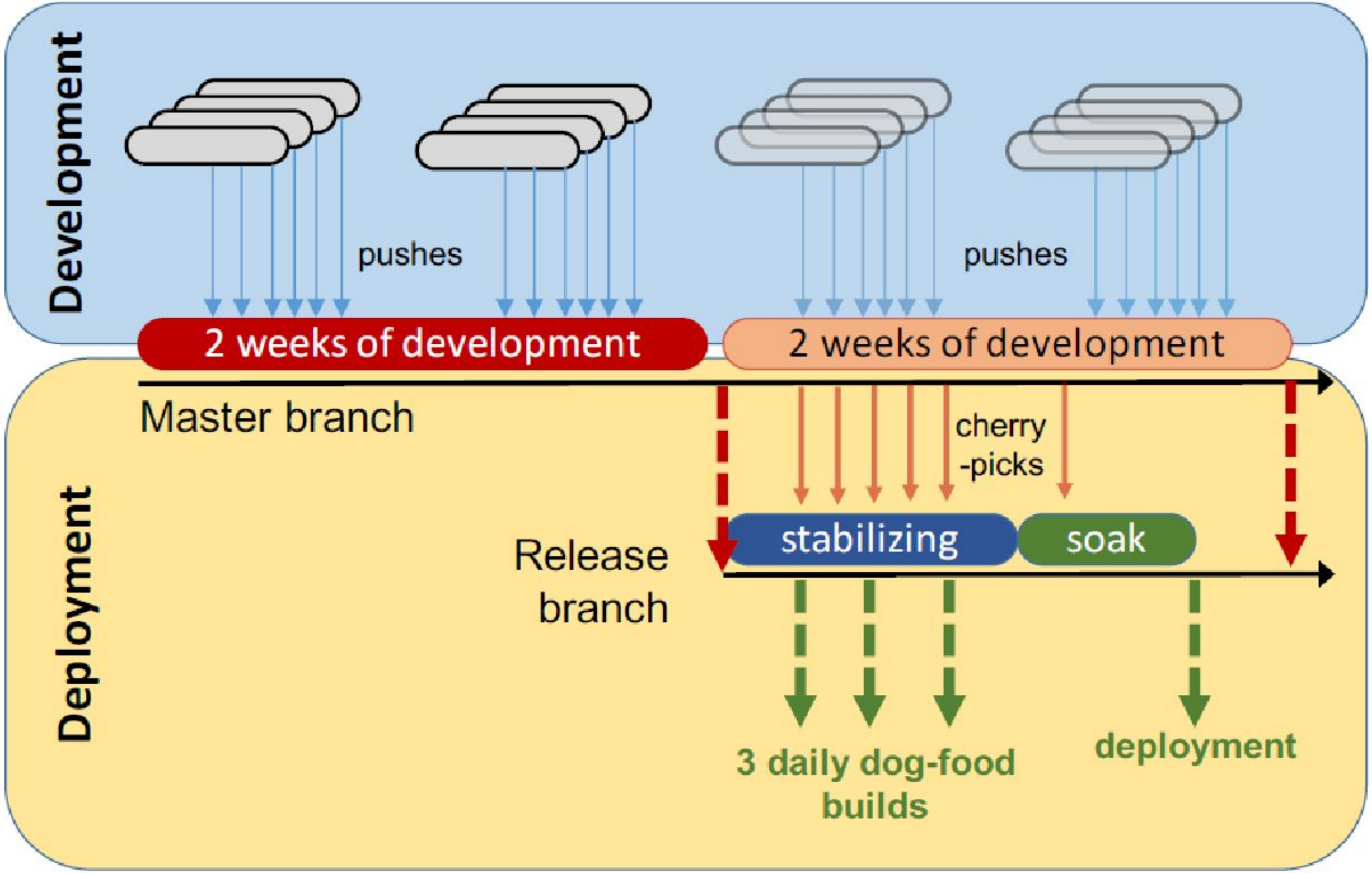


Release management with branches



Release management with branches


Bug fix
Release 2
QA passes - goes alpha
Public release
Bug fix
Bug fix
Release 1
QA passes - goes alpha
Public release
Development
End of Release 1 development
End of Release 2 development
New feature 1 (for Release 2)
New feature 2 (for Release 2)
New feature 3 (for Release 3)
Project milestone
End of branch
Create branch/merge changes


Development
pushes
pushes
2 weeks of development
2 weeks of development
Master branch
cherry
-picks
Deployment
Release
branch
stabilizing
soak
3 daily dog-food
builds
deployment

Example: Pre-2017 release management model at Facebook

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)

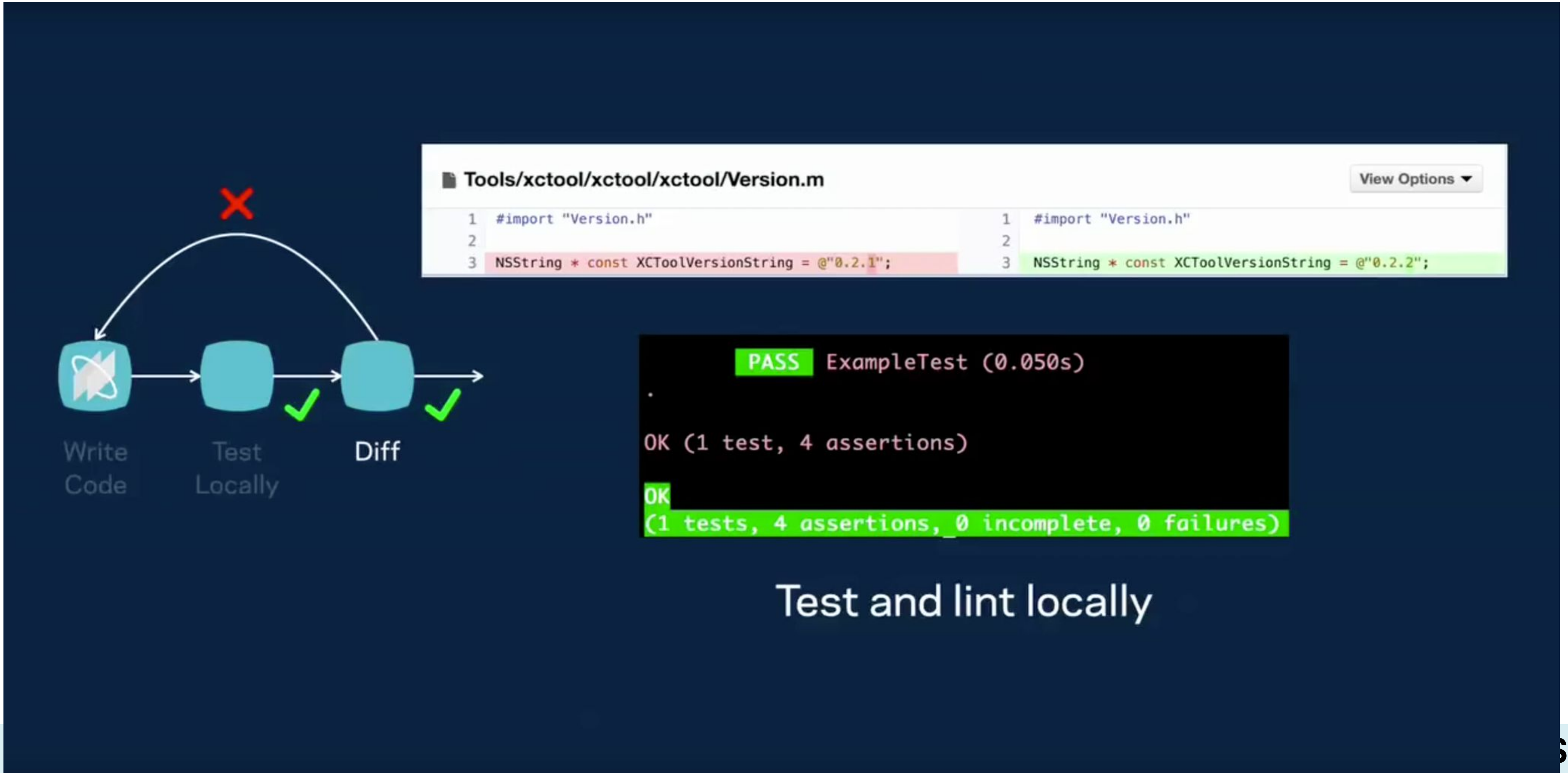
Integration tests (black box, in simulators)

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.

Diff lifecycle: local testing



Tools/xctool/xctool/xctool/Version.m

```
#import "Version.h"                                      #import "Version.h"

NSString * const XCToolVersionString = @"0.2.1";         NSString * const XCToolVersionString = @"0.2.2";
```

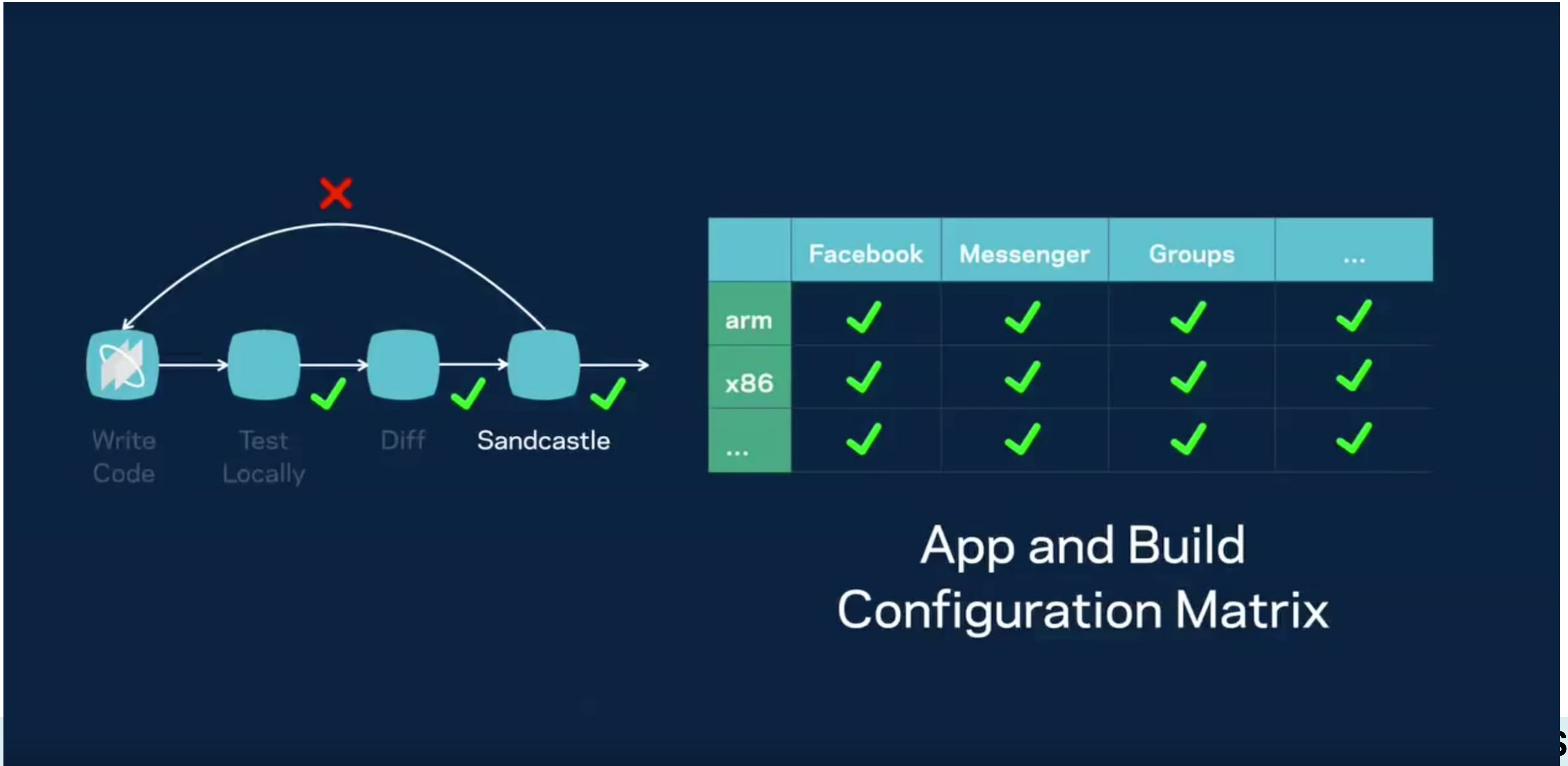
```
      PASS  ExampleTest (0.050s)
.

OK (1 test, 4 assertions)

OK
(1 tests, 4 assertions, 0 incomplete, 0 failures)
```

Test and lint locally

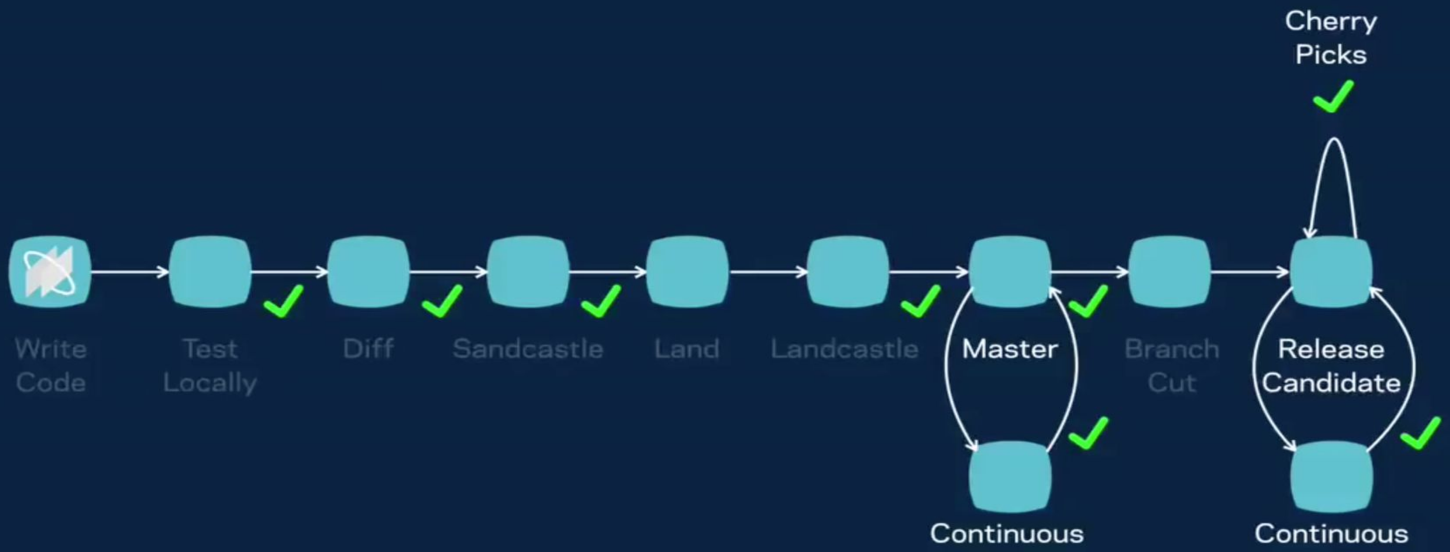
Diff lifecycle: CI testing (data center)



	Facebook	Messenger	Groups	...
arm	✓	✓	✓	✓
x86	✓	✓	✓	✓
...	✓	✓	✓	✓

App and Build Configuration Matrix

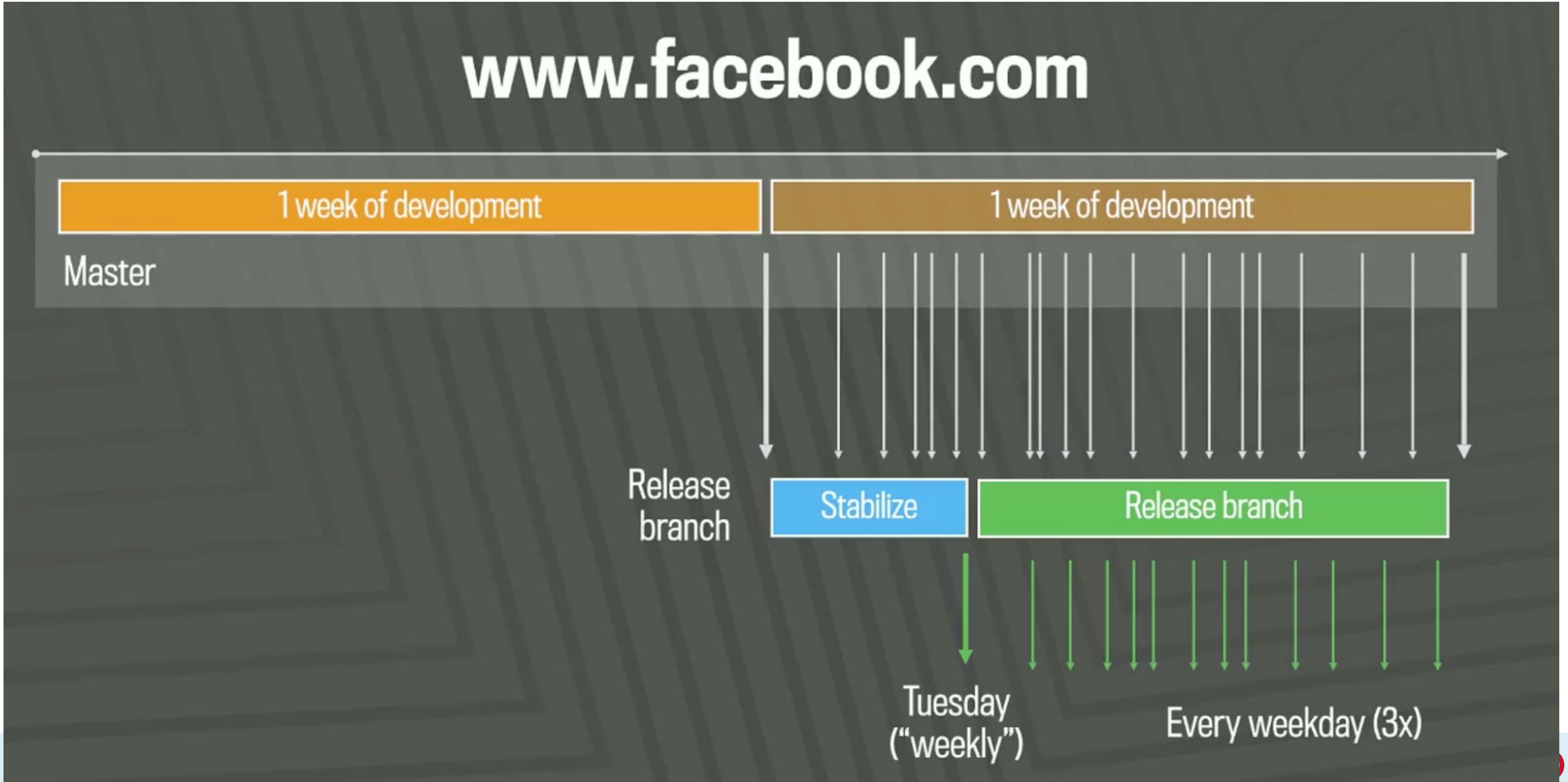
Diff lifecycle: diff ends up on main branch



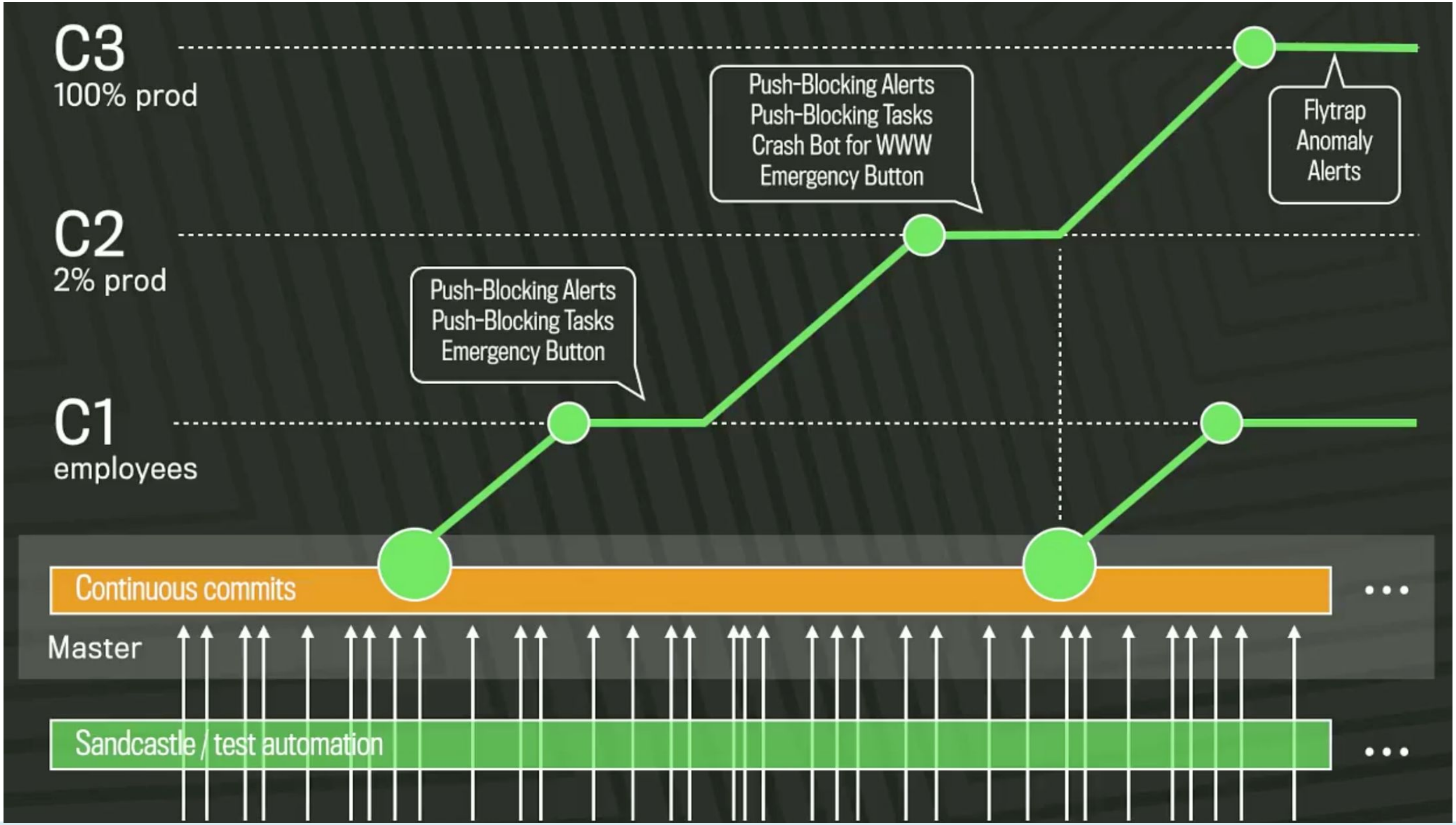
Dogfooding

(the use of one’s own products)

