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

Part 4: et cetera

Toward SE in practice: DevOps and branch management

Josh Bloch **Charlie Garrod**

## Administrivia

- Homework 6 available
  - Checkpoint deadline tonight
  - Due next Wednesday, April 29th

## Key concepts from last Thursday

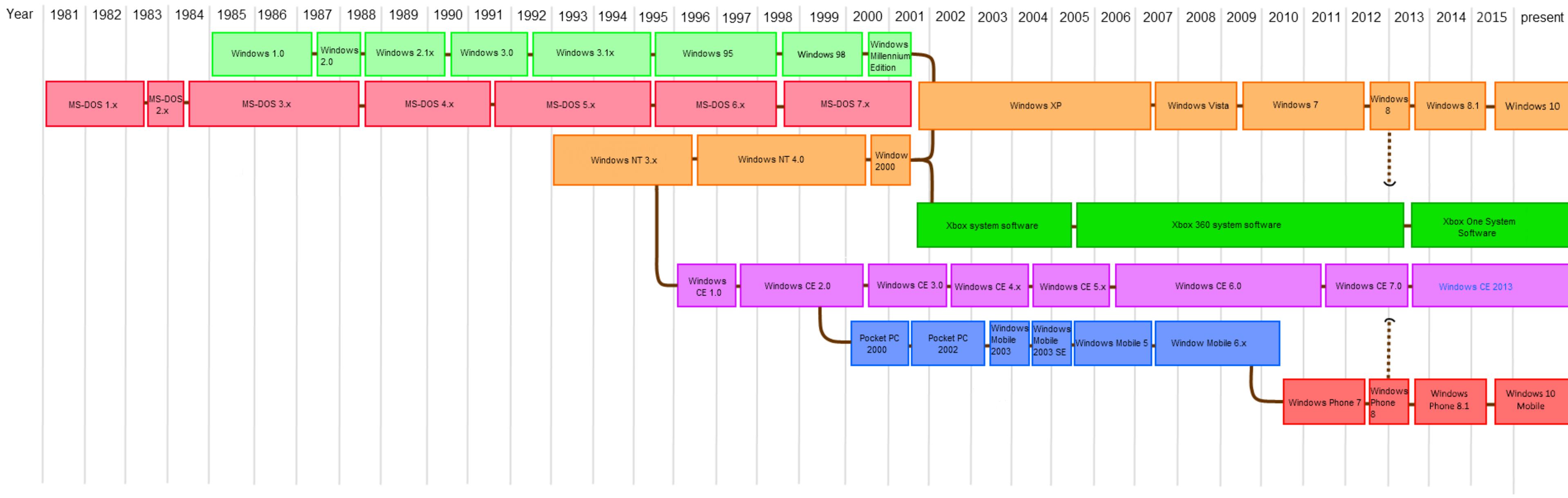
- SE empirical methods:  Test-driven development case study
- Version and release management

## Today:  Software engineering in practice

- Release management, introduction to DevOps
- Choose your own adventure…
- Monolithic repositories

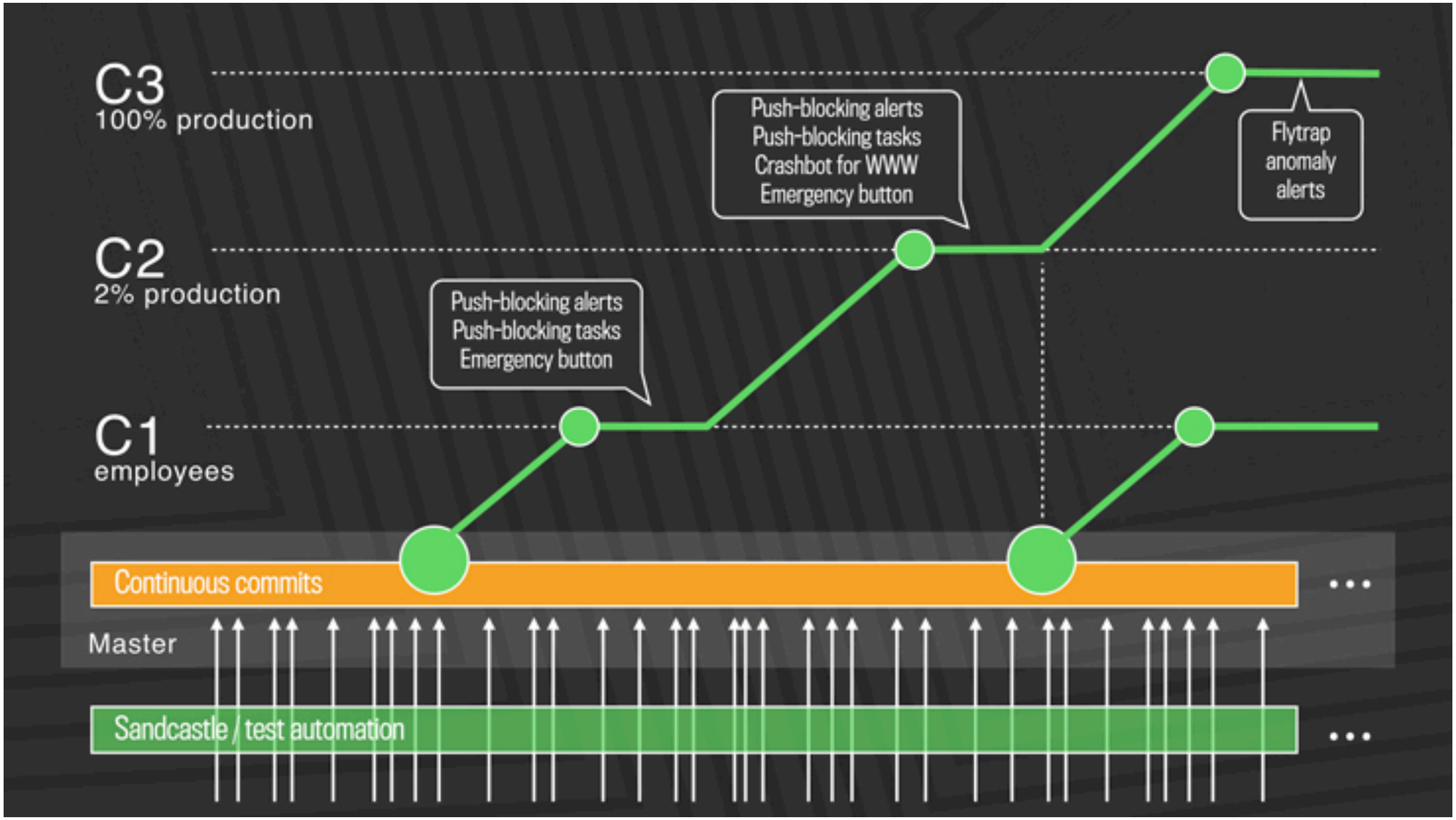
# Consider: timelines of traditional software development