Release every two weeks



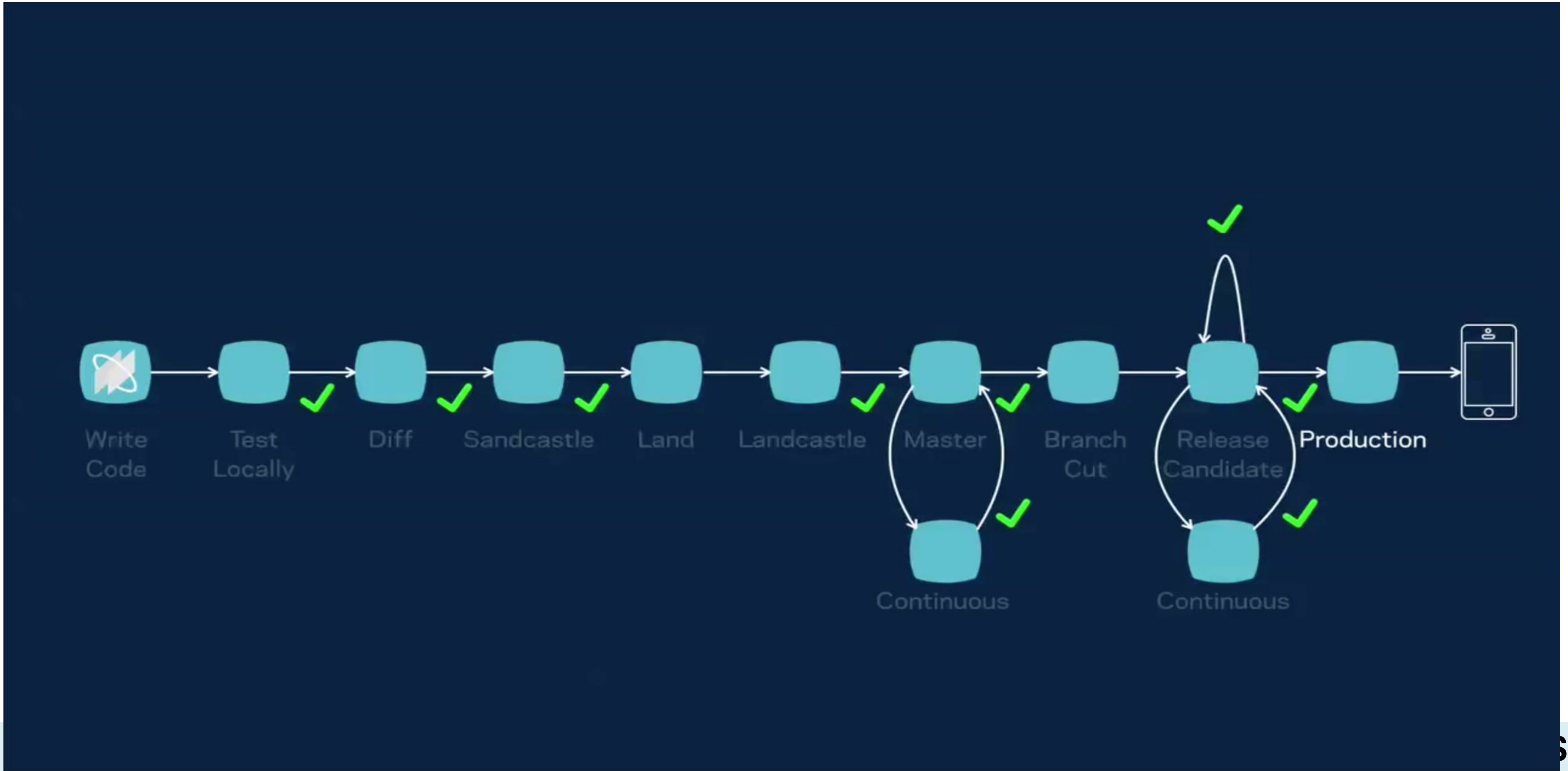
Quasi-continuous push from master (1,000+ devs, 1,000 diffs/day); 10 pushes/day



https://samritchie.wordpress.com/2013/10/16/build-server-traffic-lights/

https://www.softwire.com/blog/2013/09/26/continuous-integration-traffic-lights-revamp/index.html

Diff lifecycle: in production



Release Challenges for Mobile Apps

- Large downloads
- Download time at user discretion
- Different versions in production
- Pull support for old releases?

Any alternatives?

Release Challenges for Mobile Apps

- Large downloads
- Download time at user discretion
- Different versions in production
- Pull support for old releases?

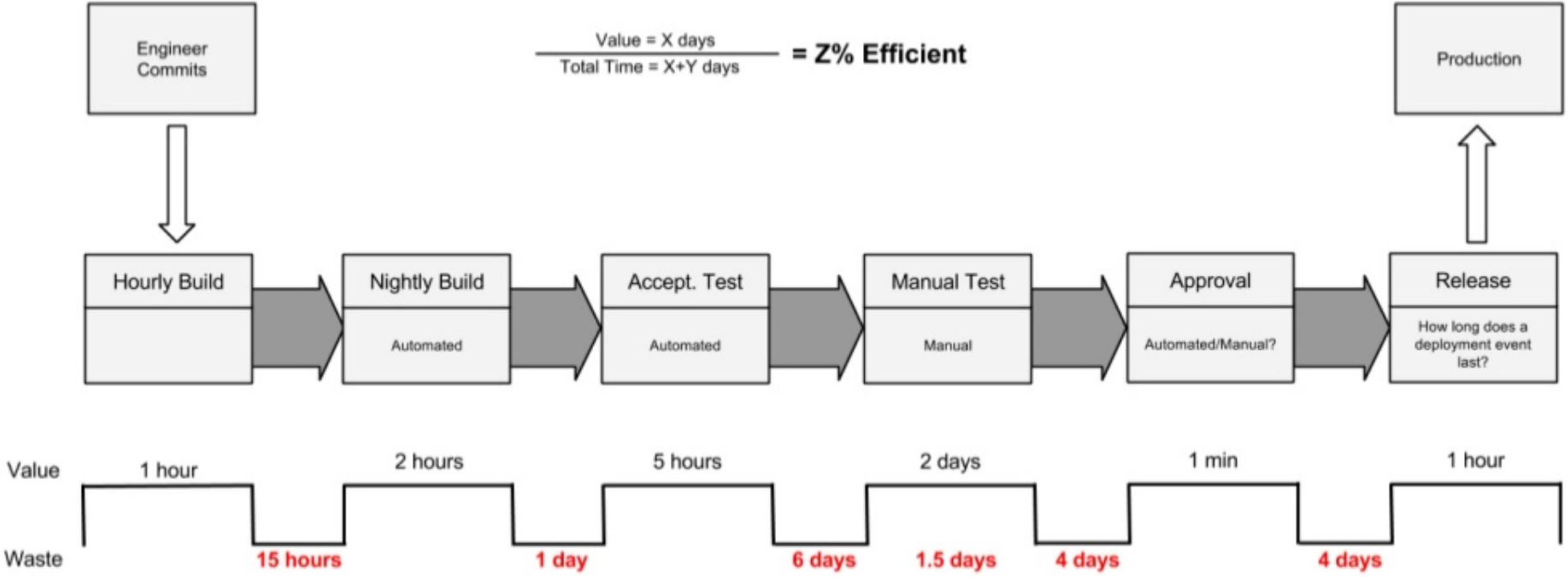
Current trend:

- App as container, most content + layout from server
- Server side releases silent and quick, consistent

From Release Date to Continuous Release

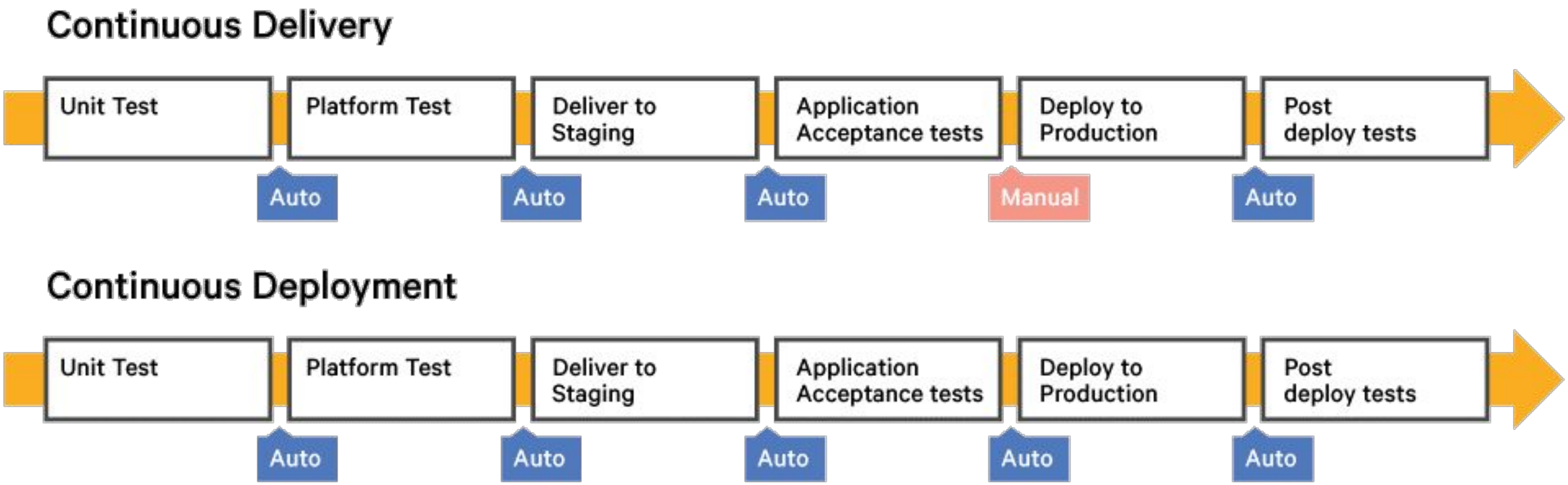
- Traditional View: Boxed Software
  - Working toward fixed release date, QA heavy before release
  - Release and move on
  - Fix post-release defects in next release or through expensive patches
- Frequent releases
  - Incremental updates delivered frequently (weeks, days, …), e.g. Browsers
  - Automated 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

Efficiency of release pipeline



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

Let’s automate all the things!

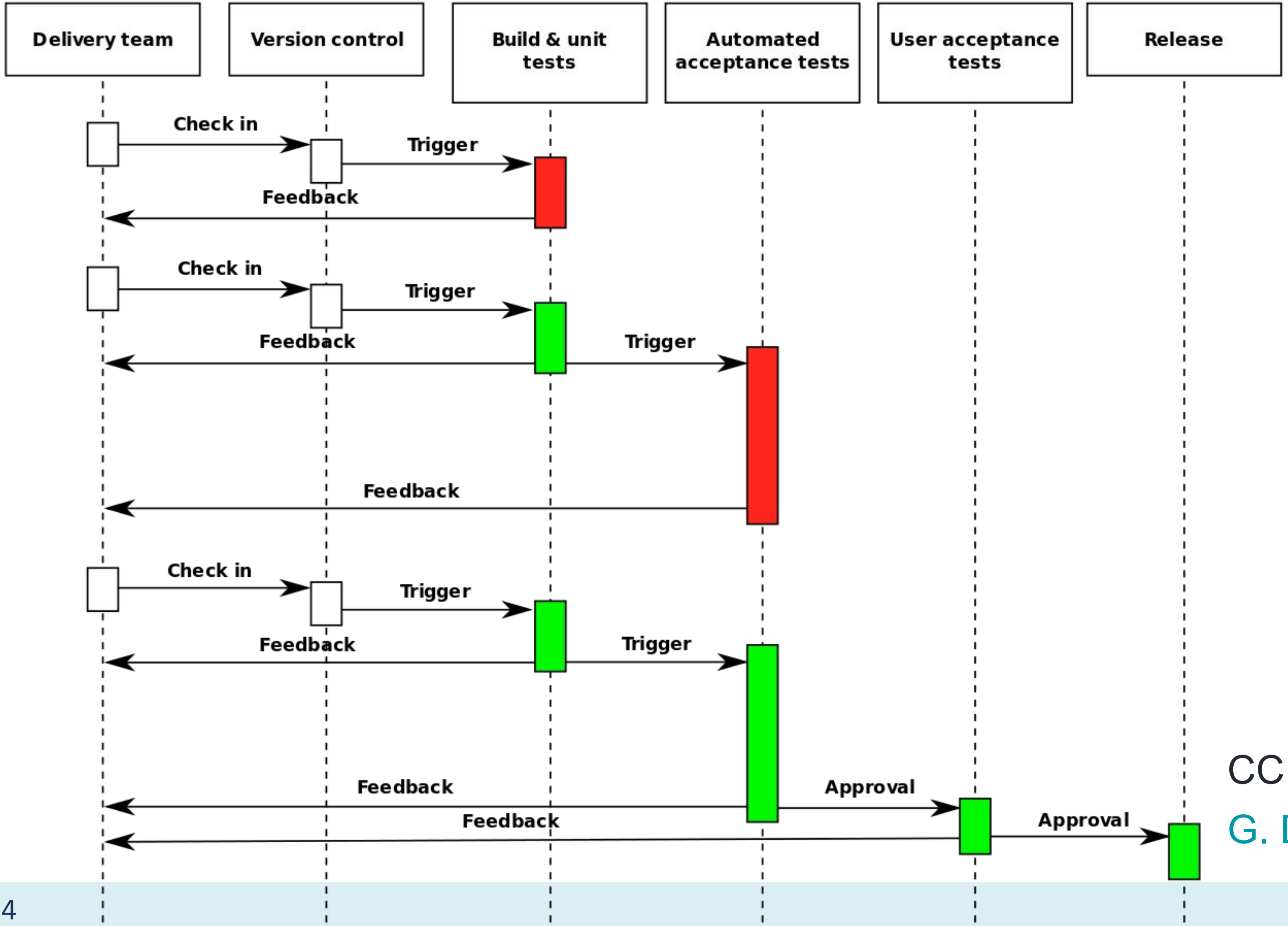


Continuous Delivery

Unit Test → Auto → Platform Test → Auto → Deliver to Staging → Auto → Application Acceptance tests → Manual → Deploy to Production → Auto → Post deploy tests

Continuous Deployment

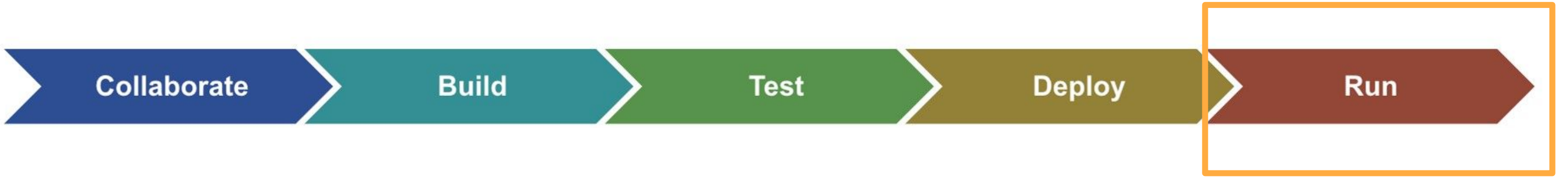
Unit Test → Auto → Platform Test → Auto → Deliver to Staging → Auto → Application Acceptance tests → Auto → Deploy to Production → Auto → Post deploy tests



CC BY-SA 4.0

G. Détrez

Running Software



Containers drastically simplify managing ops

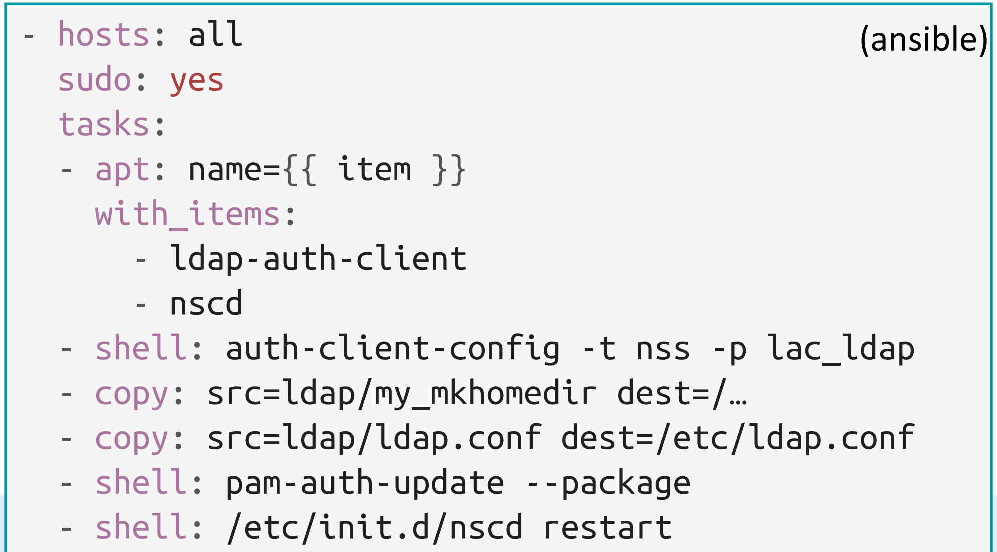
A virtual machine, but:

- Lightweight virtualization
- Sub-second boot time
- Shareable virtual images with full setup incl. configuration settings
- Separate docker images for separate services (web server, business logic, database, …)
- Used a lot in development, not just deployment

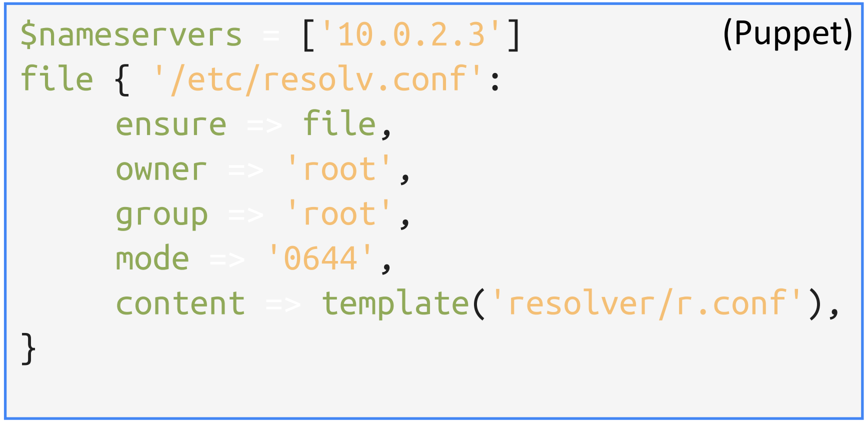
Lots more on Tuesday

Key idea: Configuration management, Infrastructure as Code

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


(ansible)

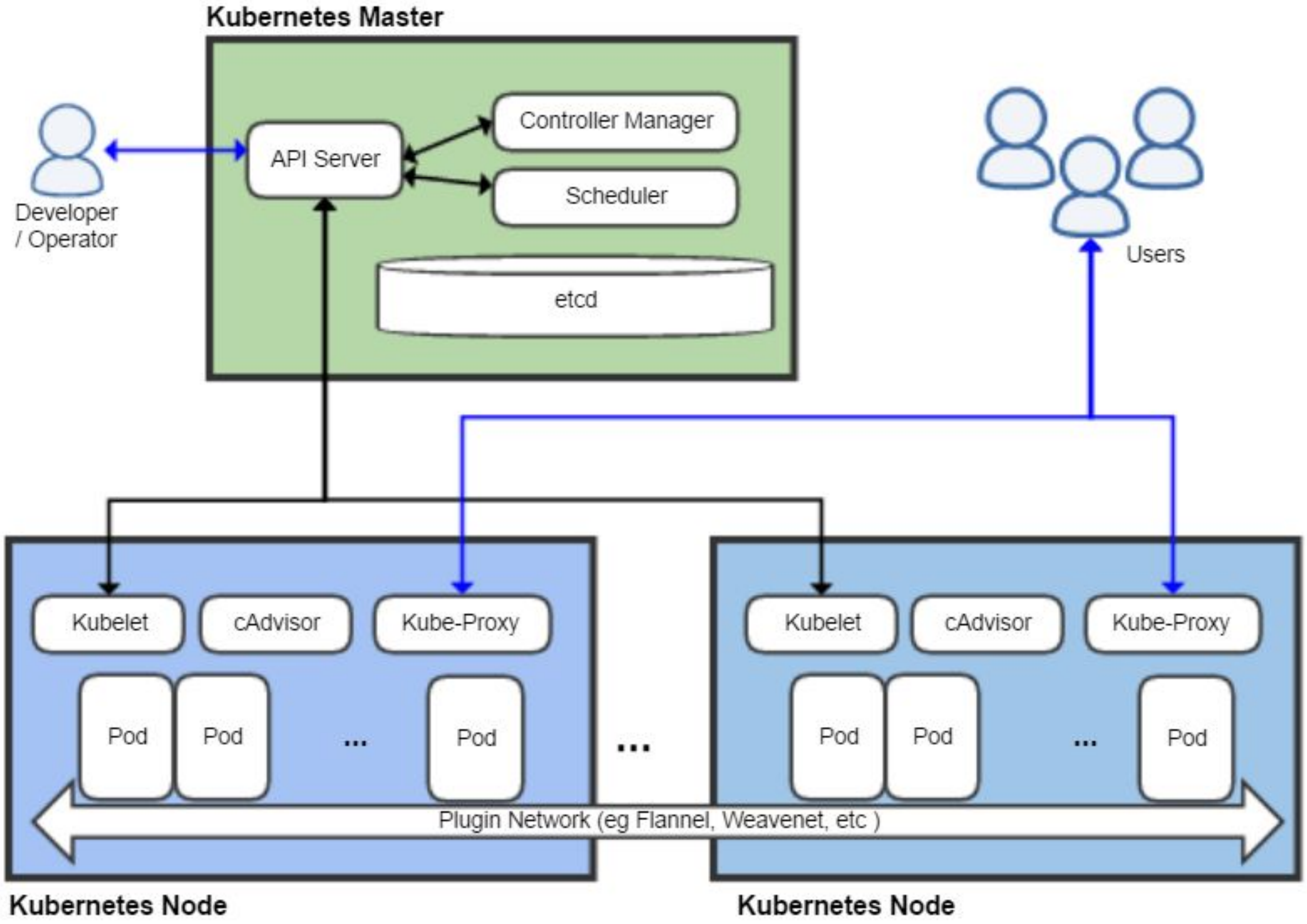
```
- hosts: all
  sudo: yes
  tasks:
  - apt: name={{ item }}
    with_items:
      - ldap-auth-client
      - nscd
  - shell: auth-client-config -t nss -p lac_ldap
  - copy: src=ldap/my_mkhomedir dest=/…
  - copy: src=ldap/ldap.conf dest=/etc/ldap.conf
  - shell: pam-auth-update --package
  - shell: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


(Puppet)

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

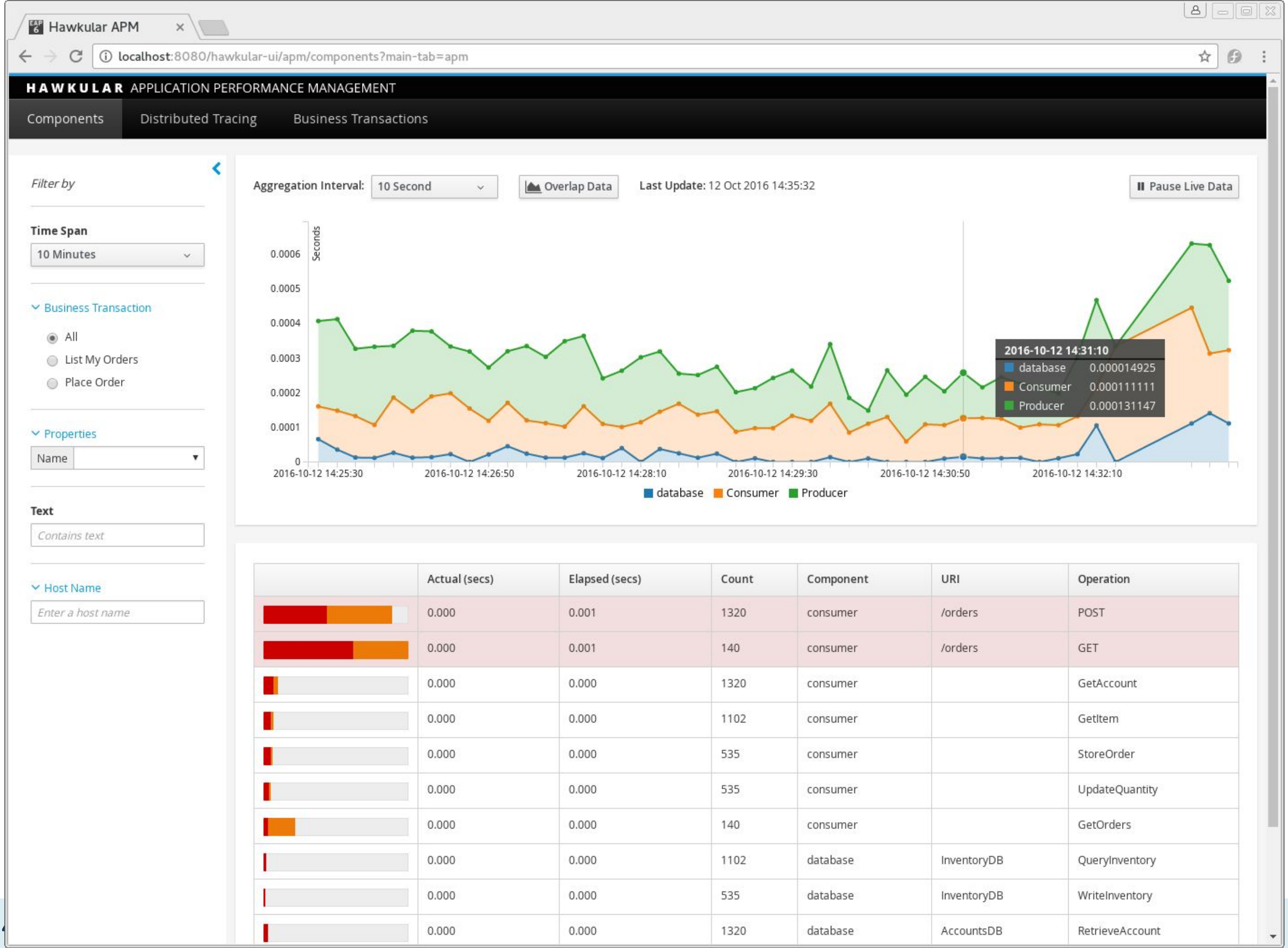
Container Orchestration with Kubernetes

- 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


Kubernetes Master
Developer / Operator
API Server
Controller Manager
Scheduler
etcd
Users
Kubelet
cAdvisor
Kube-Proxy
Pod
Pod
...
Pod
...
Kubelet
cAdvisor
Kube-Proxy
Pod
Pod
...
Pod
Plugin Network (eg Flannel, Weavenet, etc)
Kubernetes Node
Kubernetes Node

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



Time Span: 10 Minutes

Business Transaction:
● All
○ List My Orders
○ Place Order

Name

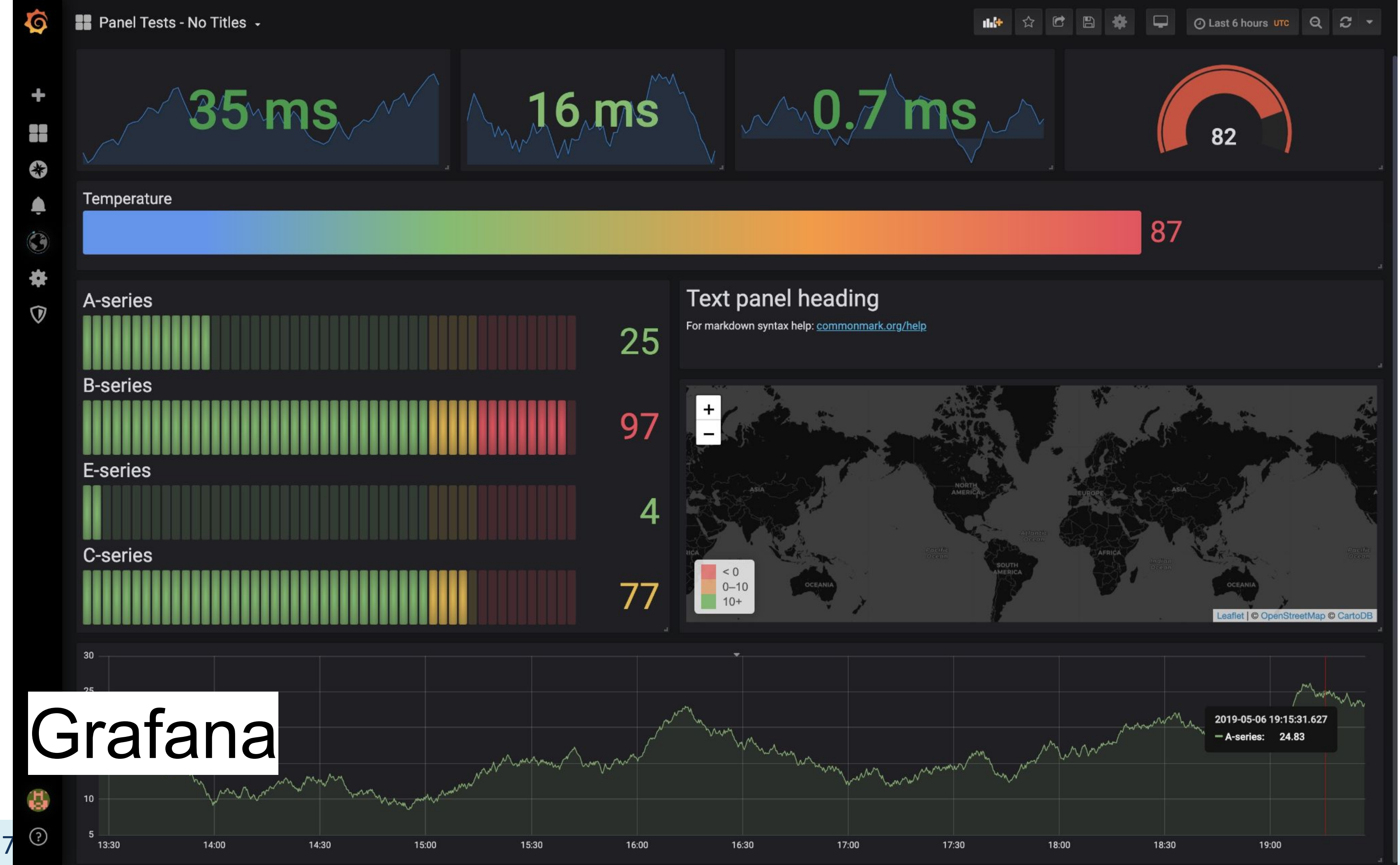
Text: Contains text

Host Name: Enter a host name

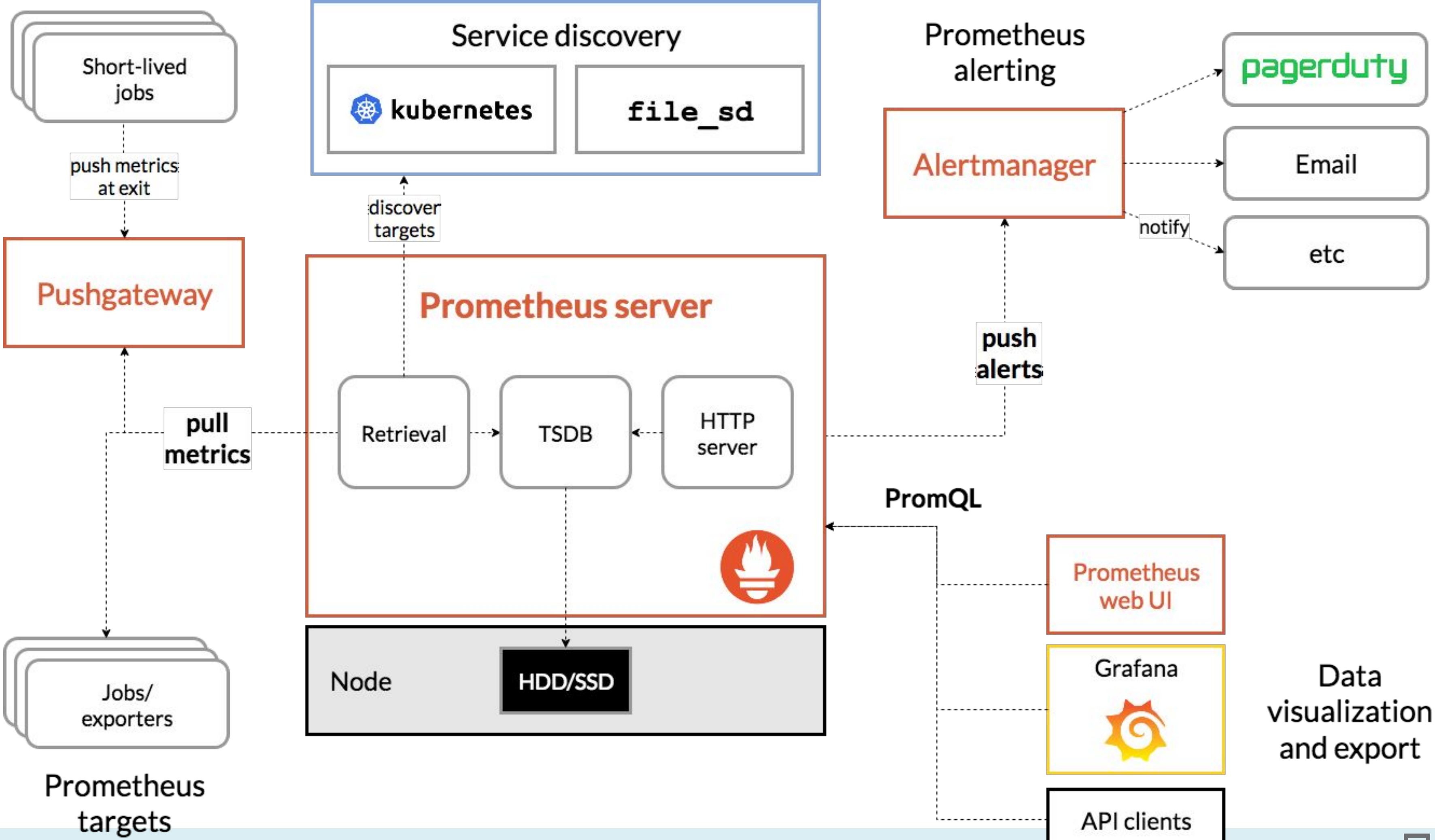
Aggregation Interval: 10 Second

Last Update: 12 Oct 2016 14:35:32

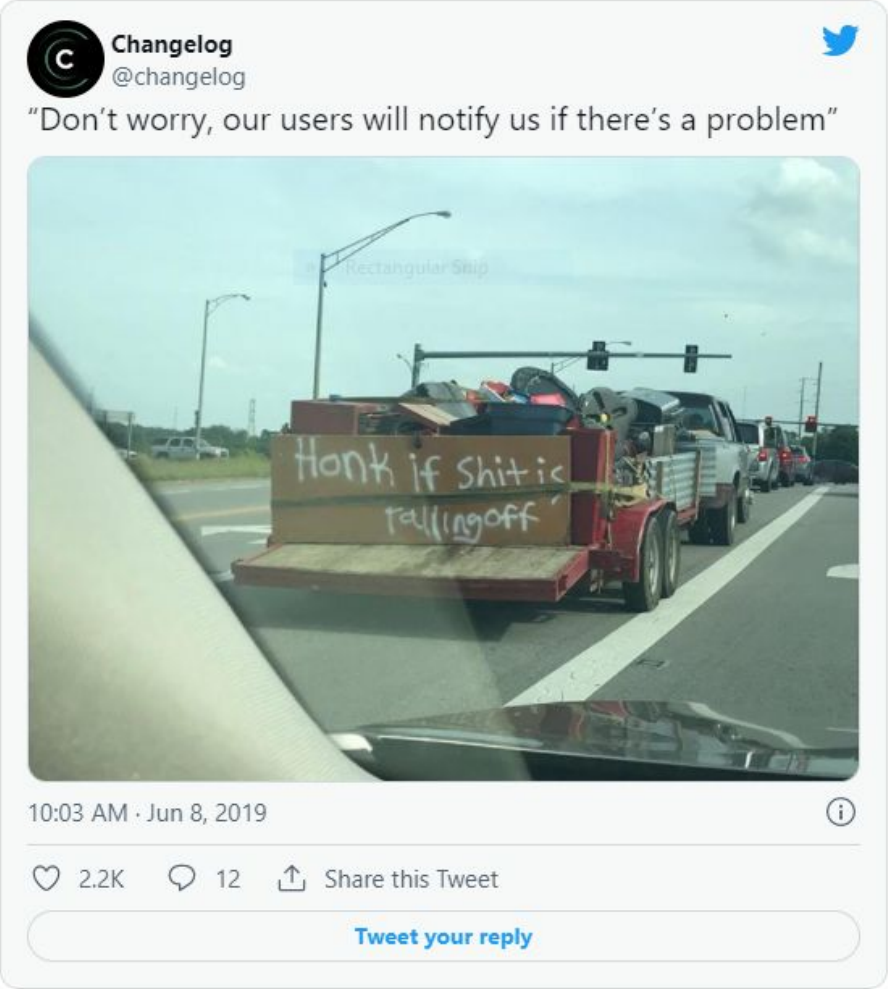
	Actual (secs)	Elapsed (secs)	Count	Component	URI	Operation
	0.000	0.001	1320	consumer	/orders	POST
	0.000	0.001	140	consumer	/orders	GET
	0.000	0.000	1320	consumer		GetAccount
	0.000	0.000	1102	consumer		GetItem
	0.000	0.000	535	consumer		StoreOrder
	0.000	0.000	535	consumer		UpdateQuantity
	0.000	0.000	140	consumer		GetOrders
	0.000	0.000	1102	database	InventoryDB	QueryInventory
	0.000	0.000	535	database	InventoryDB	WriteInventory
	0.000	0.000	1320	database	AccountsDB	RetrieveAccount


Panel Tests - No Titles
Last 6 hours UTC
35 ms
16 ms
0.7 ms
82
Temperature
87
A-series
25
B-series
97
E-series
4
C-series
77
Text panel heading
For markdown syntax help: commonmark.org/help
+
−
< 0
0–10
10+
NORTH AMERICA
SOUTH AMERICA
EUROPE
AFRICA
ASIA
OCEANIA
Leaflet | © OpenStreetMap © CartoDB
30
25
10
5
13:30
14:00
14:30
15:00
15:30
16:00
16:30
17:00
17:30
18:00
18:30
19:00
2019-05-06 19:15:31.627
A-series: 24.83

Grafana


Short-lived jobs
push metrics at exit
Pushgateway
pull metrics
Jobs/ exporters
Prometheus targets
Service discovery
kubernetes
file_sd
discover targets
Prometheus server
Retrieval
TSDB
HTTP server
Node
HDD/SSD
Prometheus alerting
pagerduty
Alertmanager
Email
notify
etc
push alerts
PromQL
Prometheus web UI
Grafana
API clients
Data visualization and export

QA doesn’t stop in Dev: Testing in Production

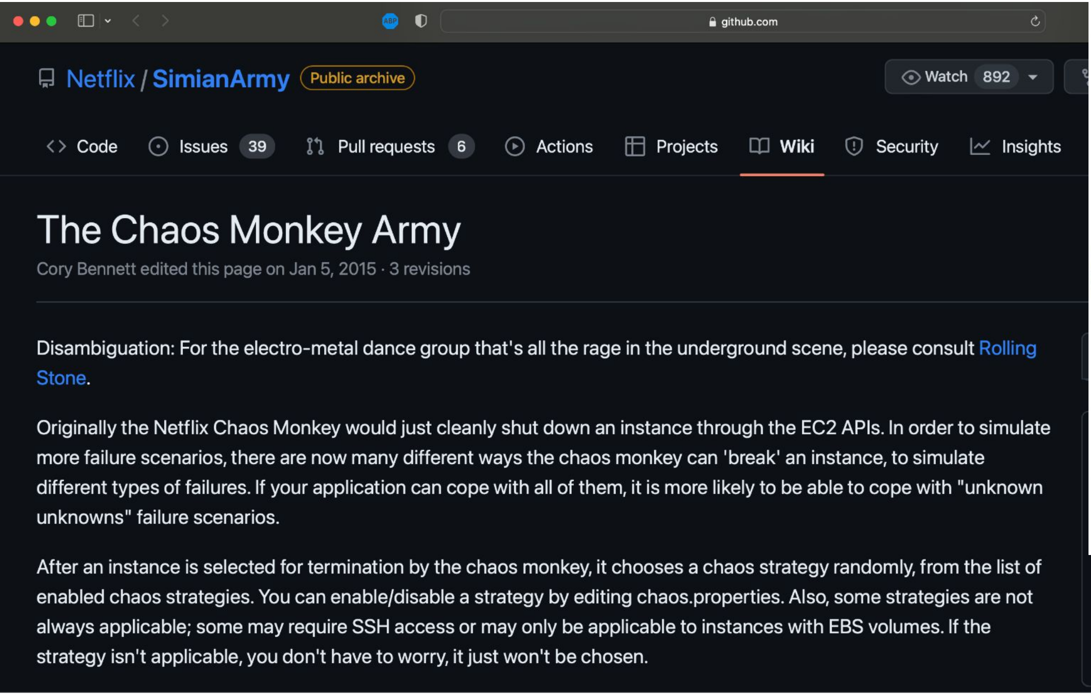


Changelog
@changelog

“Don’t worry, our users will notify us if there’s a problem”

10:03 AM · Jun 8, 2019

Chaos Experiments



The Chaos Monkey Army

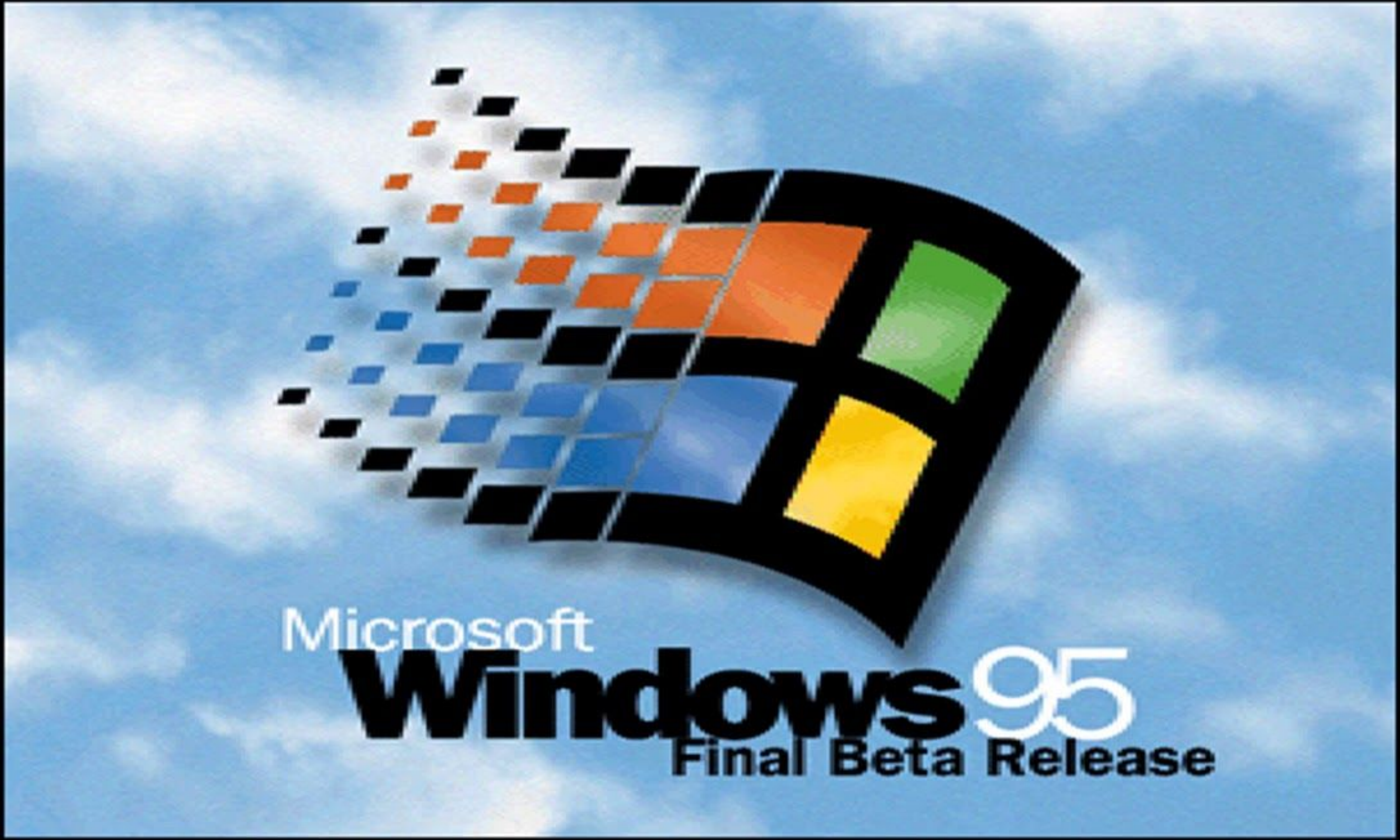
Cory Bennett edited this page on Jan 5, 2015 · 3 revisions

Disambiguation: For the electro-metal dance group that's all the rage in the underground scene, please consult Rolling Stone.

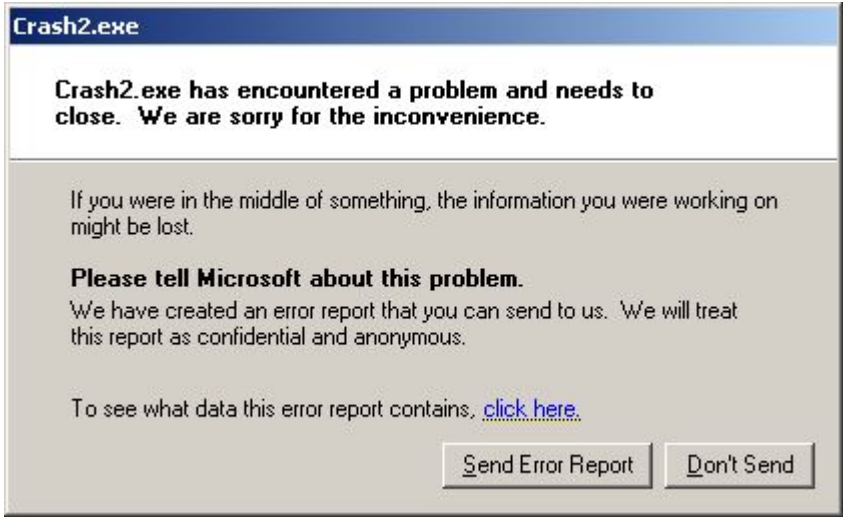
Originally the Netflix Chaos Monkey would just cleanly shut down an instance through the EC2 APIs. In order to simulate more failure scenarios, there are now many different ways the chaos monkey can 'break' an instance, to simulate different types of failures. If your application can cope with all of them, it is more likely to be able to cope with "unknown unknowns" failure scenarios.

After an instance is selected for termination by the chaos monkey, it chooses a chaos strategy randomly, from the list of enabled chaos strategies. You can enable/disable a strategy by editing chaos.properties. Also, some strategies are not always applicable; some may require SSH access or may only be applicable to instances with EBS volumes. If the strategy isn't applicable, you don't have to worry, it just won't be chosen.

- Chaos Monkey
- Janitor Monkey
- Conformity Monkey
- Migration
- Support


Microsoft
Windows 95
Final Beta Release

Crash Telemetry



Crash2.exe has encountered a problem and needs to close. We are sorry for the inconvenience.

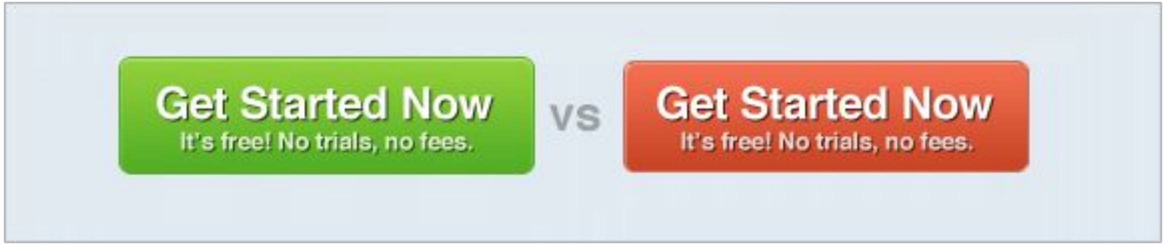
If you were in the middle of something, the information you were working on might be lost.

Please tell Microsoft about this problem.

We have created an error report that you can send to us. We will treat this report as confidential and anonymous.