e.g., the Microsoft* OS development history

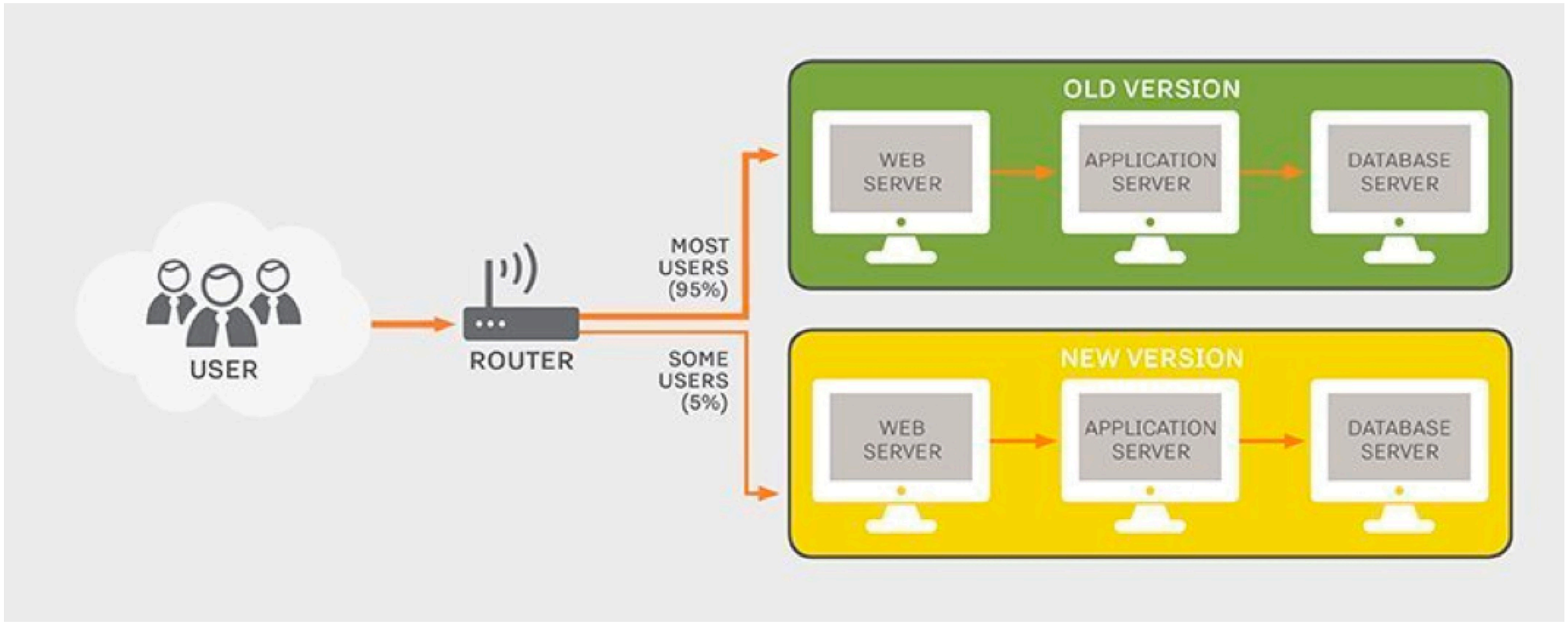


Source: By Paulire - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=46634740