To see what data this error report contains, click here.

A/B Testing

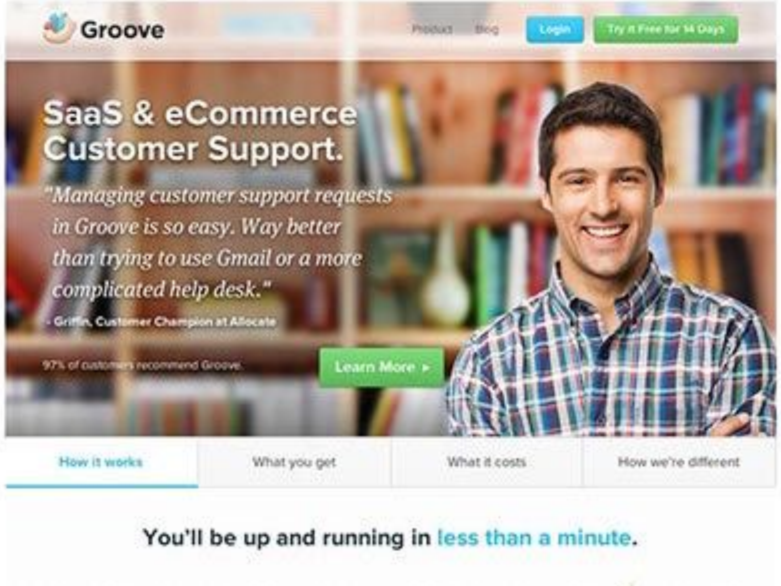


Get Started Now
It's free! No trials, no fees.

vs

Get Started Now
It's free! No trials, no fees.

Original: 2.3%



Groove

Product Blog Login Try it Free for 14 Days

SaaS & eCommerce Customer Support.

"Managing customer support requests in Groove is so easy. Way better than trying to use Gmail or a more complicated help desk."

- Griffin, Customer Champion at Allocate

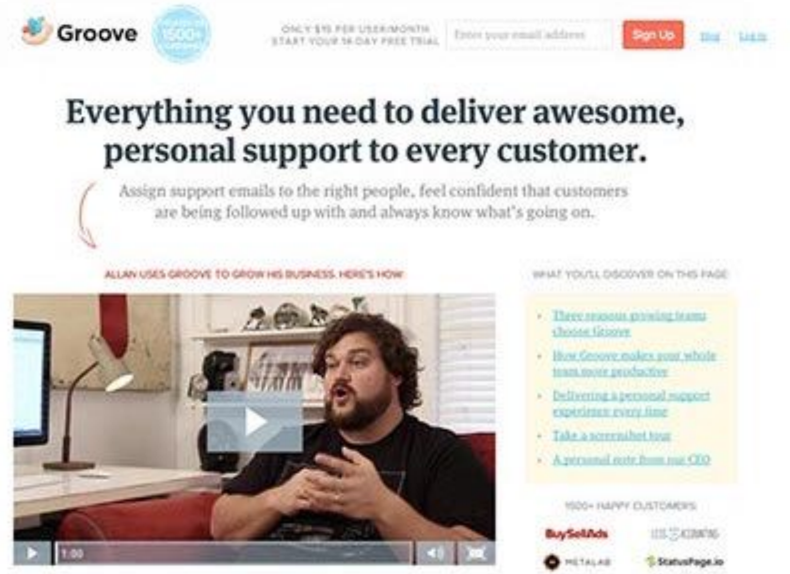
97% of customers recommend Groove.

Learn More ▸

How it works | What you get | What it costs | How we're different

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

Long Form: 4.3%



Groove

ONLY $15 PER USER/MONTH
START YOUR 14-DAY FREE TRIAL

Enter your email address

Sign Up

Everything you need to deliver awesome, personal support to every customer.

Assign support emails to the right people, feel confident that customers are being followed up with and always know what's going on.

ALLAN USES GROOVE TO GROW HIS BUSINESS. HERE'S HOW

1:00

WHAT YOU'LL DISCOVER ON THIS PAGE

- Three reasons growing teams choose Groove
- How Groove makes your whole team more productive
- Delivering a personal support experience every time
- Take a screenshot tour
- A personal note from our CEO

1500+ HAPPY CUSTOMERS

BuySellAds
StatusPage.io
METALAB

What If

... we had 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
- 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

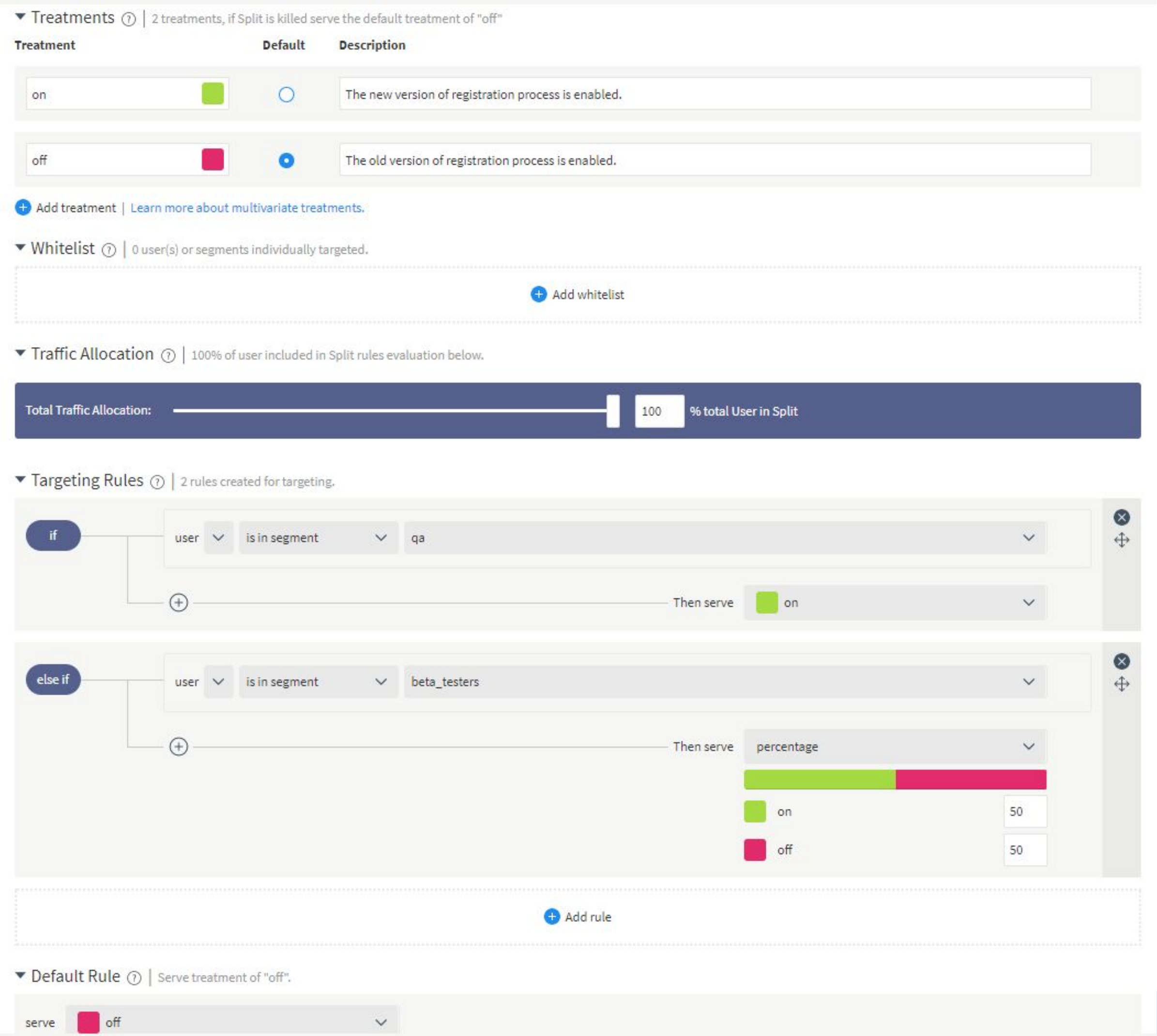
Good practices: tracked explicitly, documented, keep them localized and independent

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 🟩	○	The new version of registration process is enabled.
off 🟥	●	The old version of registration process is enabled.

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

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

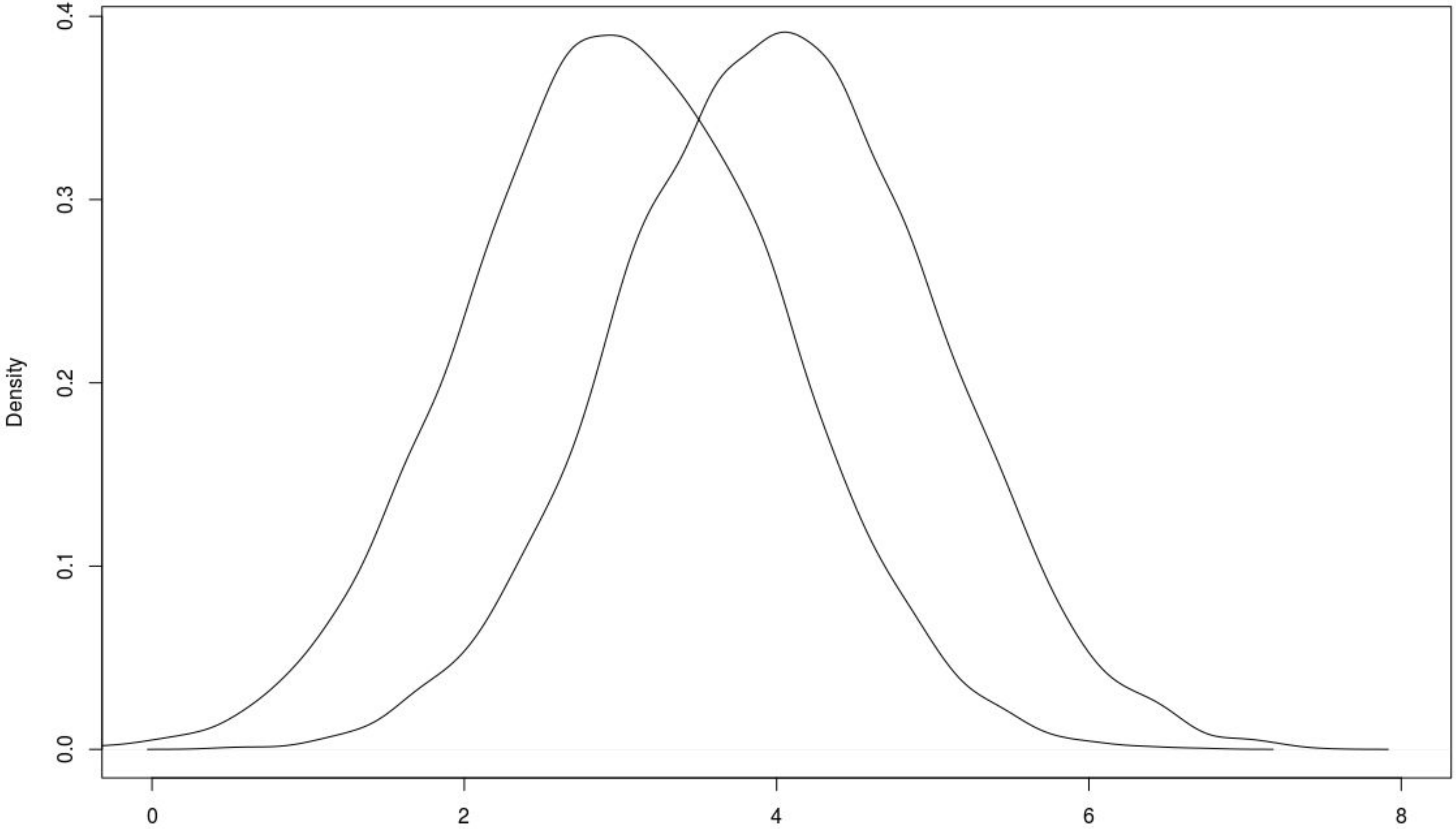
🟥 off 50

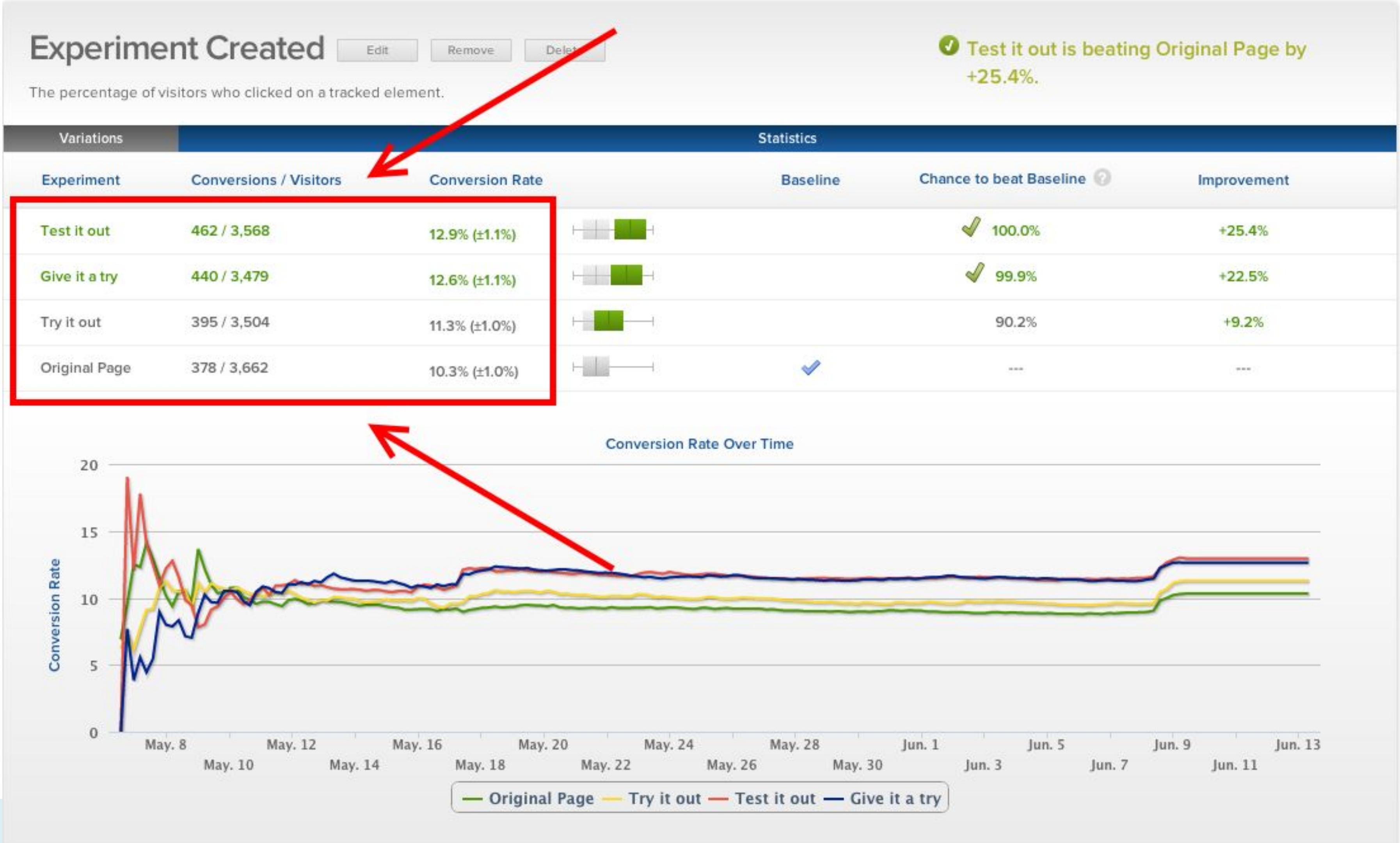
Default Rule ⓘ | Serve treatment of "off".

serve 🟥 off

Comparing Outcomes

Group A	Group B
base game	game with extra god cards
2158 Users	10 Users
average 18:13 min time on site	average 20:24 min time on site


Density
0.0
0.1
0.2
0.3
0.4
0
2
4
6
8



Experiment Created

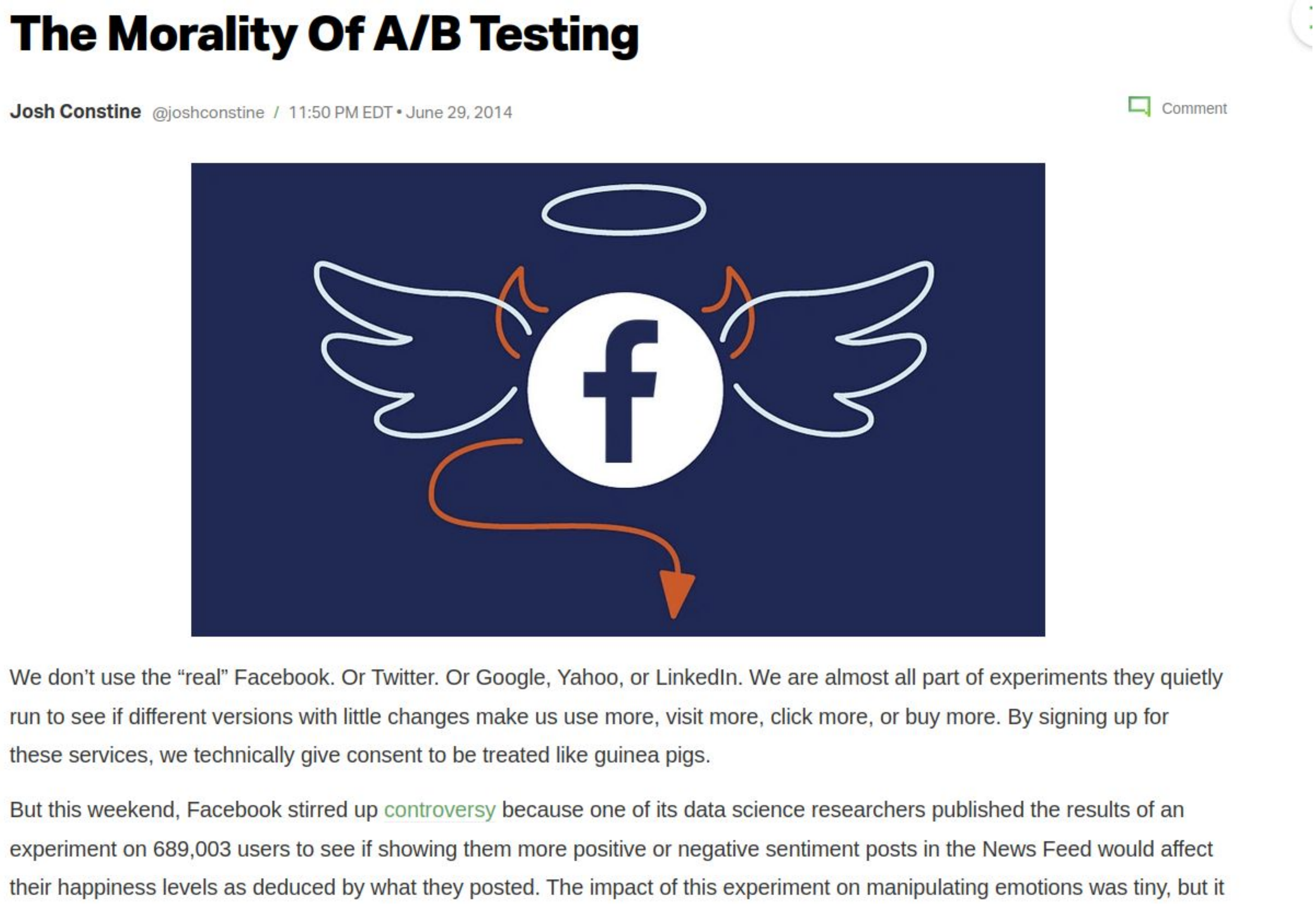
Edit Remove Delete

✓ 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

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

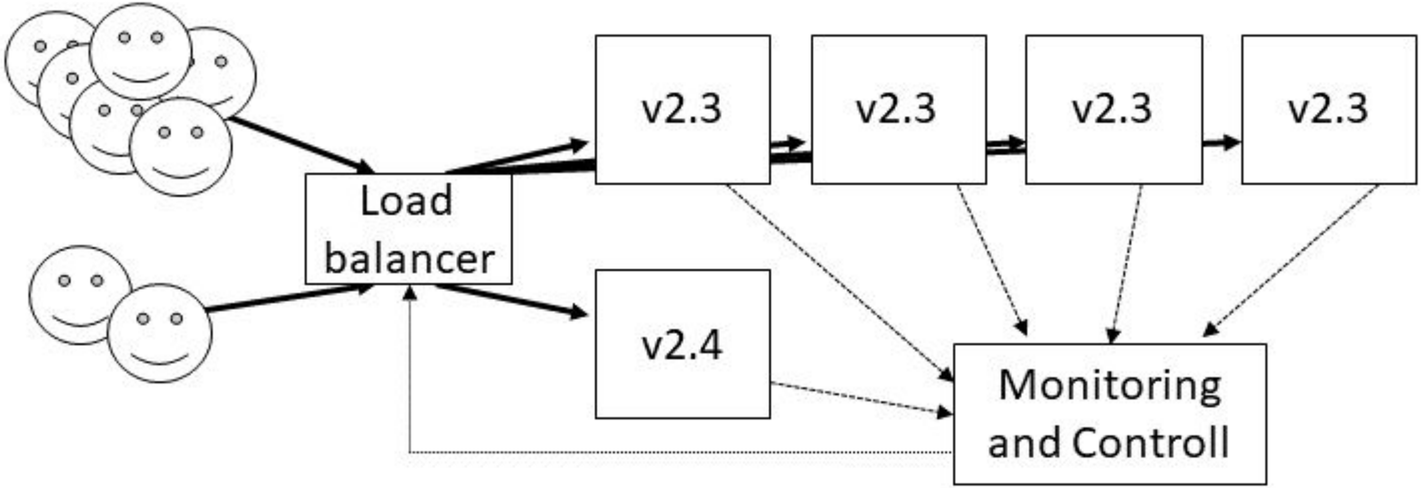
https://techcrunch.com/2014/06/29/ethics-in-a-data-driven-world/

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.

TAing in Fall 2023?

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/F23/ (select 17214)

Summary

Increasing automation of tests and deployments

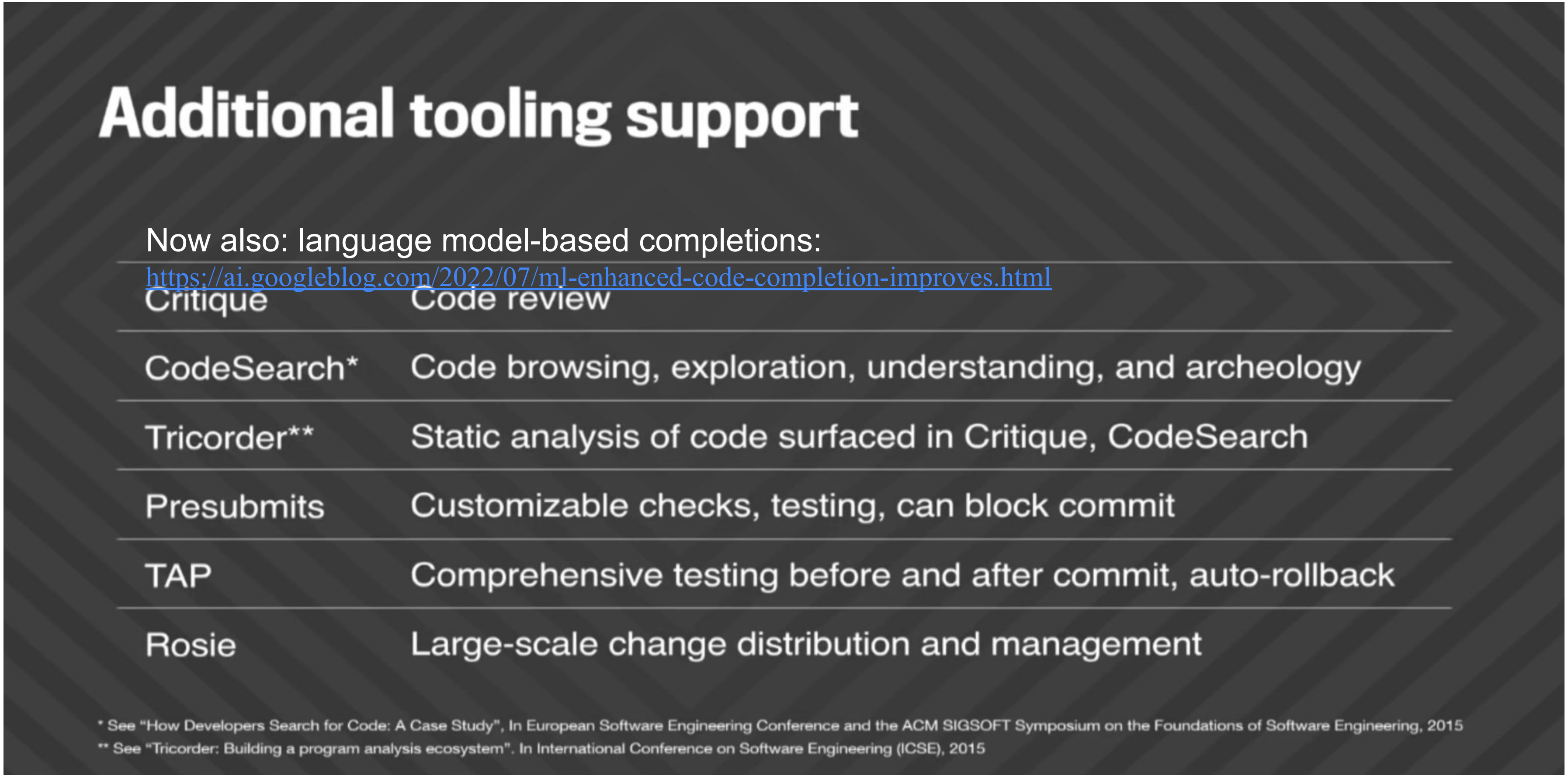
Containers and configuration management tools help with automation, deployment, and rollbacks

Monitoring becomes important

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

Bonus: You need smarter tools to operate at modern scale

1. Lots of automation (example from Google)



Additional tooling support

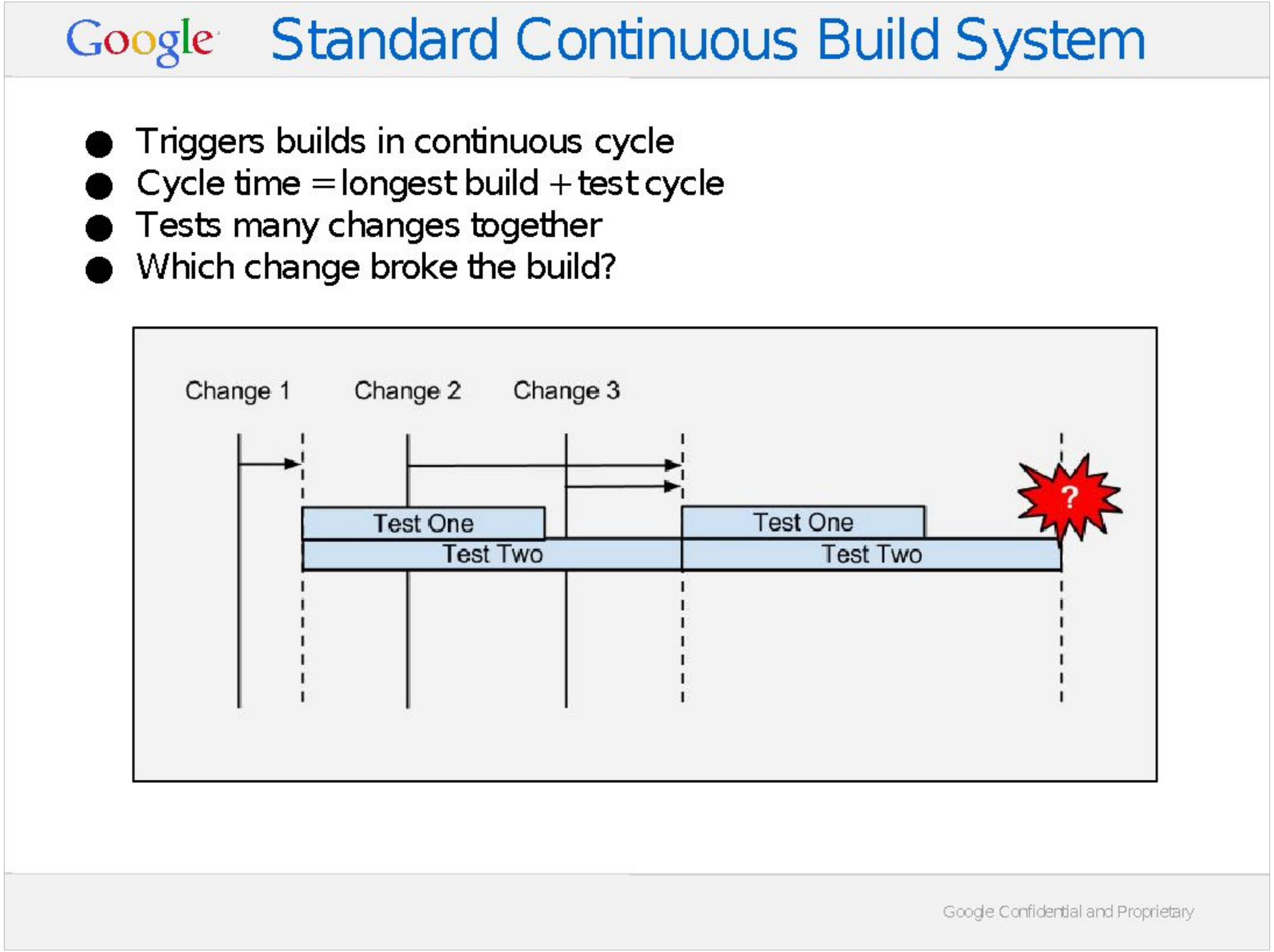
Now also: language model-based completions: https://ai.googleblog.com/2022/07/ml-enhanced-code-completion-improves.html

Critique	Code review
CodeSearch*	Code browsing, exploration, understanding, and archeology
Tricorder**	Static analysis of code surfaced in Critique, CodeSearch
Presubmits	Customizable checks, testing, can block commit
TAP	Comprehensive testing before and after commit, auto-rollback
Rosie	Large-scale change distribution and management

* See "How Developers Search for Code: A Case Study", In European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, 2015

** See "Tricorder: Building a program analysis ecosystem". In International Conference on Software Engineering (ICSE), 2015

2. Build system

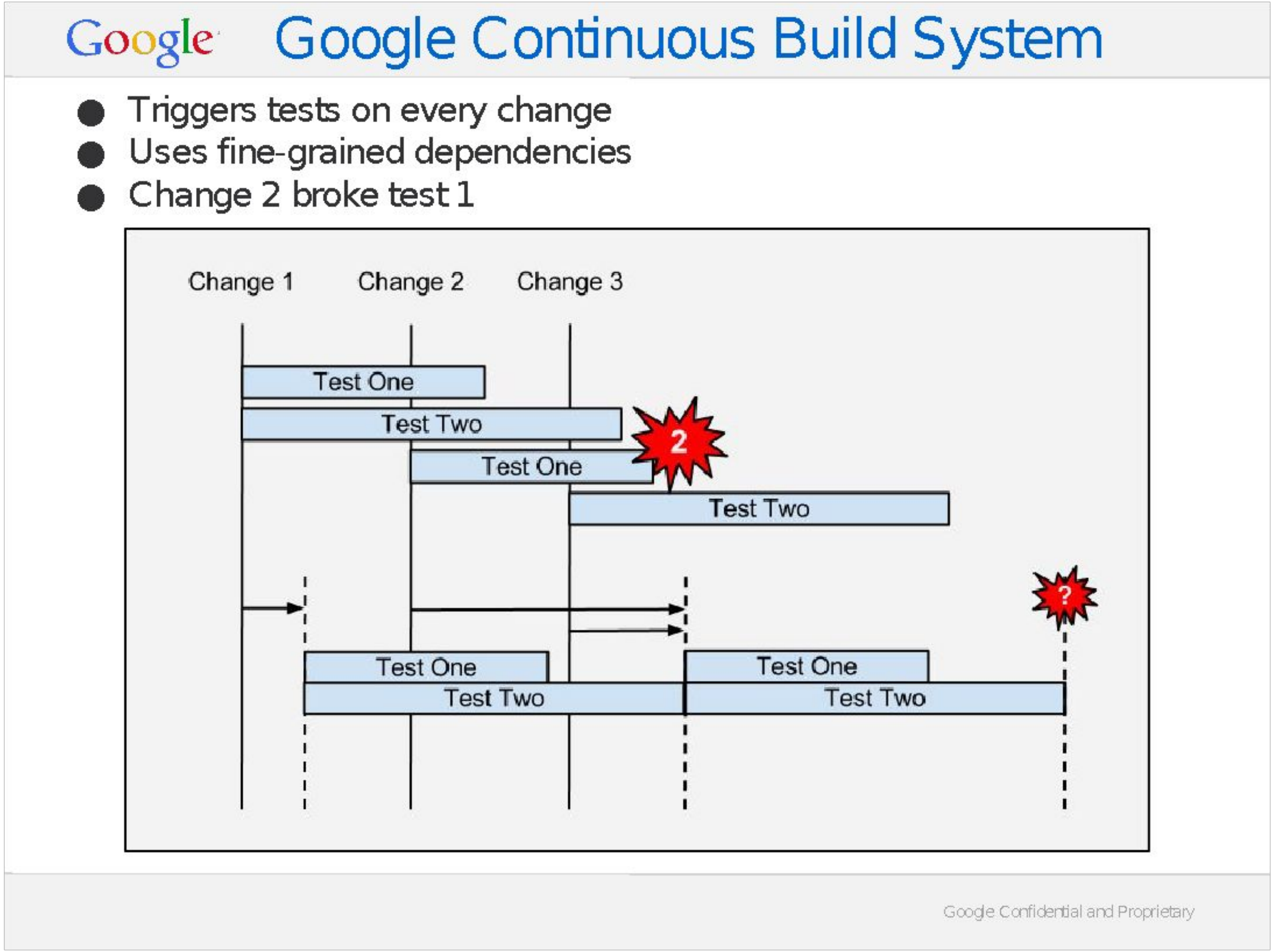


Google Standard Continuous Build System

- Triggers builds in continuous cycle
- Cycle time = longest build + test cycle
- Tests many changes together
- Which change broke the build?

Google Confidential and Proprietary

2. Build system

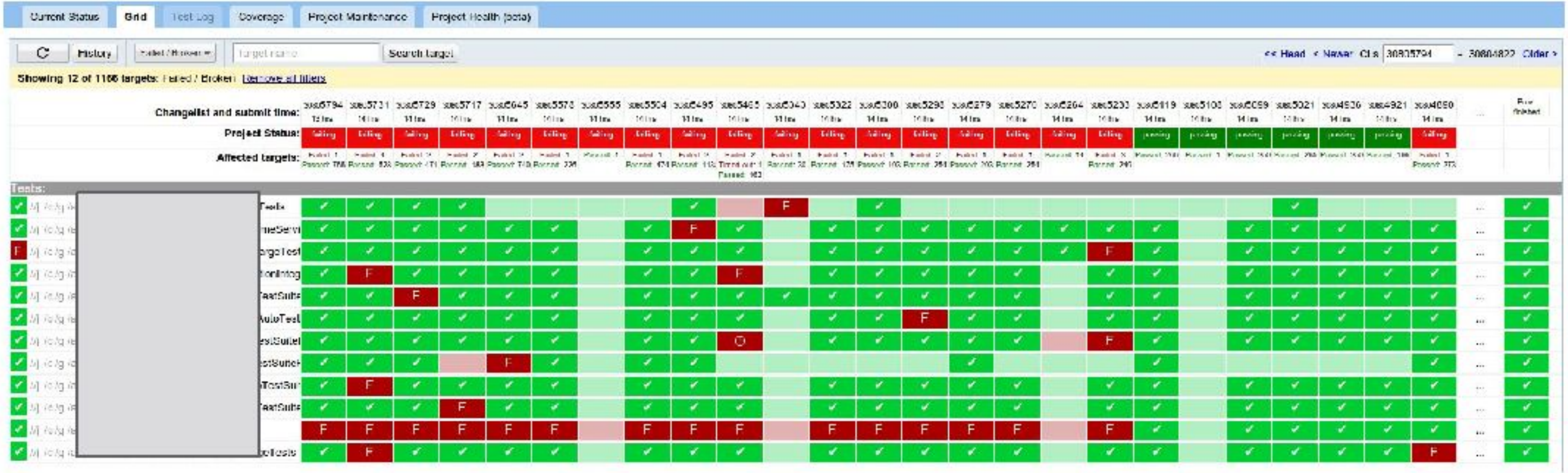


Google Continuous Build System

- Triggers tests on every change
- Uses fine-grained dependencies
- Change 2 broke test 1

Google Confidential and Proprietary

Google Continuous Integration Display



Current Status | Grid | Test Log | Coverage | Project Maintenance | Project Health (beta)

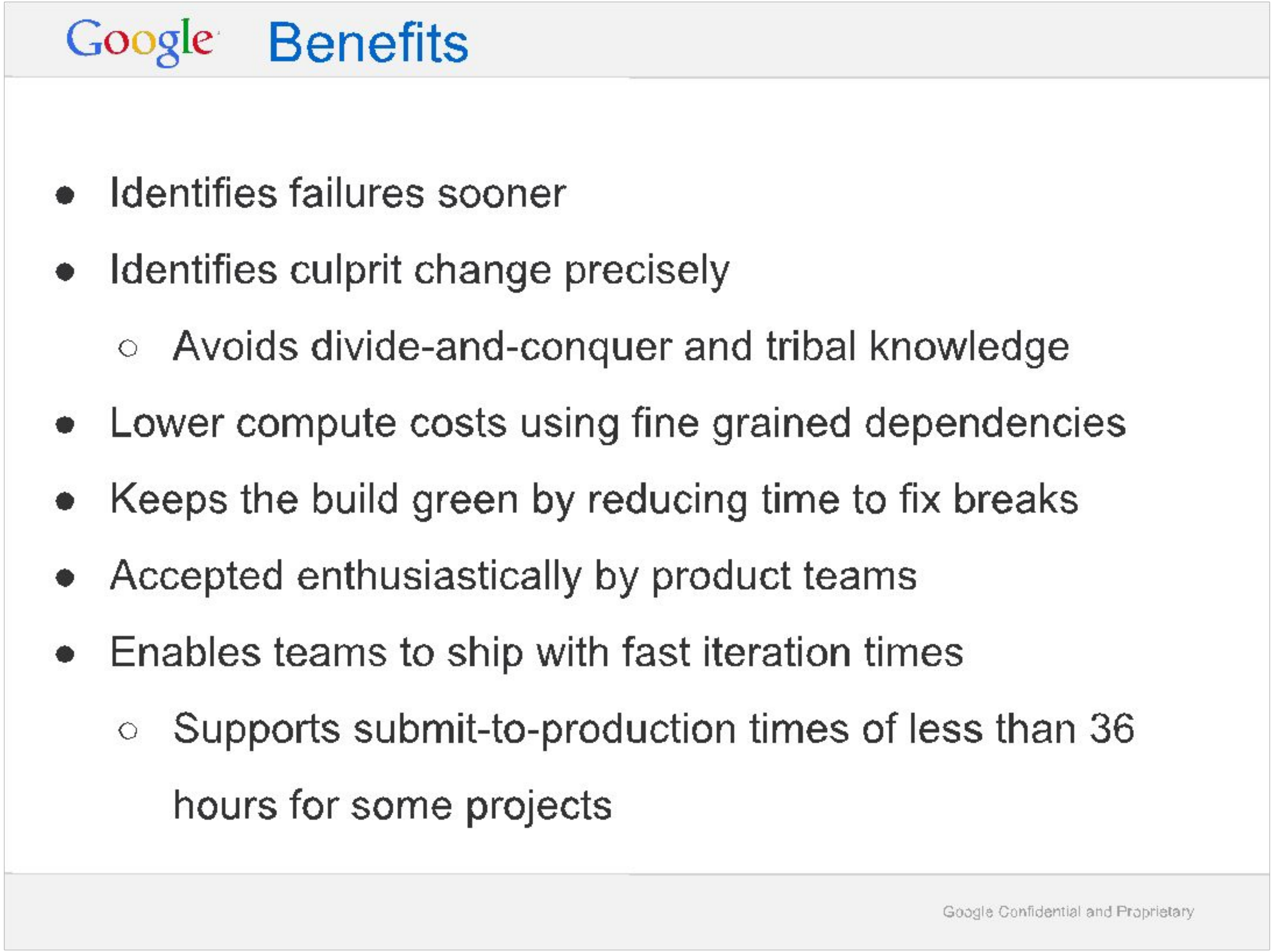
C | History | Failed / Broken ▾ | target name | Search target

<< Head < Newer CLs 30805794 - 30804822 Older >

Showing 12 of 1166 targets: Failed / Broken Remove all filters

Changelist and submit time:	30805794 14 hrs	30805731 16 hrs	30805729 14 hrs	30805717 16 hrs	30805645 14 hrs	30805578 16 hrs	30805555 14 hrs	30805504 16 hrs	30805495 14 hrs	30805465 16 hrs	30805340 14 hrs	30805322 16 hrs	30805300 14 hrs	30805293 16 hrs	30805279 14 hrs	30805270 16 hrs	30805264 14 hrs	30805233 16 hrs	30805119 14 hrs	30805108 16 hrs	30805099 14 hrs	30805021 16 hrs	30804926 14 hrs	30804921 16 hrs	30804850 14 hrs	...	Run finished
Project Status:	failing	failing	failing	failing	failing	failing	failing	failing	failing	failing	failing	failing	failing	failing	failing	failing	failing	failing	passing	passing	passing	passing	passing	passing	failing		
Affected targets:	Failed 1 Passed 786	Failed 4 Passed 533	Failed 3 Passed 471	Failed 2 Passed 183	Failed 3 Passed 749	Failed 1 Passed 236	Passed 1	Failed 1 Passed 474	Failed 3 Passed 113	Failed 2 Timed out 1 Passed 183	Failed 5 Passed 32	Failed 7 Passed 125	Failed 5 Passed 103	Failed 2 Passed 284	Failed 5 Passed 203	Failed 7 Passed 284	Passed 74	Failed 3 Passed 246	Passed 248	Passed 1	Passed 254	Passed 286	Passed 254	Passed 186	Failed 1 Passed 373		
Tests:																											
✓ //j/c/g/ [redacted] Tests	✓	✓	✓	✓					✓		F		✓									✓				...	✓
✓ //j/c/g/ [redacted] meServi	✓	✓	✓	✓	✓	✓		✓	F	✓		✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	...	✓
F //j/c/g/ [redacted] argeTest	✓	✓	✓	✓	✓	✓		✓	✓	✓		✓	✓	✓	✓	✓	✓	F	✓		✓	✓	✓	✓	✓	...	✓
✓ //j/c/g/ [redacted] tonInteg	✓	F	✓	✓	✓	✓		✓	✓	F		✓	✓	✓	✓	✓		✓	✓		✓	✓	✓	✓	✓	...	✓
✓ //j/c/g/ [redacted] TestSuite	✓	✓	F	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓		✓	✓	✓	✓	✓	...	✓
✓ //j/c/g/ [redacted] AutoTest	✓	✓	✓	✓	✓	✓		✓	✓	✓		✓	✓	F	✓	✓		✓	✓		✓	✓	✓	✓	✓	...	✓
✓ //j/c/g/ [redacted] estSuite	✓	✓	✓	✓	✓	✓		✓	✓	O		✓	✓	✓	✓	✓		F	✓		✓	✓	✓	✓	✓	...	✓
✓ //j/c/g/ [redacted] estSuite	✓	✓	✓		F	✓		✓	✓						✓				✓						✓	...	✓
✓ //j/c/g/ [redacted] TestSui	✓	F	✓	✓	✓	✓		✓	✓	✓		✓	✓	✓	✓	✓		✓	✓		✓	✓	✓	✓	✓	...	✓
✓ //j/c/g/ [redacted] TestSuite	✓	✓	✓	F	✓	✓		✓	✓	✓		✓	✓	✓	✓	✓		✓	✓		✓	✓	✓	✓	✓	...	✓
✓ //j/c/g/ [redacted]	F	F	F	F	F	F		F	F	F		F	F	F	F	F		F	✓		✓	✓	✓	✓	✓	...	✓
✓ //j/c/g/ [redacted] eTests	✓	F	✓	✓	✓	✓		✓	✓	✓		✓	✓	✓	✓	✓		✓	✓		✓	✓	✓	✓	F	...	✓

2. Build system

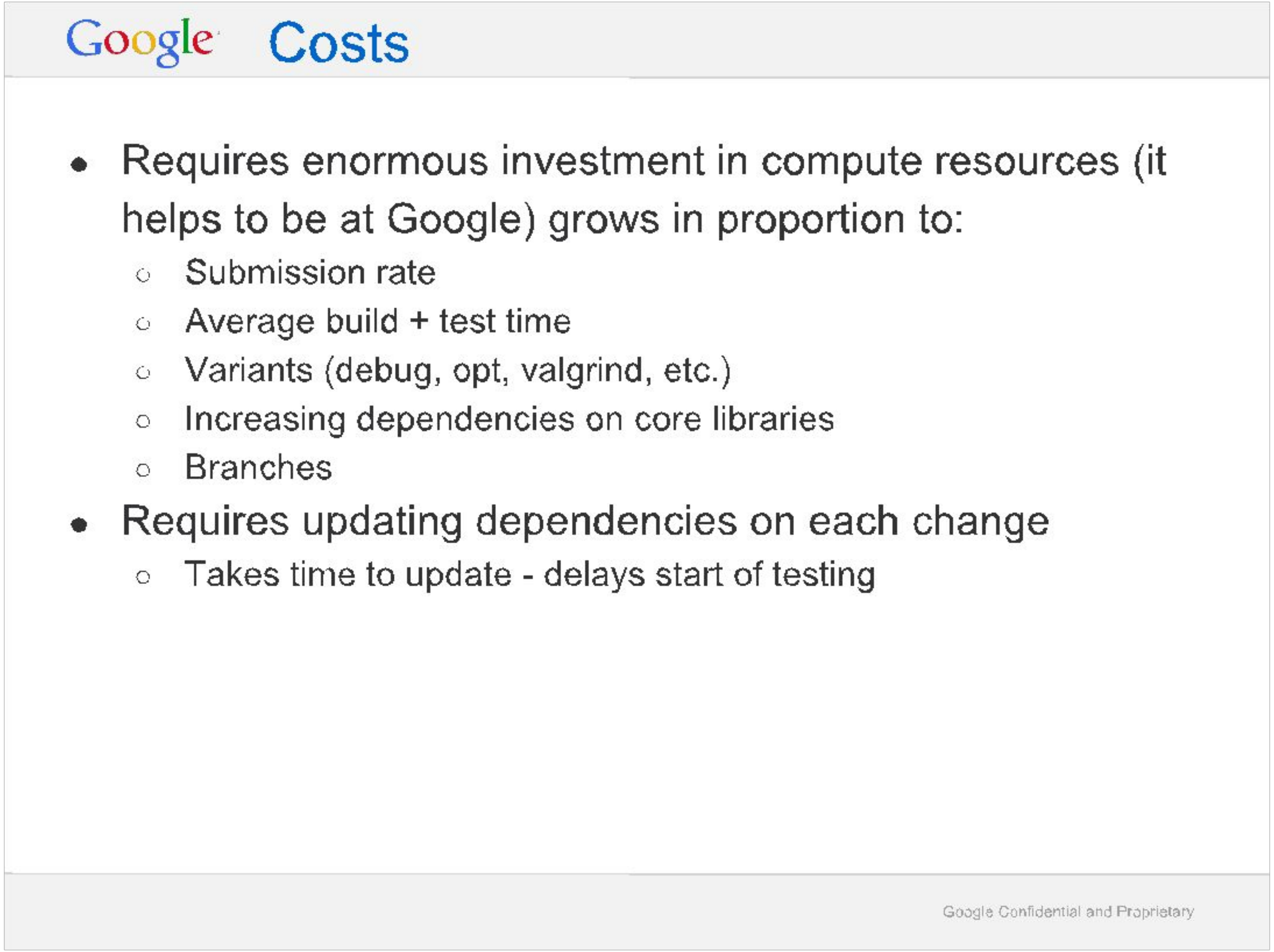


Google Benefits

- Identifies failures sooner
- Identifies culprit change precisely
  - Avoids divide-and-conquer and tribal knowledge
- Lower compute costs using fine grained dependencies
- Keeps the build green by reducing time to fix breaks
- Accepted enthusiastically by product teams
- Enables teams to ship with fast iteration times
  - Supports submit-to-production times of less than 36 hours for some projects

Google Confidential and Proprietary

2. Build system



Google Costs

- Requires enormous investment in compute resources (it helps to be at Google) grows in proportion to:
  - Submission rate
  - Average build + test time
  - Variants (debug, opt, valgrind, etc.)
  - Increasing dependencies on core libraries
  - Branches
- Requires updating dependencies on each change
  - Takes time to update - delays start of testing

Google Confidential and Proprietary

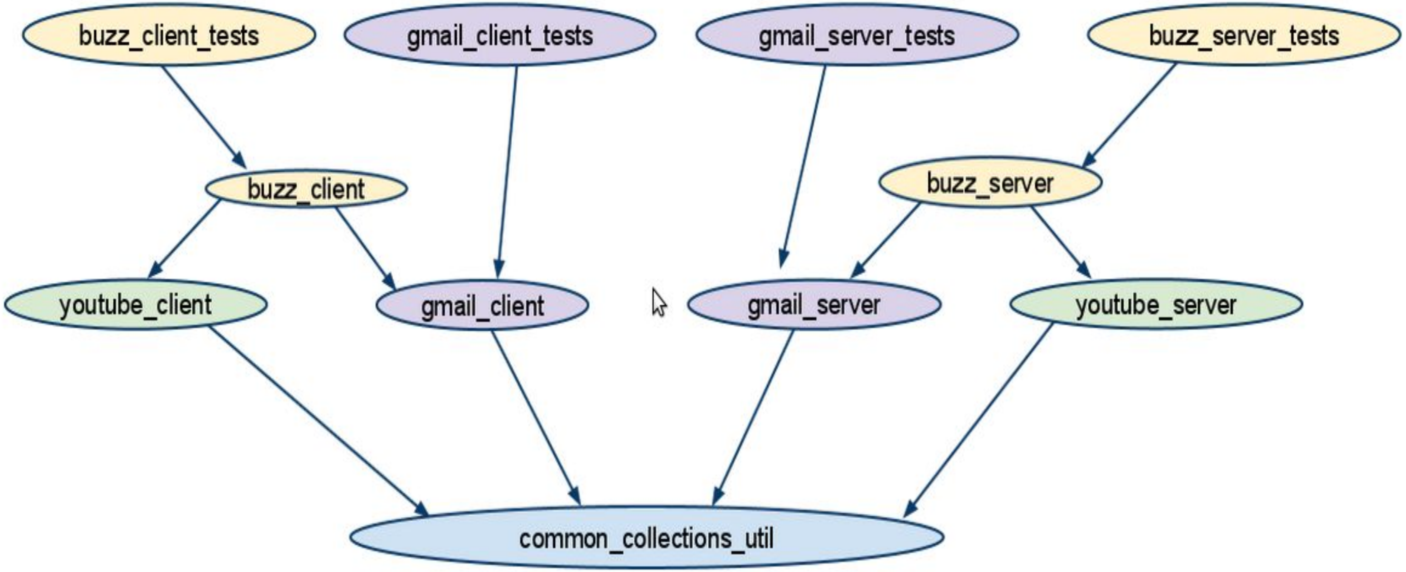
Which tests to run?

GMAIL

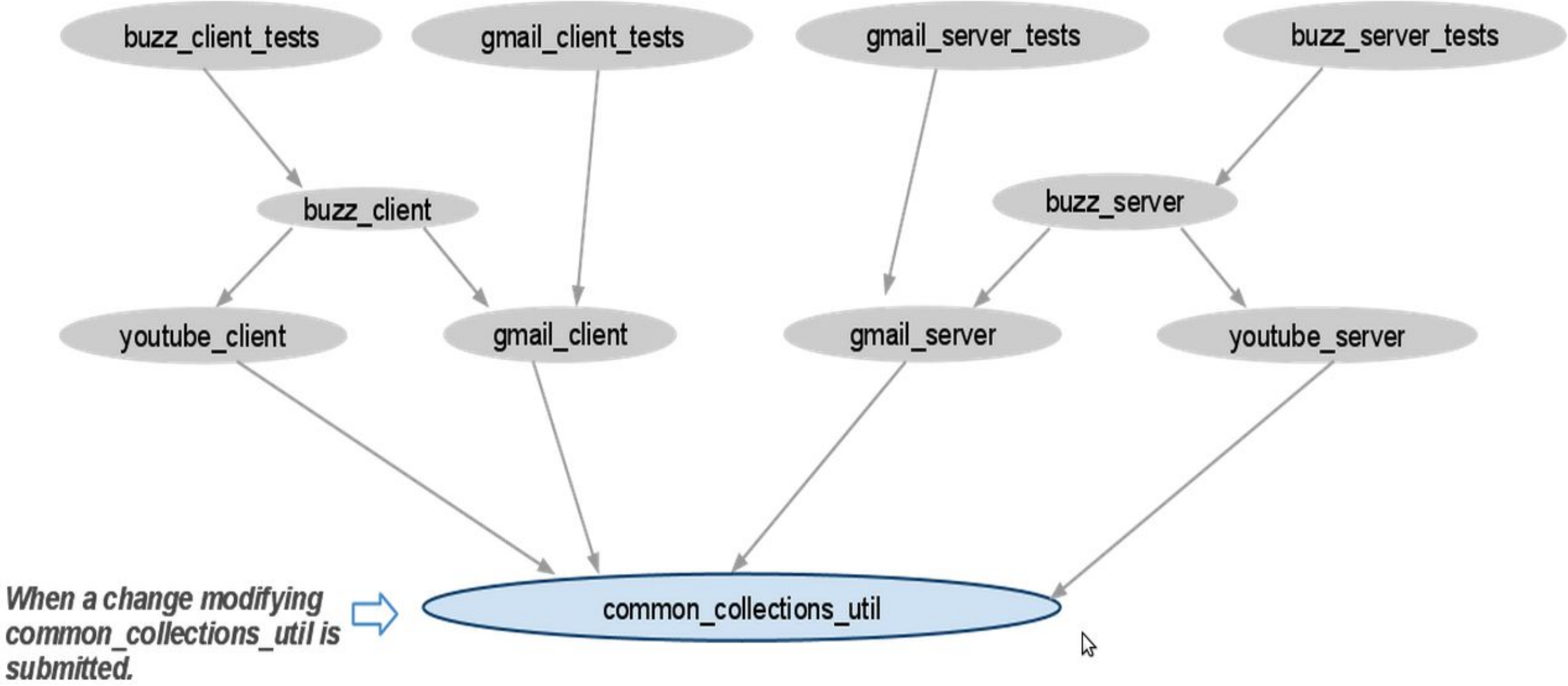
Test Target:
name: //depot/gmail_client_tests
name: //depot/gmail_server_tests

BUZZ

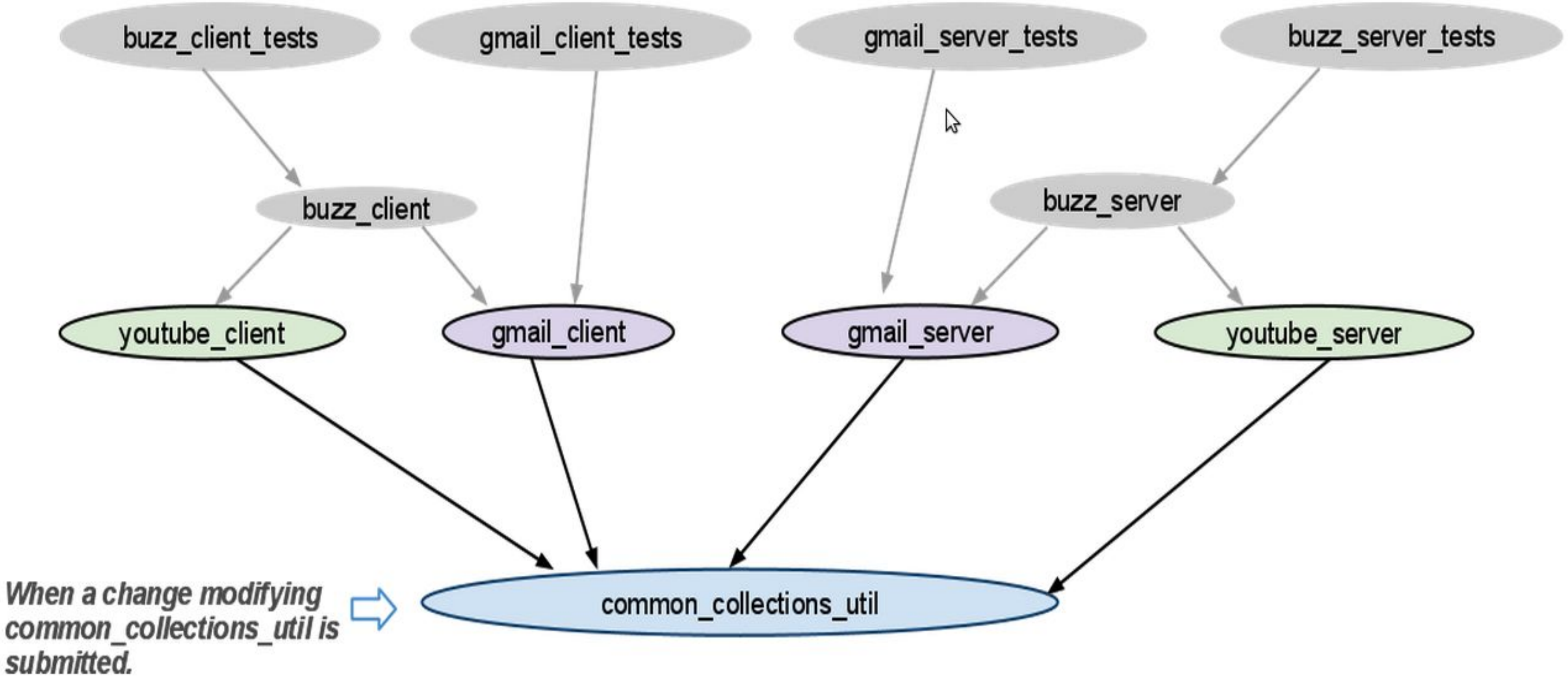
Test targets:
name: //depot/buzz_server_tests
name: //depot/buzz_client_tests



Scenario 1: a change modifies common_collections_util

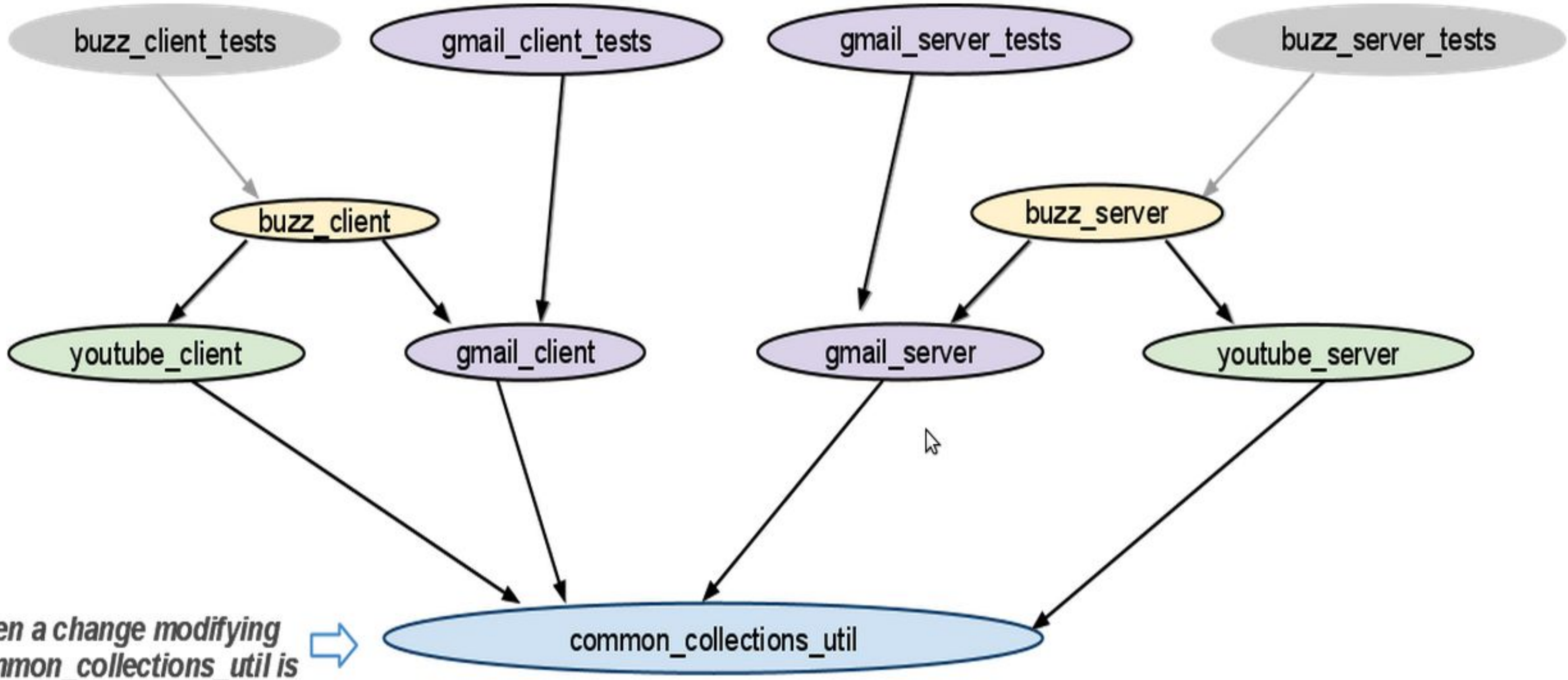


Scenario 1: a change modifies common_collections_util



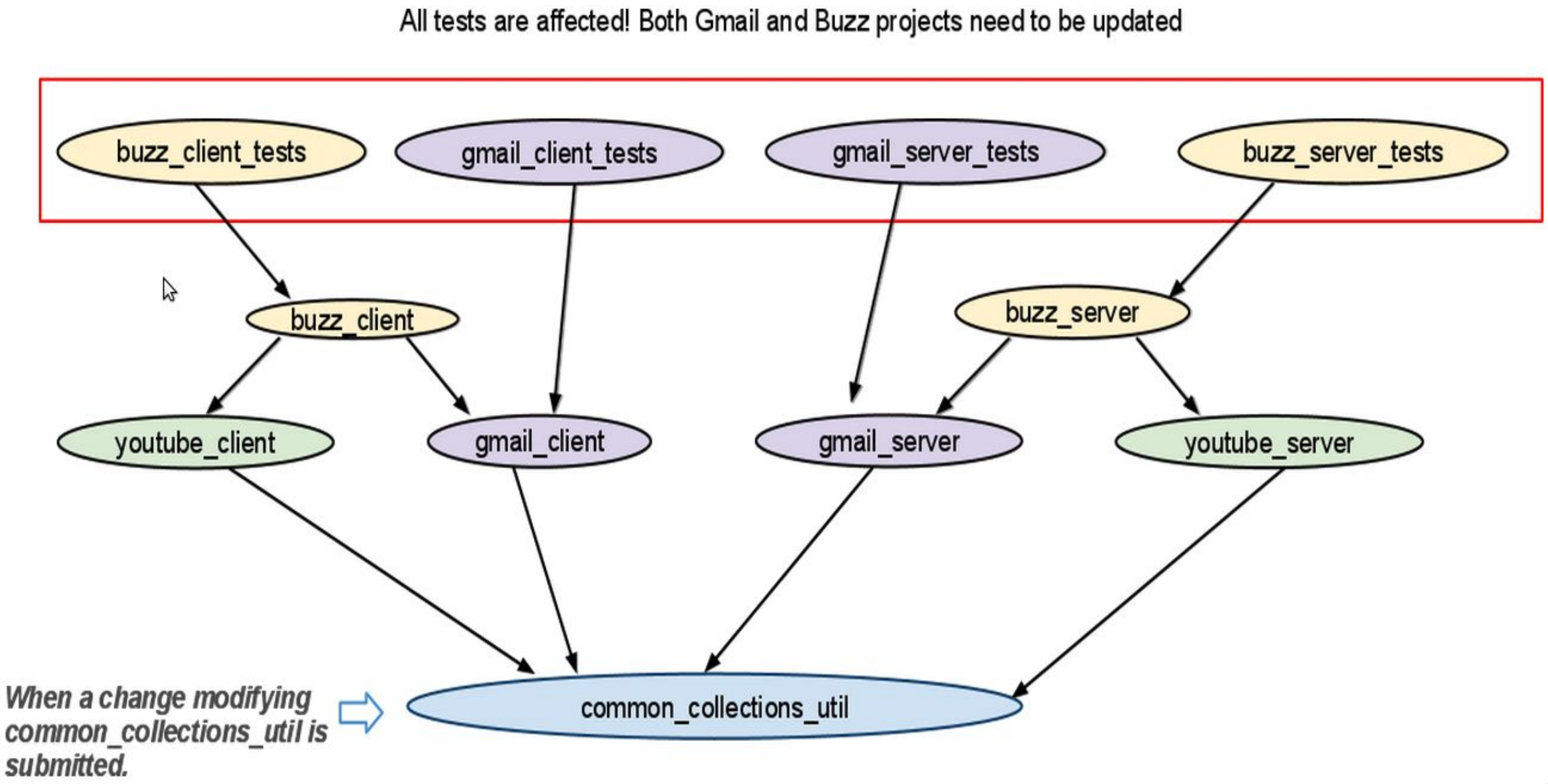
When a change modifying common_collections_util is submitted.

Scenario 1: a change modifies common_collections_util

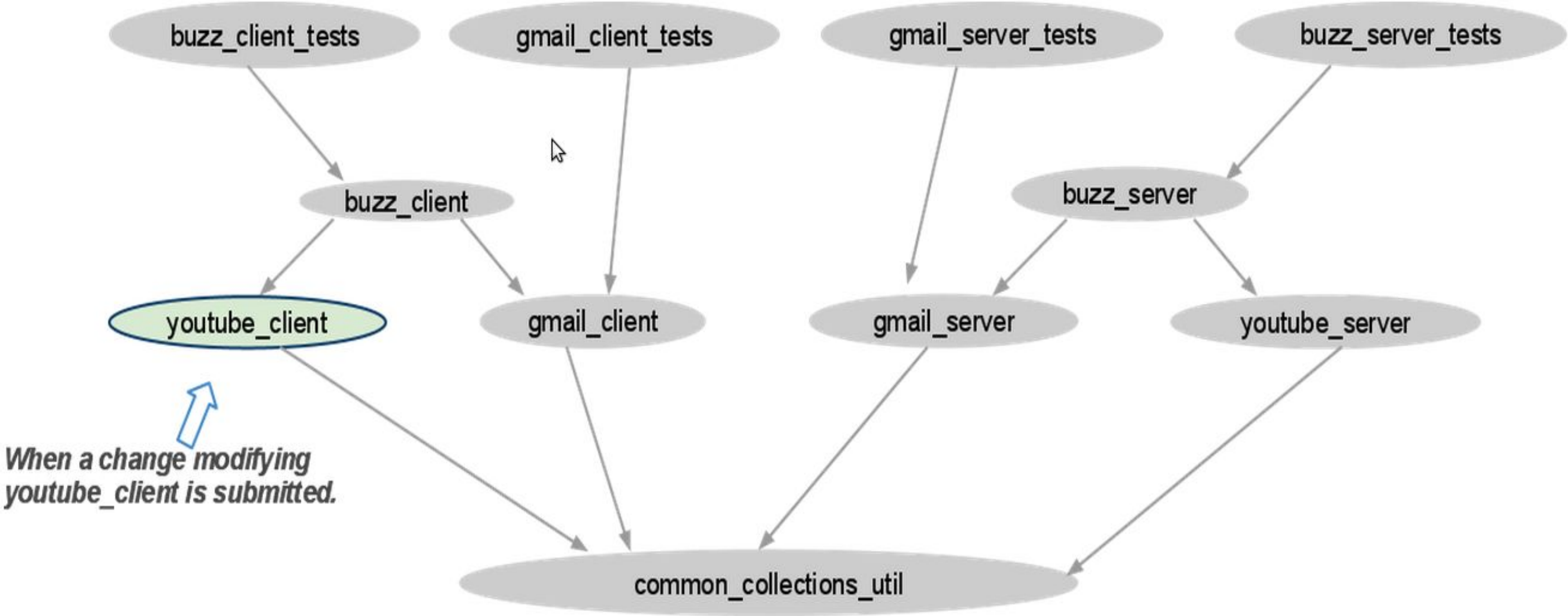


When a change modifying common_collections_util is submitted.

Scenario 1: a change modifies common_collections_util