# Modern Facebook release cycle (1000+ diffs / day)

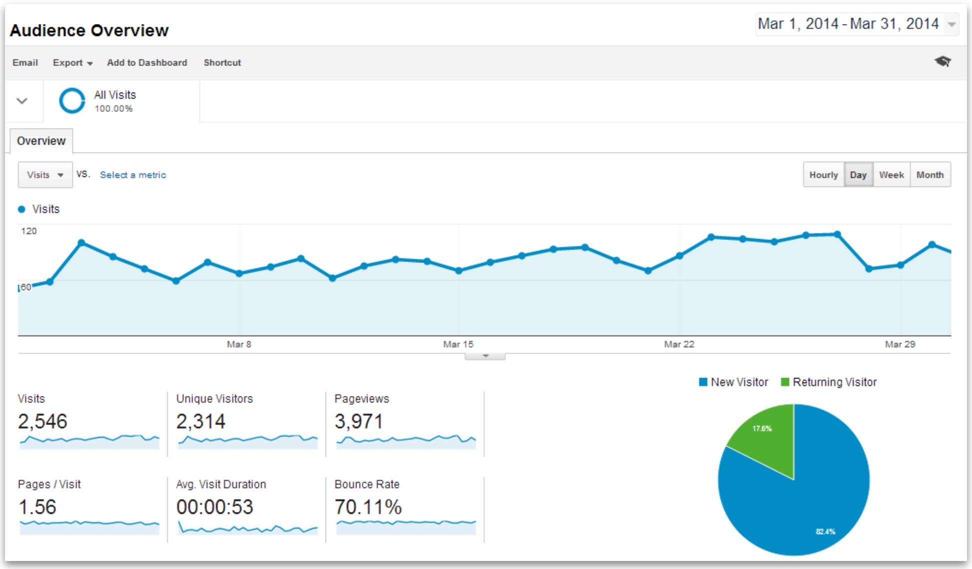


## Aside: Canary testing



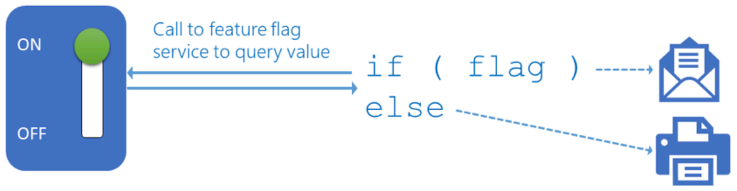
# Aside:  Dark launches and A/B testing

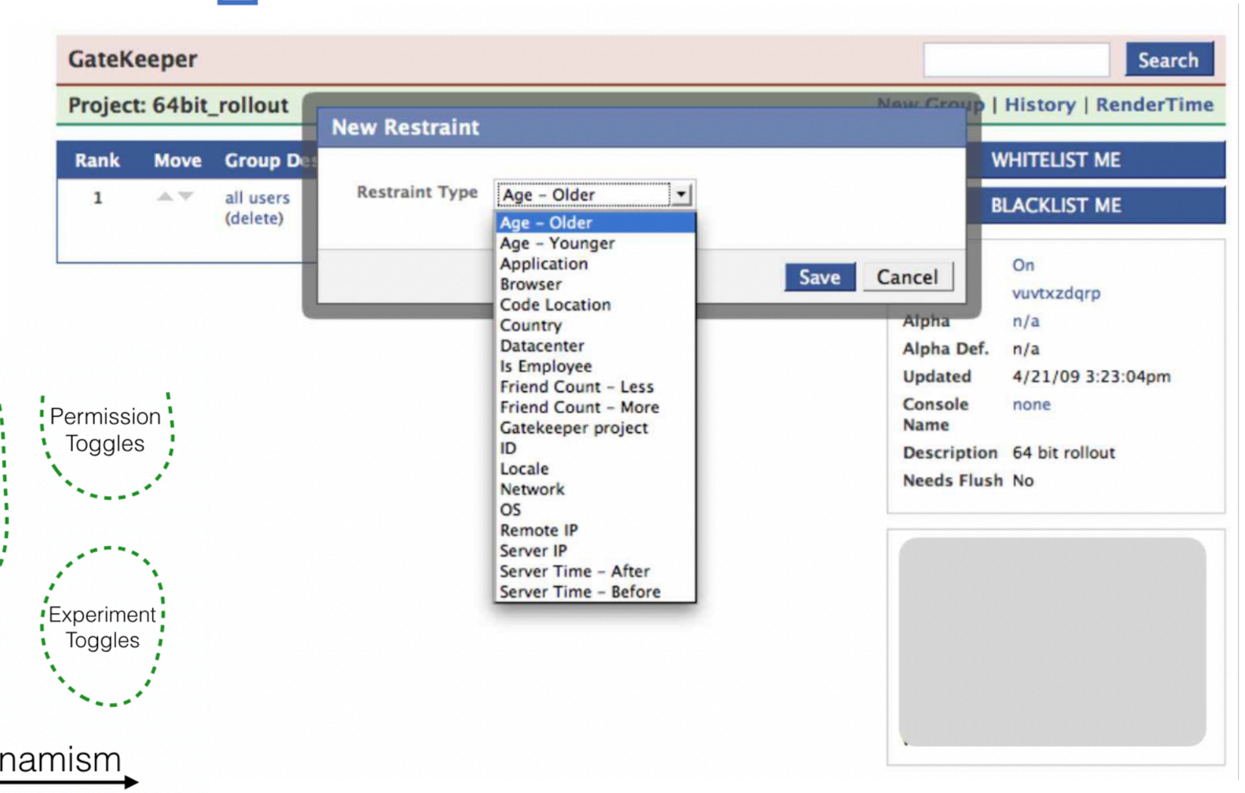
- Focuses on user response to frontend changes rather than performance of backend
- Measure user response via *metrics: engagement, adoption*



Visits 2,546
Unique Visitors 2,314
Pageviews 3,971
Pages / Visit 1.56
Avg. Visit Duration 00:00:53
Bounce Rate 70.11%

# Version management using feature flags



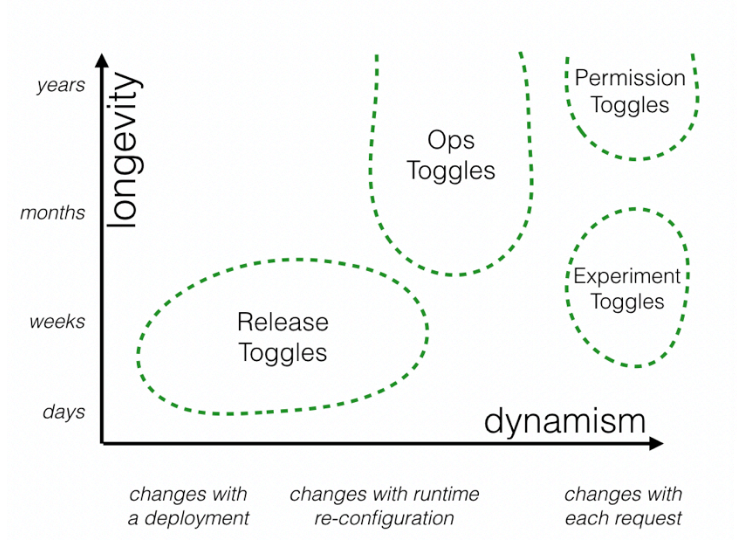


Project: 64bit_rollout

| Rank | Move | Group Des |
|---|---|---|
| 1 | | all users (delete) |

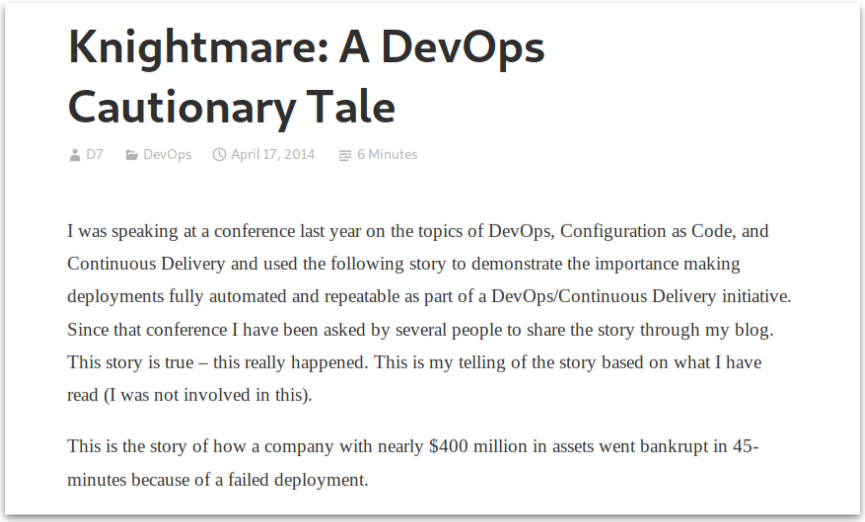
Restraint Type: Age – Older

| | |
|---|---|
| | On |
| | vuvtxzdqrp |
| Alpha | n/a |
| Alpha Def. | n/a |
| Updated | 4/21/09 3:23:04pm |
| Console Name | none |
| Description | 64 bit rollout |
| Needs Flush | No |



https://martinfowler.com/articles/feature-toggles.html
https://docs.microsoft.com/en-us/azure/devops/migrate/phase-features-with-feature-flags?view=azure-devops