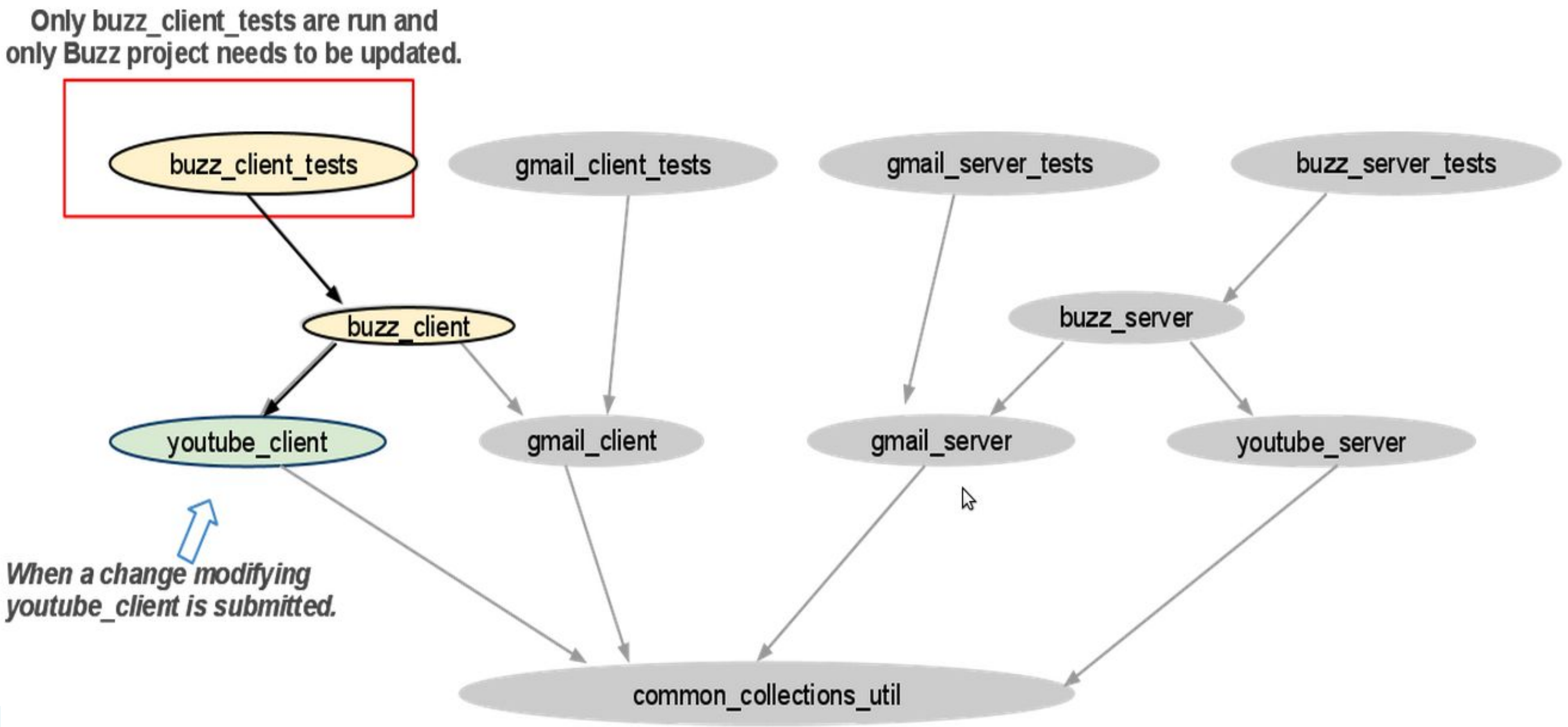


Scenario 2: a change modifies the youtube_client



When a change modifying youtube_client is submitted.

Scenario 2: a change modifies the youtube_clie

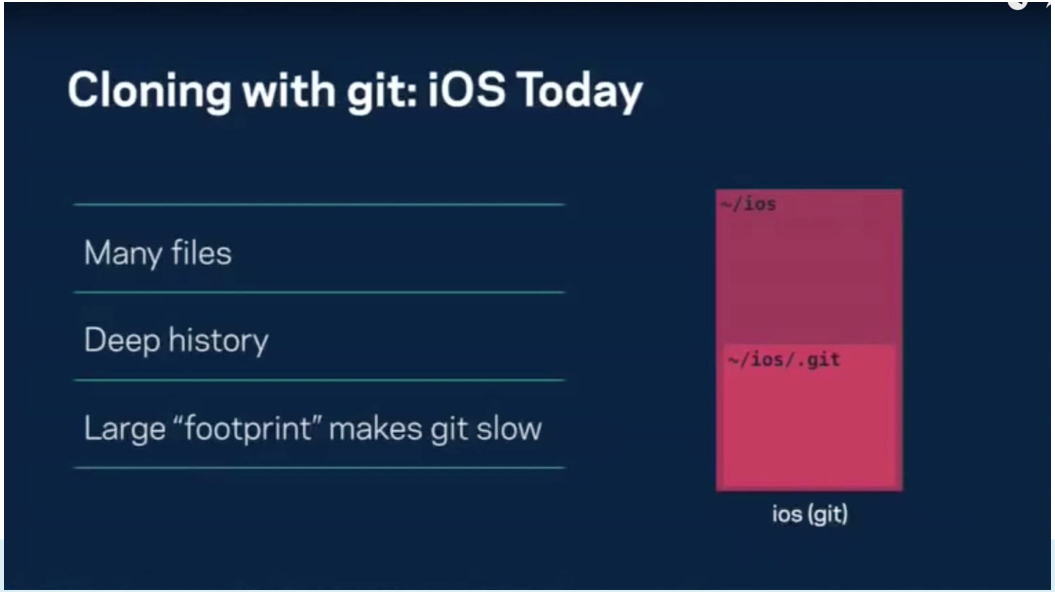


Only buzz_client_tests are run and only Buzz project needs to be updated.

When a change modifying youtube_client is submitted.

3. Version control

- Problem: even git can get slow at Facebook scale
  - 1M+ source control commands run per day
  - 100K+ commits per week



Cloning with git: iOS Today

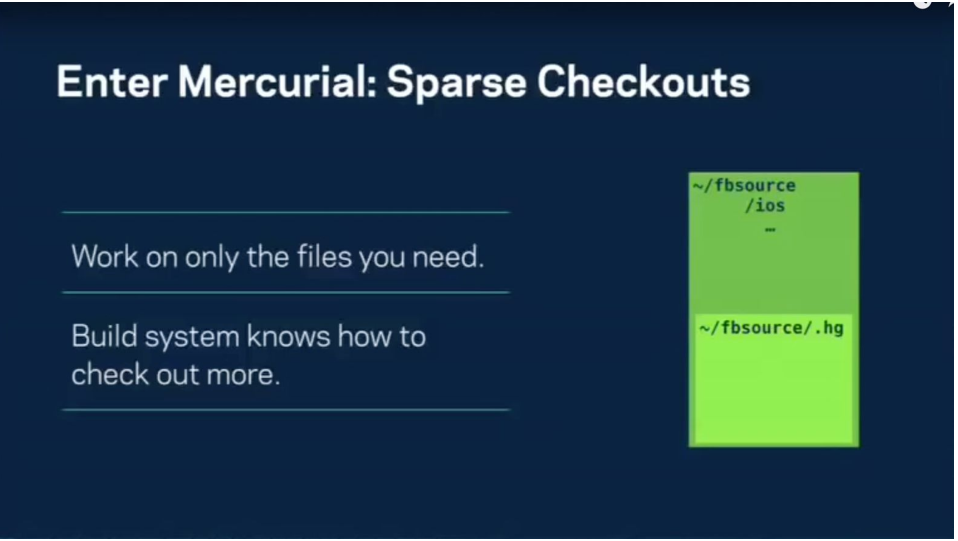
Many files

Deep history

Large "footprint" makes git slow

3. Version control

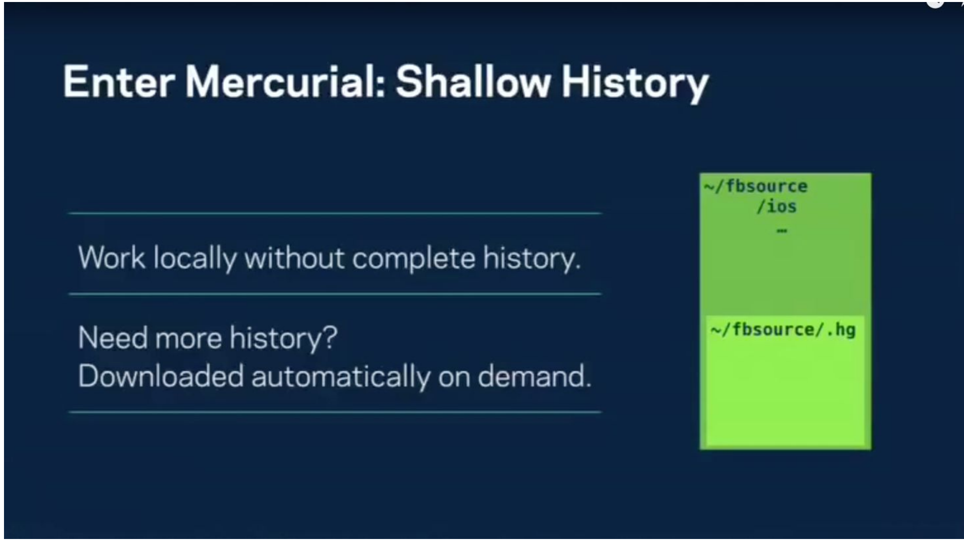
- Solution: redesign version control
  - Sparse checkouts: only fetch metadata (lightweight), get source on-demand
  - Don’t fetch entire history. Can do this with git too (git clone --depth=1), but won’t work for distributed collaboration



Enter Mercurial: Sparse Checkouts

Work on only the files you need.

Build system knows how to check out more.



Enter Mercurial: Shallow History

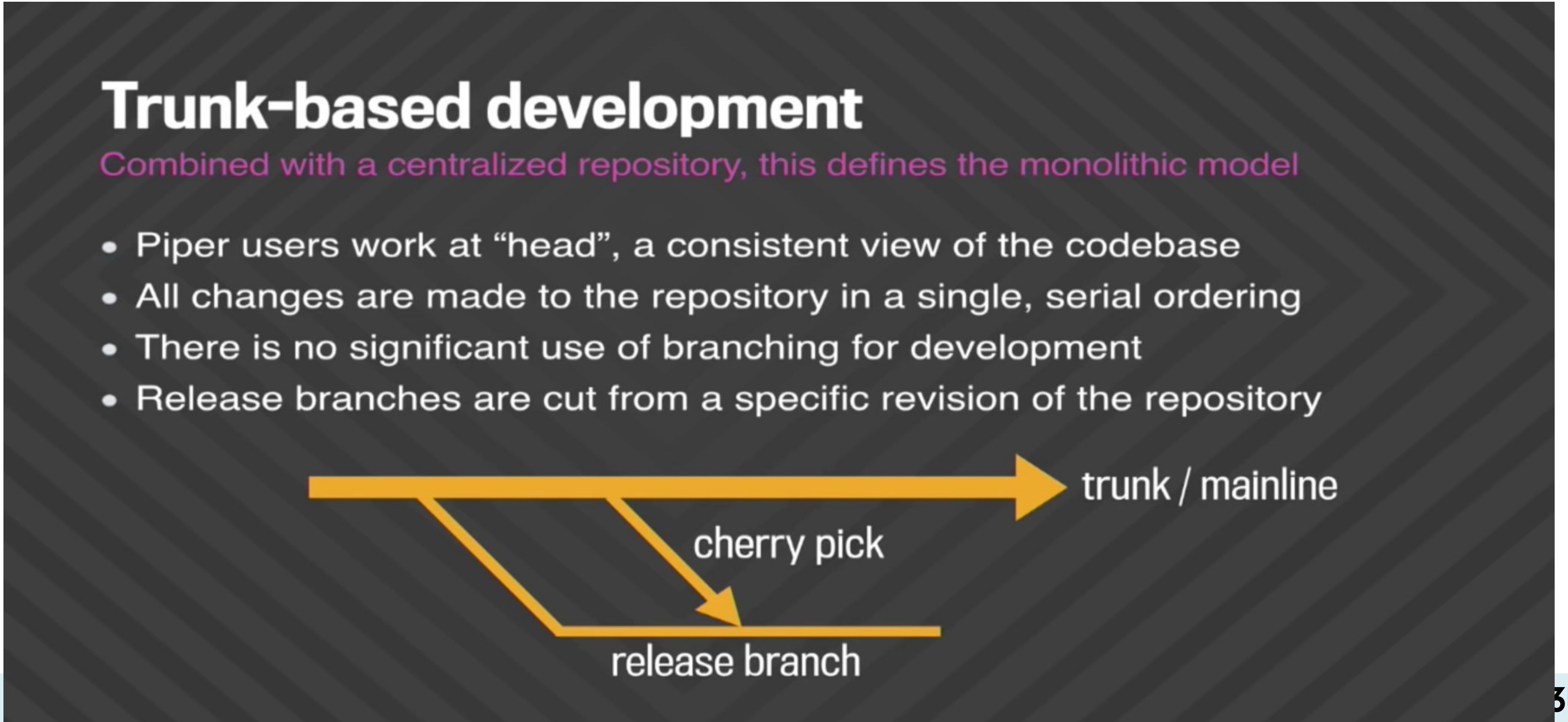
Work locally without complete history.

Need more history?
Downloaded automatically on demand.

Some Common Principles

- Ensure Isolation
  - Of impacts of a given changeset
    - On the build status
    - On production code
  - Not dissimilar to distributed systems!
    - Which makes sense; this is also a distributed system, just made up of people
- Work incrementally
  - Release carefully, monitor heavily
  - Cut costs where possible by building & testing as little as possible

Monolithic repository – no major use of branches for development



Trunk-based development

Combined with a centralized repository, this defines the monolithic model

- Piper users work at “head”, a consistent view of the codebase
- All changes are made to the repository in a single, serial ordering
- There is no significant use of branching for development
- Release branches are cut from a specific revision of the repository

A recent history of code organization

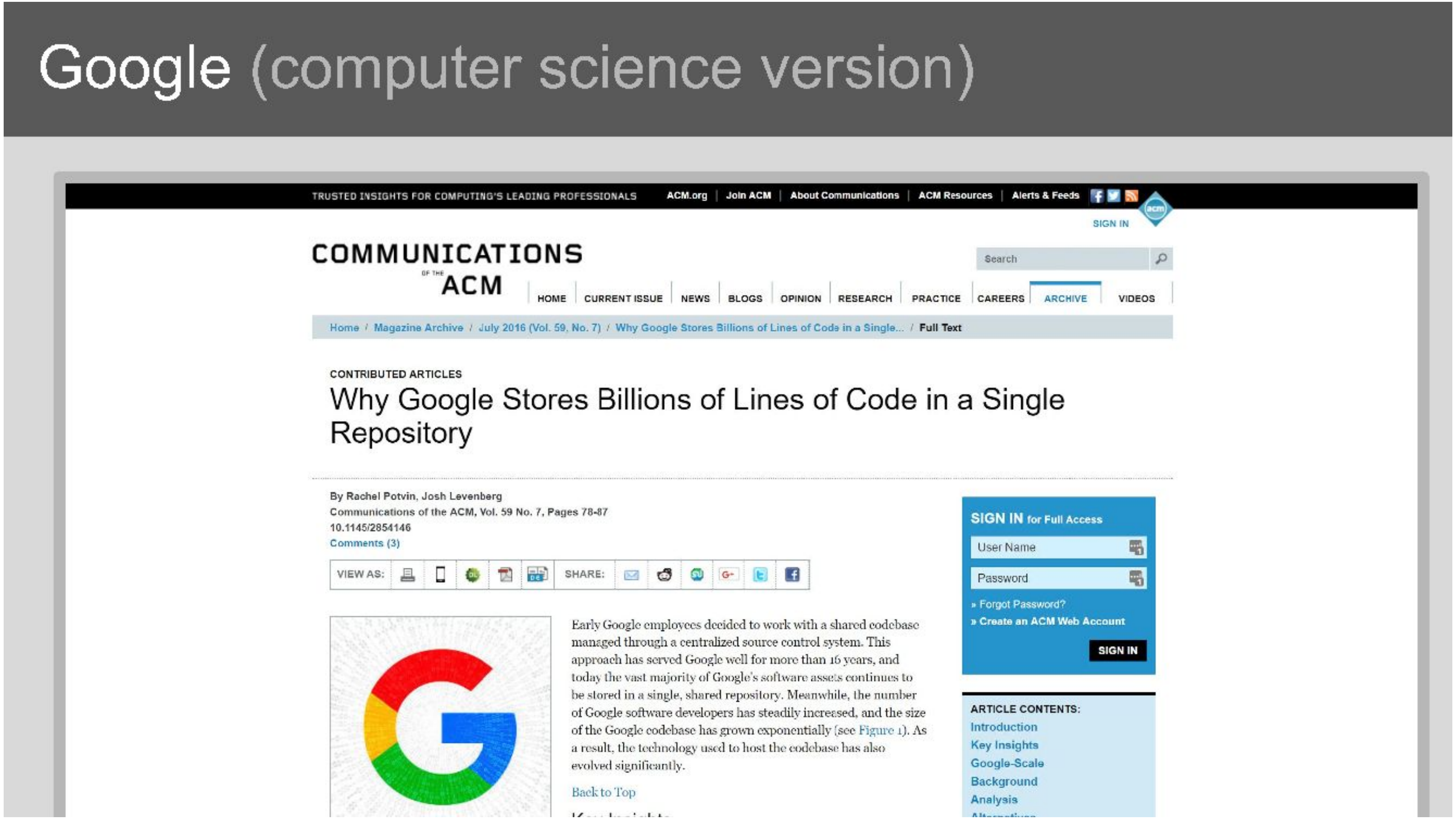
- A single team with a monolithic application in a single repository
- …
- Multiple teams with many separate applications in many separate repositories
- Multiple teams with many ~~separate applications~~ **microservices** in many separate repositories
- A single team with many microservices in many repositories
- …
- Many teams with many applications in one big **Monorepo**

What is a monolithic repository (monorepo)?

- A **single** version control repository containing multiple
  - Projects
  - Applications
  - Libraries
- Often using a common build system

2015 talk by Benjamin Eberlei

Monorepos in industry



Google (computer science version)

TRUSTED INSIGHTS FOR COMPUTING'S LEADING PROFESSIONALS
ACM.org | Join ACM | About Communications | ACM Resources | Alerts & Feeds
SIGN IN
COMMUNICATIONS OF THE ACM
Search
HOME | CURRENT ISSUE | NEWS | BLOGS | OPINION | RESEARCH | PRACTICE | CAREERS | ARCHIVE | VIDEOS

Home / Magazine Archive / July 2016 (Vol. 59, No. 7) / Why Google Stores Billions of Lines of Code in a Single... / Full Text

CONTRIBUTED ARTICLES

Why Google Stores Billions of Lines of Code in a Single Repository

By Rachel Potvin, Josh Levenberg
Communications of the ACM, Vol. 59 No. 7, Pages 78-87
10.1145/2854146
Comments (3)

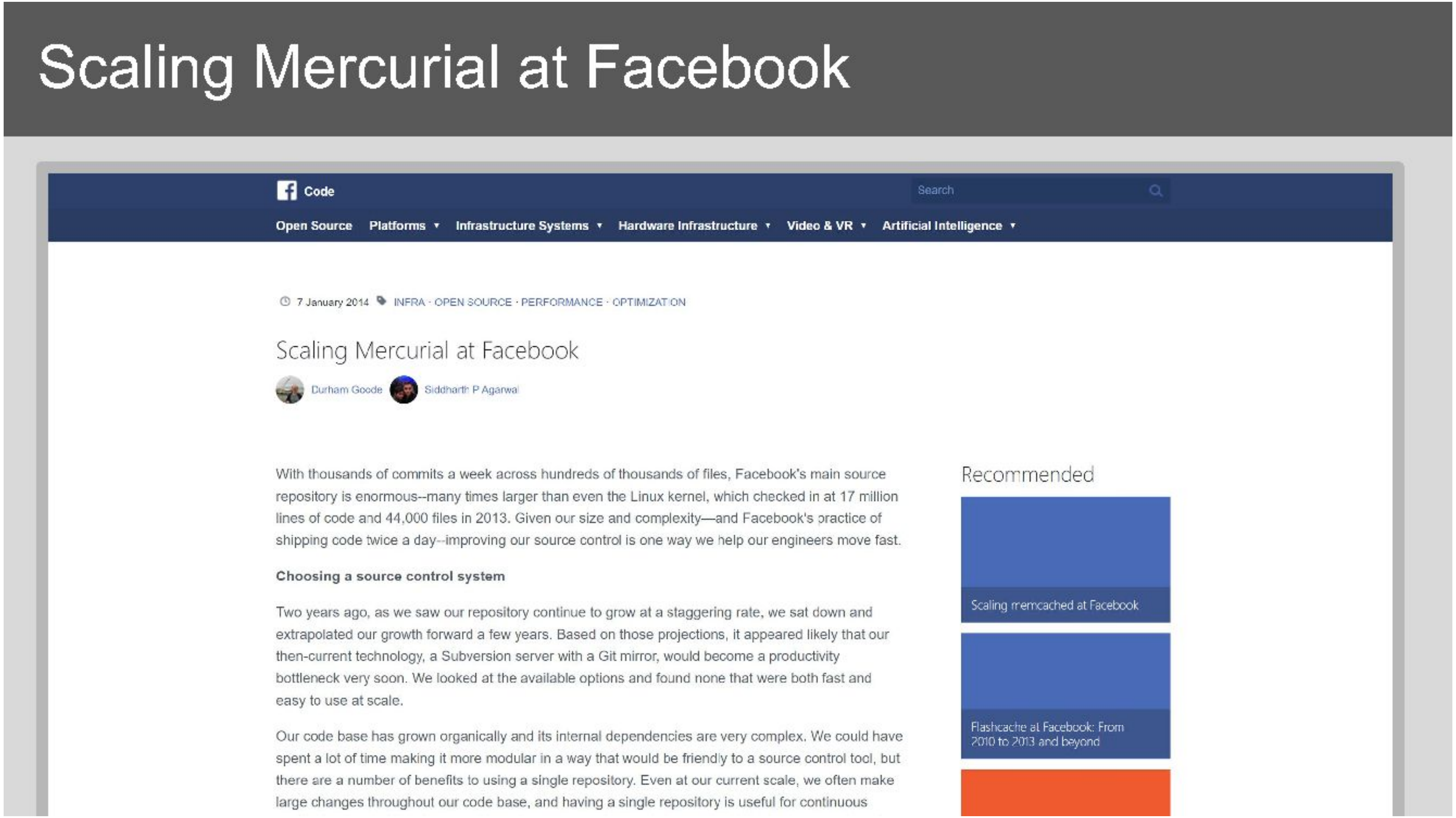
Early Google employees decided to work with a shared codebase managed through a centralized source control system. This approach has served Google well for more than 16 years, and today the vast majority of Google's software assets continues to be stored in a single, shared repository. Meanwhile, the number of Google software developers has steadily increased, and the size of the Google codebase has grown exponentially (see Figure 1). As a result, the technology used to host the codebase has also evolved significantly.

Back to Top

Key Insights

ARTICLE CONTENTS:
Introduction
Key Insights
Google-Scale
Background
Analysis
Alternatives

Monorepos in industry



Scaling Mercurial at Facebook

7 January 2014 INFRA · OPEN SOURCE · PERFORMANCE · OPTIMIZATION

Scaling Mercurial at Facebook

Durham Goode Siddharth P Agarwal

With thousands of commits a week across hundreds of thousands of files, Facebook's main source repository is enormous--many times larger than even the Linux kernel, which checked in at 17 million lines of code and 44,000 files in 2013. Given our size and complexity—and Facebook's practice of shipping code twice a day--improving our source control is one way we help our engineers move fast.

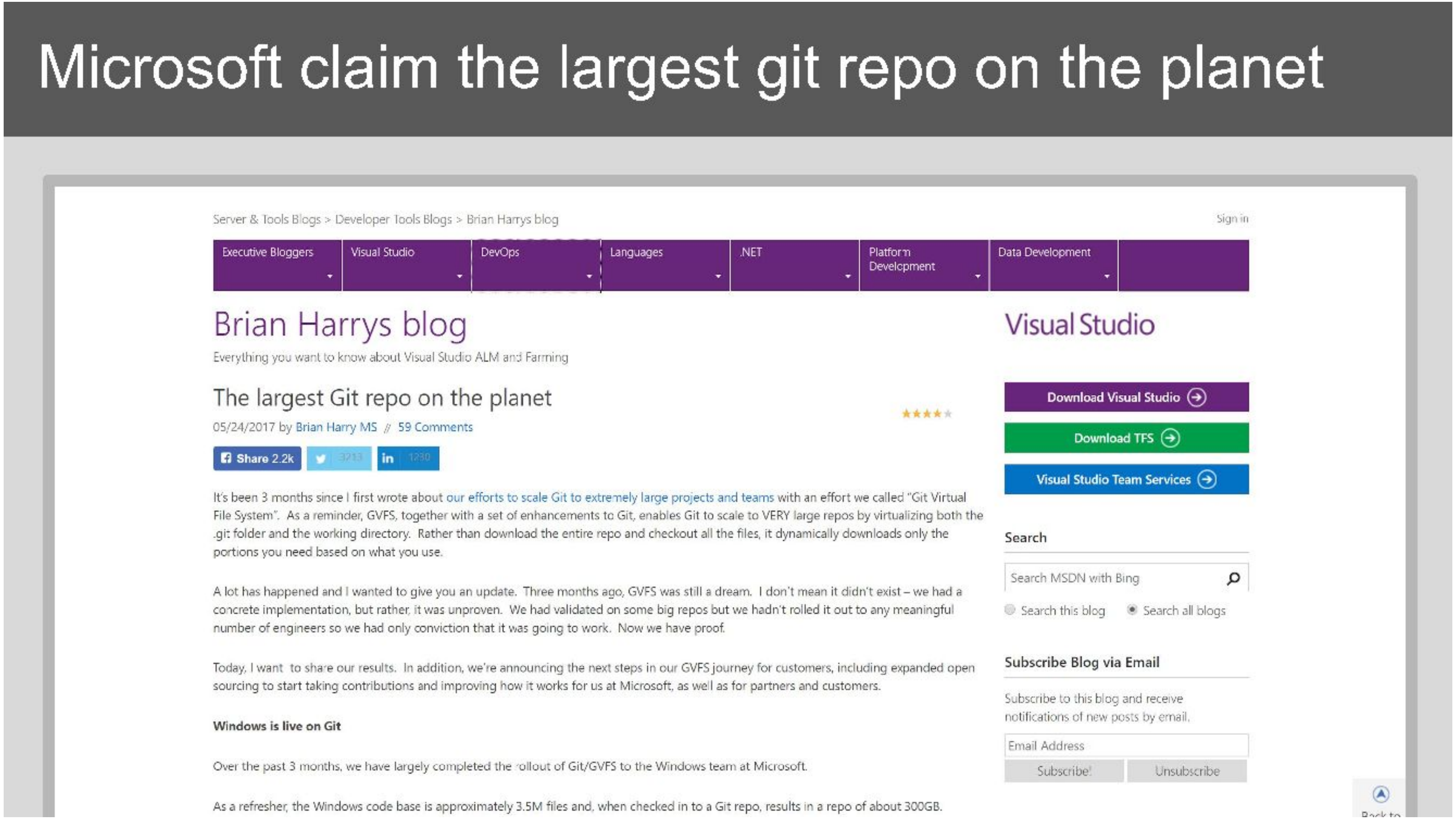
Choosing a source control system

Two years ago, as we saw our repository continue to grow at a staggering rate, we sat down and extrapolated our growth forward a few years. Based on those projections, it appeared likely that our then-current technology, a Subversion server with a Git mirror, would become a productivity bottleneck very soon. We looked at the available options and found none that were both fast and easy to use at scale.

Our code base has grown organically and its internal dependencies are very complex. We could have spent a lot of time making it more modular in a way that would be friendly to a source control tool, but there are a number of benefits to using a single repository. Even at our current scale, we often make large changes throughout our code base, and having a single repository is useful for continuous

Recommended
Scaling memcached at Facebook
Flashcache at Facebook: From 2010 to 2013 and beyond

Monorepos in industry



Microsoft claim the largest git repo on the planet

Brian Harrys blog

Everything you want to know about Visual Studio ALM and Farming

The largest Git repo on the planet

05/24/2017 by Brian Harry MS // 59 Comments

It's been 3 months since I first wrote about our efforts to scale Git to extremely large projects and teams with an effort we called "Git Virtual File System". As a reminder, GVFS, together with a set of enhancements to Git, enables Git to scale to VERY large repos by virtualizing both the .git folder and the working directory. Rather than download the entire repo and checkout all the files, it dynamically downloads only the portions you need based on what you use.

A lot has happened and I wanted to give you an update. Three months ago, GVFS was still a dream. I don't mean it didn't exist – we had a concrete implementation, but rather, it was unproven. We had validated on some big repos but we hadn't rolled it out to any meaningful number of engineers so we had only conviction that it was going to work. Now we have proof.

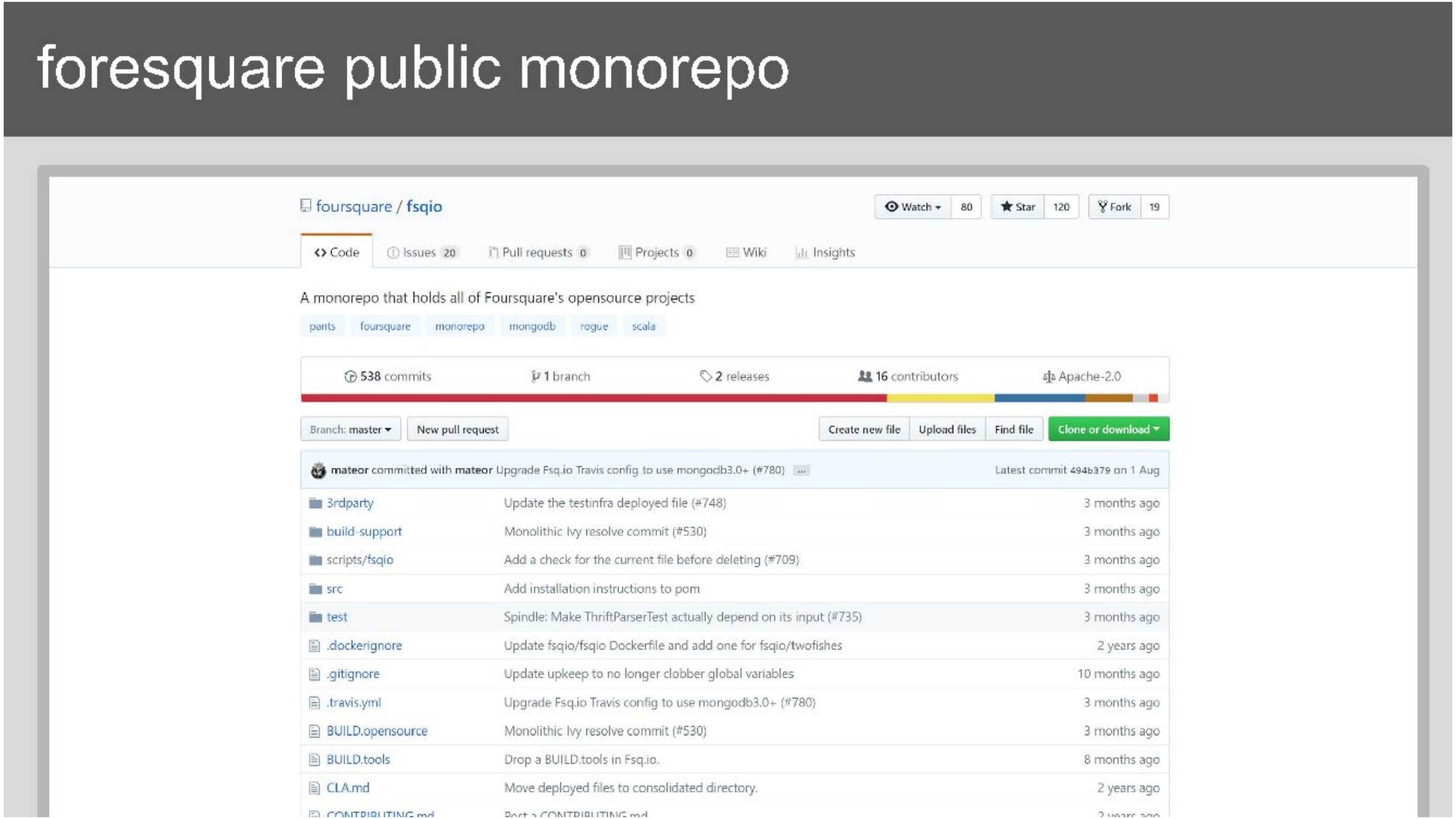
Today, I want to share our results. In addition, we're announcing the next steps in our GVFS journey for customers, including expanded open sourcing to start taking contributions and improving how it works for us at Microsoft, as well as for partners and customers.

Windows is live on Git

Over the past 3 months, we have largely completed the rollout of Git/GVFS to the Windows team at Microsoft.

As a refresher, the Windows code base is approximately 3.5M files and, when checked in to a Git repo, results in a repo of about 300GB.

Monorepos in open-source



foresquare public monorepo

foursquare / fsqio

A monorepo that holds all of Foursquare's opensource projects

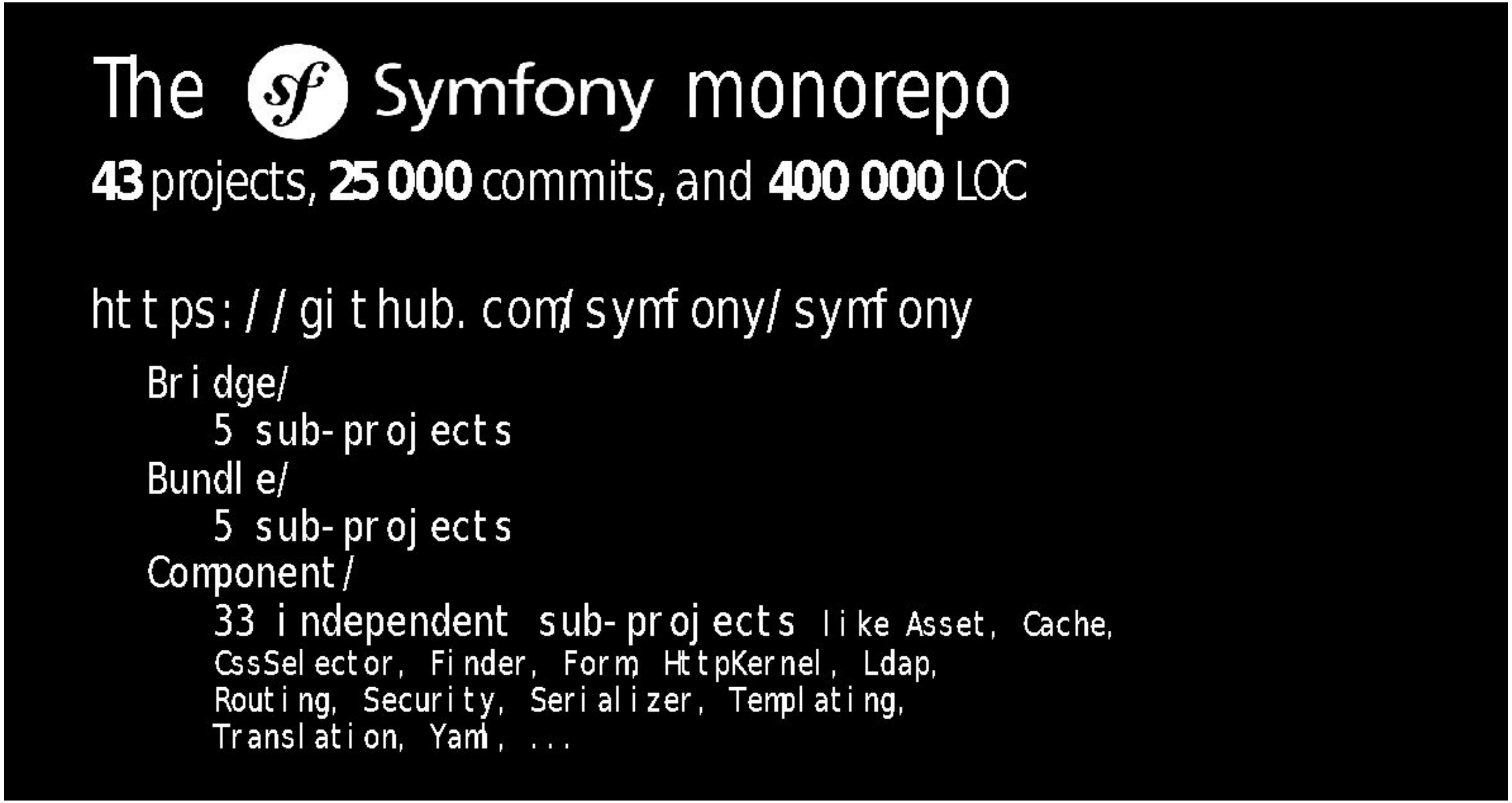
538 commits | 1 branch | 2 releases | 16 contributors | Apache-2.0

mateor committed with mateor Upgrade Fsq.io Travis config to use mongodb3.0+ (#780) ... Latest commit 494b379 on 1 Aug

3rdparty	Update the testinfra deployed file (#748)	3 months ago
build-support	Monolithic Ivy resolve commit (#530)	3 months ago
scripts/fsqio	Add a check for the current file before deleting (#709)	3 months ago
src	Add installation instructions to pom	3 months ago
test	Spindle: Make ThriftParserTest actually depend on its input (#735)	3 months ago
.dockerignore	Update fsqio/fsqio Dockerfile and add one for fsqio/twofishes	2 years ago
.gitignore	Update upkeep to no longer clobber global variables	10 months ago
.travis.yml	Upgrade Fsq.io Travis config to use mongodb3.0+ (#780)	3 months ago
BUILD.opensource	Monolithic Ivy resolve commit (#530)	3 months ago
BUILD.tools	Drop a BUILD.tools in Fsq.io.	8 months ago
CLA.md	Move deployed files to consolidated directory.	2 years ago
CONTRIBUTING.md	Post a CONTRIBUTING.md	2 years ago

2016 talk by FABIEN POTENCIER

Monorepos in open-source



The Symfony monorepo

43 projects, **25 000** commits, and **400 000** LOC

https://github.com/symfony/symfony