## **Warning!** Feature flags can be dangerous



Knightmare: A DevOps Cautionary Tale

D7 DevOps April 17, 2014 6 Minutes

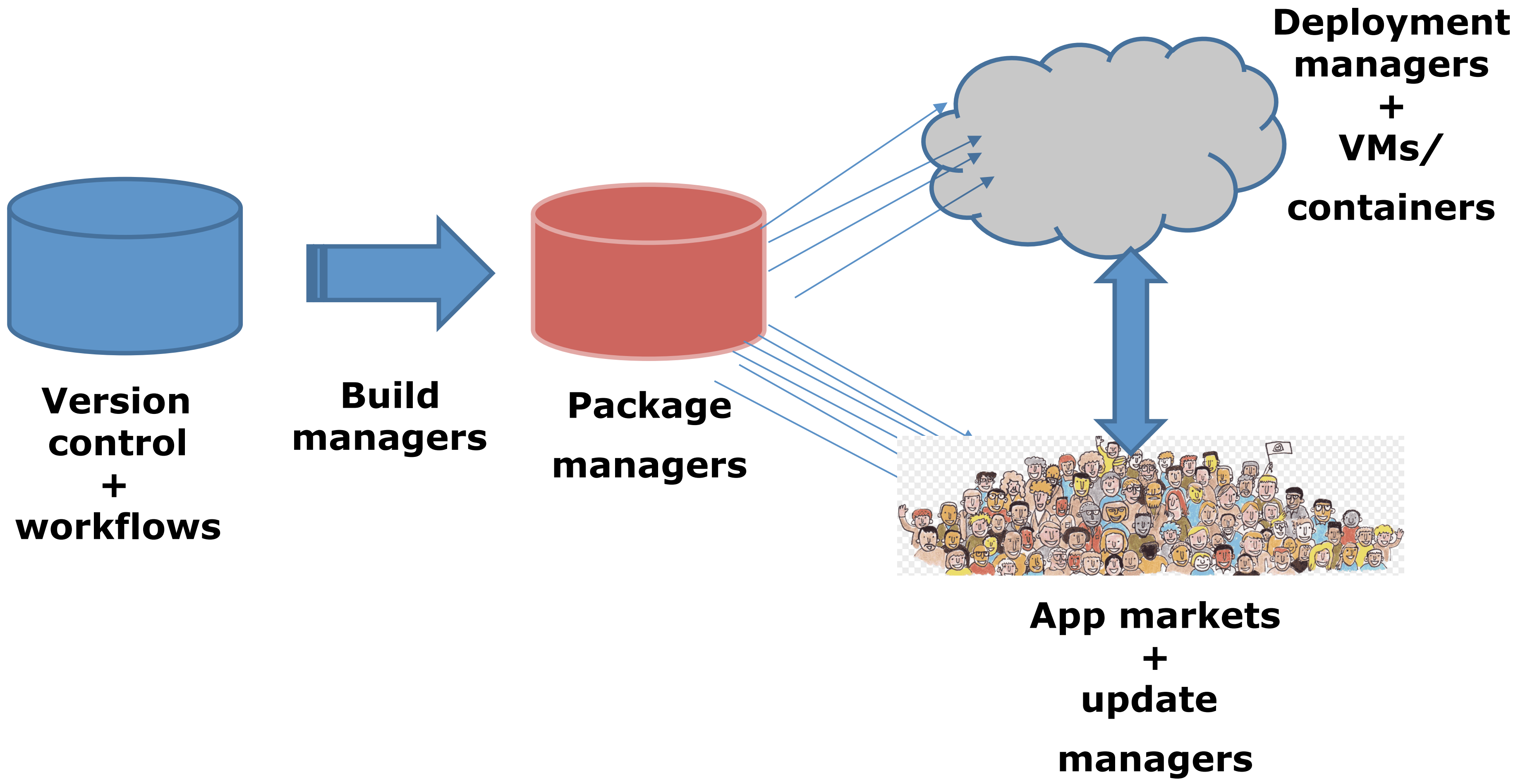
I was speaking at a conference last year on the topics of DevOps, Configuration as Code, and Continuous Delivery and used the following story to demonstrate the importance making deployments fully automated and repeatable as part of a DevOps/Continuous Delivery initiative. Since that conference I have 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.

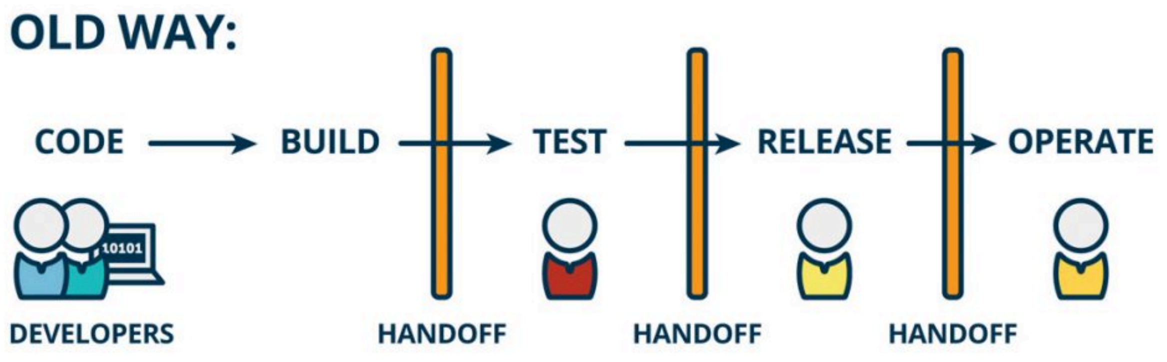
**Knight Capital Group realized a $460 million loss in 45-minutes, going from being the largest trader in US equities to bankruptcy.**

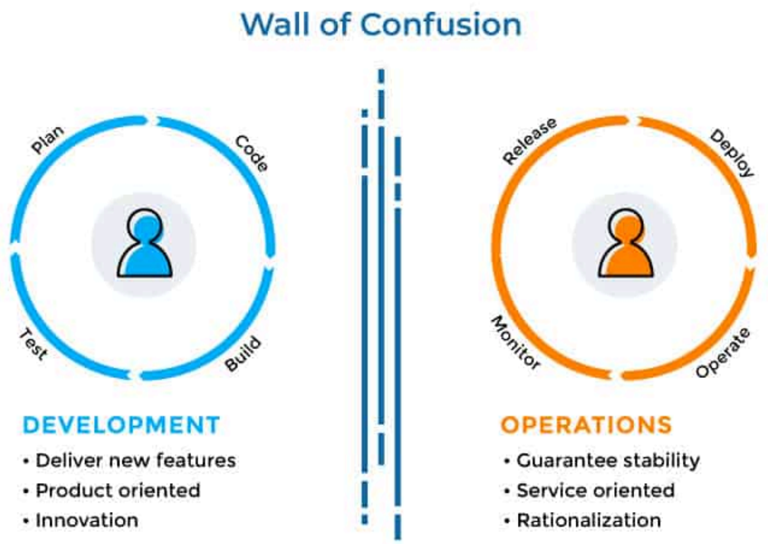
https://dougseven.com/2014/04/17/knightmare-a-devops-cautionary-tale/

# Configuration management in the modern world



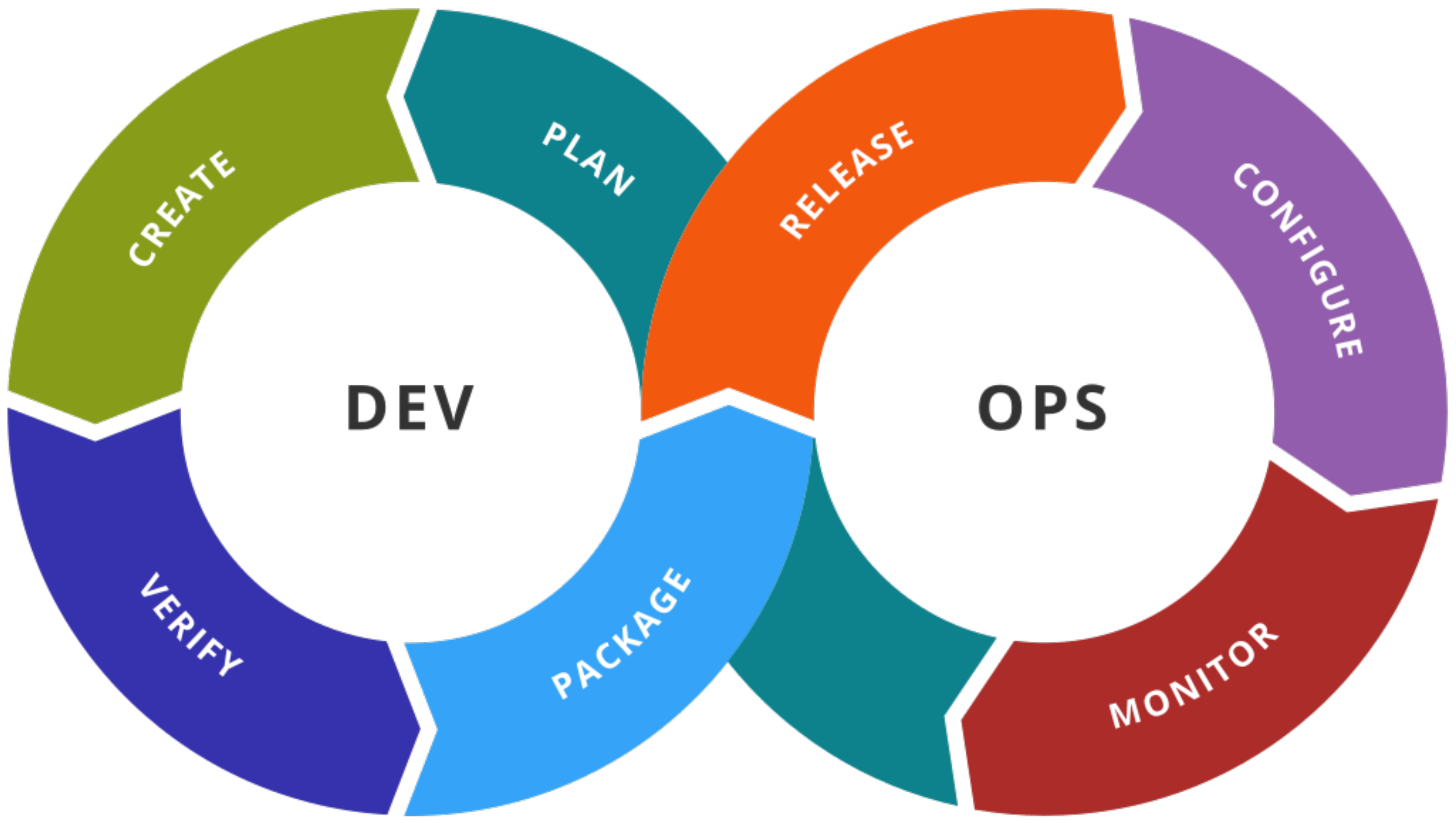
## Devs, Ops, and The Wall of Confusion



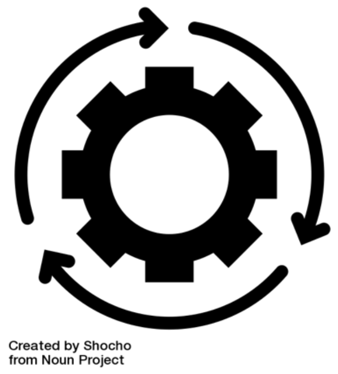


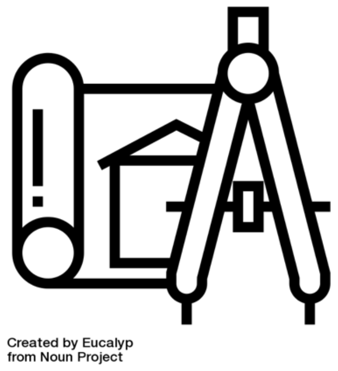
https://www.plutora.com/blog/what-is-enterprise-devops
https://www.yudiz.com/welcome-devops-prevent-defects/