```
Bridge/
    5 sub-projects
Bundle/
    5 sub-projects
Component/
    33 independent sub-projects like Asset, Cache,
    CssSelector, Finder, Form, HttpKernel, Ldap,
    Routing, Security, Serializer, Templating,
    Translation, Yaml, ...
```

2016 talk by FABIEN POTENCIER

Advantages of Monorepos

- High discoverability
  - Developers can read & search the entire codebase
- High reuse
  - The same tools (e.g., linters, auto-complete) are globally available
  - Any package can become a library
    - Which is why you always build an API!
- Simplifies maintenance
  - Global refactorings, cleanup
    - Orgs like Google will regularly dedicate a specific day to a type of improvement (e.g., improve documentation), flag all potentially problematic sites

Some more advantages

- Easy continuous integration and code review for changes spanning several projects
- (Internal) dependency management is a non-issue
- Less context switching for developers
- Code more reusable in other contexts
- Access control is easy

Releasing at scale in industry

- Facebook: https://atscaleconference.com/videos/rapid-release-at-massive-scale/
- Google: https://www.slideshare.net/JohnMicco1/2016-0425-continuous-integration-at-google-scal
  https://testing.googleblog.com/2011/06/testing-at-speed-and-scale-of-google.html
- Why Google Stores Billions of Lines of Code in a Single Repository: https://www.youtube.com/watch?v=W71BTkUbdqE
- F8 2015 - Big Code: Developer Infrastructure at Facebook's Scale: https://www.youtube.com/watch?v=X0VH78ye4yY