# DevOps:  Development / Operations



# Two sides to DevOps





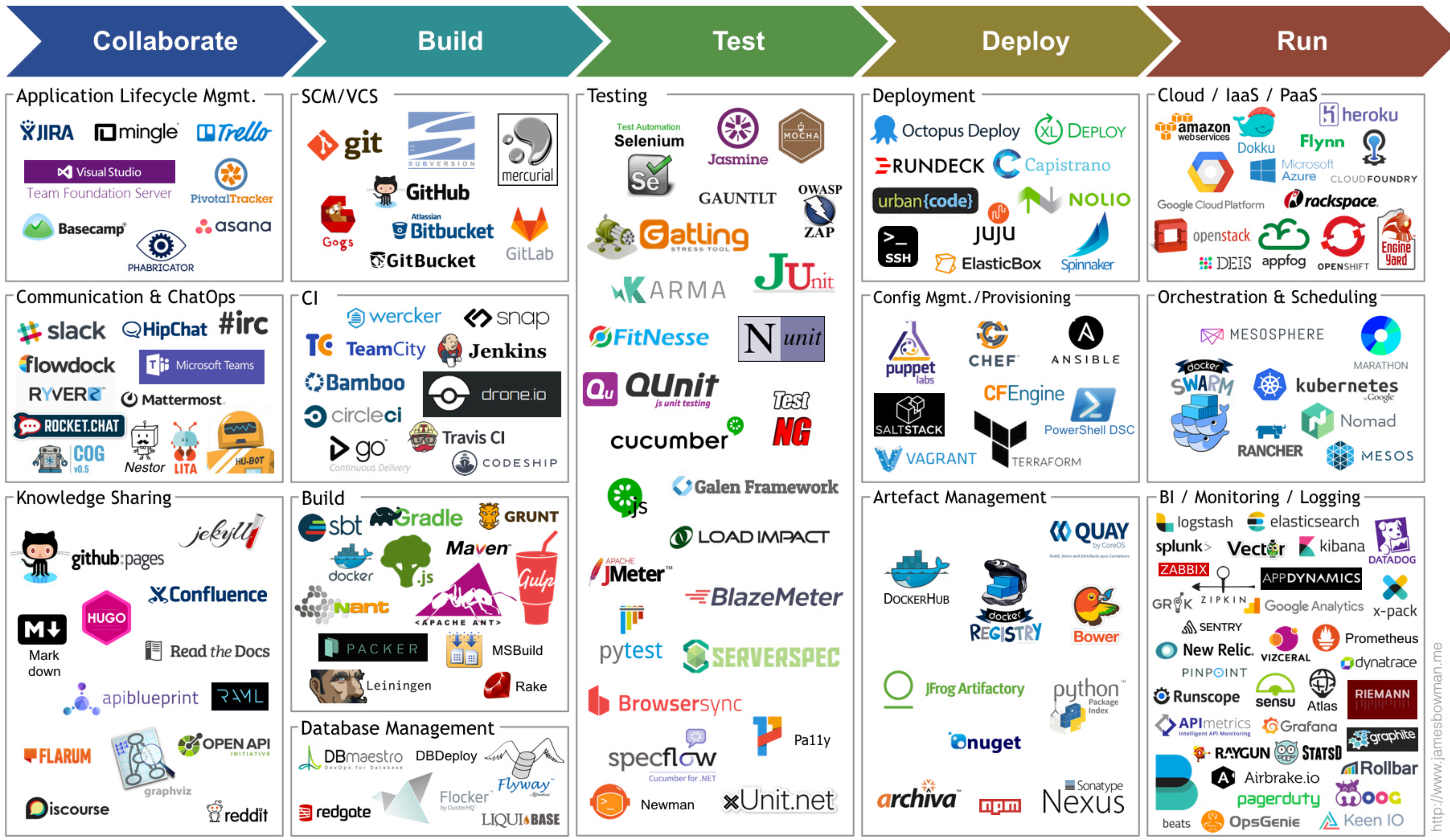
**Operations-oriented**

- Manage servers automatically
- Easier to identify and fix bugs
- Automatic logging, monitoring, and operations

**Developer-oriented**

- Agile releases!
- Easier to share and understand code
- Faster onboarding
- Safely push code through CI/CD pipeline

# DevOps ecosystems…



## Principle: Shared responsibility

- Breakdown the wall of confusion
- Improve collaboration between dev. and ops. teams
- Reduce "throw it over the fence" syndrome
- Treat failures as a learning experience...

## Principle: Rapid releases and feedback

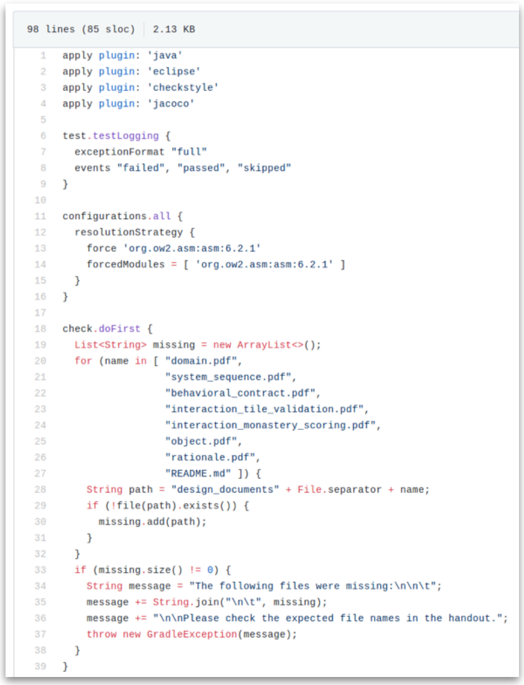
- Remove the manual and ceremonial aspects from releases
  - Possibly continuous releases
  - Incremental rollout; quick rollback
- Get feedback on your changes ASAP
  - Continuously measure quality, refine implementation, and rerelease

Travis CI

PHABRICATOR

# Principle: Configuration as code

- Manage deployment config files in your version control system
  - Travis, Gradle, Jenkins, …
- Packaging and installation
  - Docker, package.json, setup.py, pom.xml, ...
- Infrastructure and deployment
  - Docker Compose, Ansible, Puppet, Kubernetes
  - Manage servers and resources
- ...



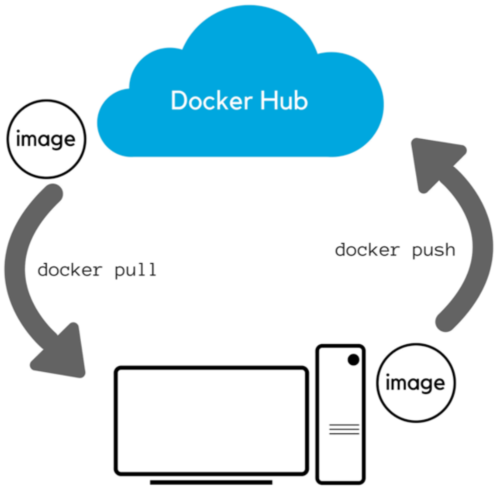
98 lines (85 sloc) | 2.13 KB

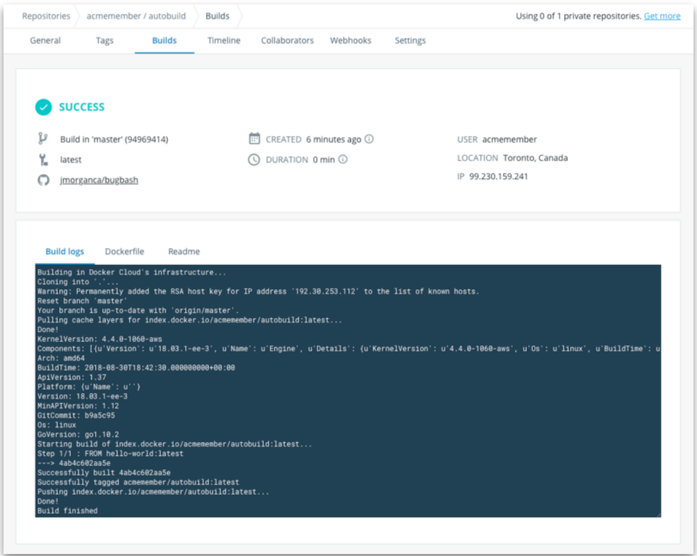
```
apply plugin: 'java'
apply plugin: 'eclipse'
apply plugin: 'checkstyle'
apply plugin: 'jacoco'

test.testLogging {
  exceptionFormat "full"
  events "failed", "passed", "skipped"
}

configurations.all {
  resolutionStrategy {
    force 'org.ow2.asm:asm:6.2.1'
    forcedModules = [ 'org.ow2.asm:asm:6.2.1' ]
  }
}

check.doFirst {
  List<String> missing = new ArrayList<>();
  for (name in [ "domain.pdf",
                 "system_sequence.pdf",
                 "behavioral_contract.pdf",
                 "interaction_tile_validation.pdf",
                 "interaction_monastery_scoring.pdf",
                 "object.pdf",
                 "rationale.pdf",
                 "README.md" ]) {
    String path = "design_documents" + File.separator + name;
    if (!file(path).exists()) {
      missing.add(path);
    }
  }
  if (missing.size() != 0) {
    String message = "The following files were missing:\n\n\t";
    message += String.join("\n\t", missing);
    message += "\n\nPlease check the expected file names in the handout.";
    throw new GradleException(message);
  }
}
```

# Aside: Docker and DockerHub

- Build an image for each release
- Quickly rollback to stable versions

```
$ docker pull mysql:8.0
$ docker push christimperley/darjeeling
```





SUCCESS

Build in 'master' (94969414)
latest
jmorganca/bugbash

CREATED 6 minutes ago
DURATION 0 min

USER acmemember
LOCATION Toronto, Canada
IP 99.230.159.241

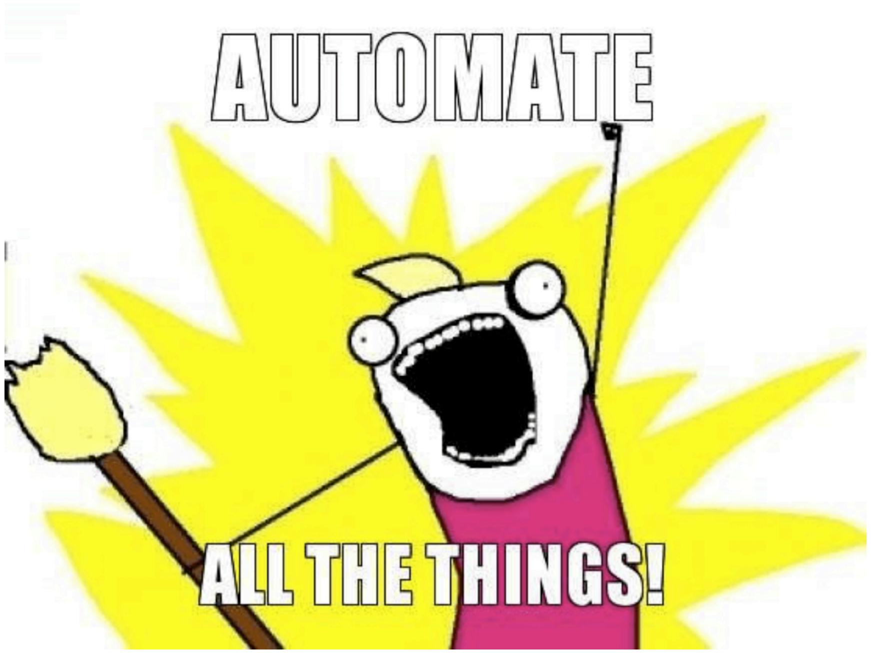
```
Building in Docker Cloud's infrastructure...
Cloning into '.'...
Warning: Permanently added the RSA host key for IP address '192.30.253.112' to the list of known hosts.
Reset branch 'master'
Your branch is up-to-date with 'origin/master'.
Pulling cache layers for index.docker.io/acmemember/autobuild:latest...
Done!
KernelVersion: 4.4.0-1060-aws
Components: [{u'Version': u'18.03.1-ee-3', u'Name': u'Engine', u'Details': {u'KernelVersion': u'4.4.0-1060-aws', u'Os': u'linux', u'BuildTime': u
Arch: amd64
BuildTime: 2018-08-30T18:42:30.000000000+00:00
ApiVersion: 1.37
Platform: {u'Name': u''}
Version: 18.03.1-ee-3
MinAPIVersion: 1.12
GitCommit: b9a5c95
Os: linux
GoVersion: go1.10.2
Starting build of index.docker.io/acmemember/autobuild:latest...
Step 1/1 : FROM hello-world:latest
---> 4ab4c602aa5e
Successfully built 4ab4c602aa5e
Successfully tagged acmemember/autobuild:latest
Pushing index.docker.io/acmemember/autobuild:latest...
Done!
Build finished
```

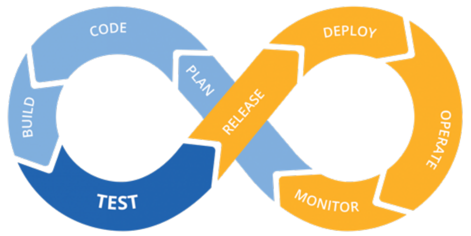
https://docs.docker.com/docker-hub/builds/
https://static.packt-cdn.com/products/9781789137231/graphics/99abf1ea-4efe-4ccd-93c3-b36e80f3263c.png

# Principle: Automation everywhere

INSTALL.SH

```
#!/bin/bash

pip install "$1" &
easy_install "$1" &
brew install "$1" &
npm install "$1" &
yum install "$1" & dnf install "$1" &
docker run "$1" &
pkg install "$1" &
apt-get install "$1" &
sudo apt-get install "$1" &
steamcmd +app_update "$1" validate &
git clone https://github.com/"$1"/"$1" &
cd "$1";./configure;make;make install &
curl "$1" | bash &
```





https://blog.chef.io/automate-all-the-things/

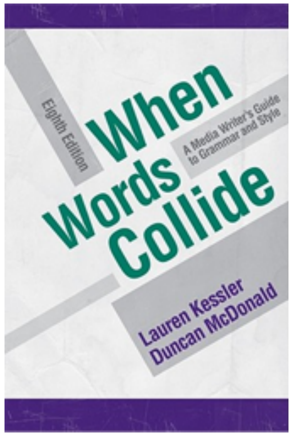
## DevOps Summary

- DevOps brings development and operations together
  - Automation, Automation, Automation
  - Infrastructure as code
- Continuous deployment is increasingly common
- Exploit opportunities of continuous deployment; perform testing in production and quickly rollback
  - Experiment, measure, and improve

## Today: Software engineering in practice

- Introduction to DevOps
- Choose your own adventure…
  - Repository branch management
  - A Java Puzzler
- Monolithic repositories

## 6. “When Words Collide”



```
public class PrintWords {
  public static void main(String[] args) {
    System.out.println(
      Words.FIRST + " " + Words.SECOND + " " + Words.THIRD);
  }
}

public class Words { // Compile PrintWords against this version
  public static final String FIRST  = "the";
  public static final String SECOND = null;
  public static final String THIRD  = "set";
}

public class Words { // Run against this version
  public static final String FIRST  = "physics";
  public static final String SECOND = "chemistry";
  public static final String THIRD  = "biology";
}
```

## What does it print?

**(a)** `the null set`
**(b)** `physics chemistry biology`
**(c)** Throws exception
**(d) None of the above**

```
public class PrintWords {
  public static void main(String[] args) {
    System.out.println(
      Words.FIRST + " " + Words.SECOND + " " + Words.THIRD);
  }
}

public class Words { // Compile PrintWords against this version
  public static final String FIRST  = "the";
  public static final String SECOND = null;
  public static final String THIRD  = "set";
}

public class Words { // Run against this version
  public static final String FIRST  = "physics";
  public static final String SECOND = "chemistry";
  public static final String THIRD  = "biology";
}
```

## What does it print?

(a) `the null set`

(b) `physics chemistry biology`

(c) Throws exception

(d) None of the above: `the chemistry set`

Java inlines *constant variables*

## What exactly is a constant variable?

- Loosely speaking, a final primitive or `String` variable whose value is a *compile-time constant*
  - See JLS3 4.12.4, 13.4.9, 15.28 for gory details
- Surprisingly, `null` *isn't* a compile-time constant

## Another look

```
public class PrintWords {
  public static void main(String[] args) {
    System.out.println(
      Words.FIRST + " " + Words.SECOND + " " + Words.THIRD);
  }
}

public class Words { // Compile PrintWords against this version
  public static final String FIRST  = "the";    // Constant variable
  public static final String SECOND = null; "; // Not a constant variable!!!
  public static final String THIRD  = "set";    // Constant variable
}

public class Words { // Run against this version
  public static final String FIRST  = "physics";
  public static final String SECOND = "chemistry";
  public static final String THIRD  = "biology";
}
```

## How do you prevent constants from being inlined?

```
// Utility function that simply returns its argument
private static String ident(String s) {
  return s;
}

// None of these fields are constant variables!
public class Words {
  public static final String FIRST  = ident("the");
  public static final String SECOND = ident(null);
  public static final String THIRD  = ident("set");
}
```

**Prints physics chemistry biology**

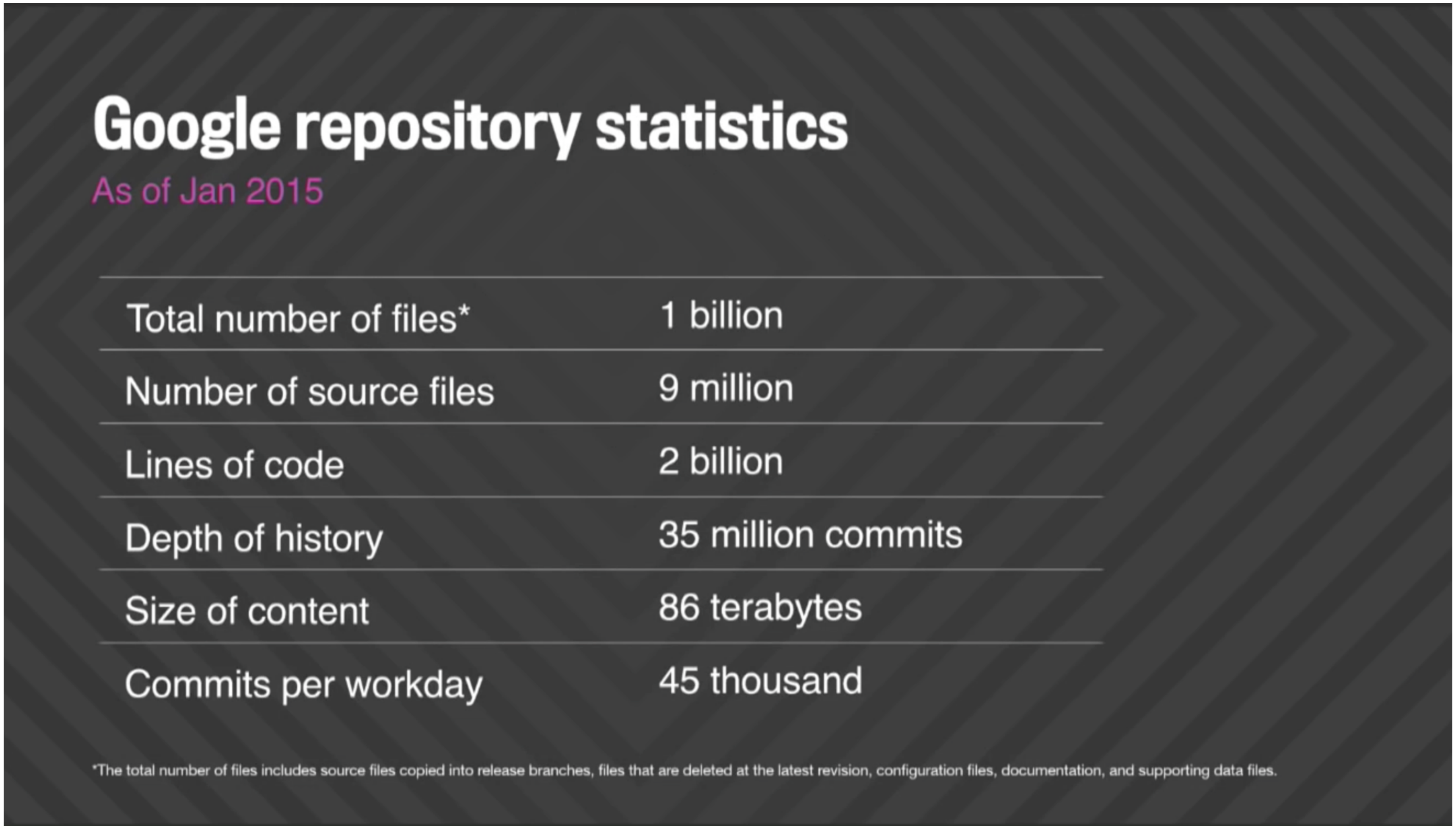
## The Moral

- Constant variable references are inlined
  - Only primitives and strings can be constant variables
  - **`null`** is not a constant variable (neither are enums)
- **If you change a constant's value without recompiling its clients, they break!**
  - Use constant variable only if value will *never* change
  - Use **`ident`** method for final primitive or string fields whose value may change
- For language designers
  - Don't inline constants in a late-binding language
  - More generally, be consistent!

## Today: Software engineering in practice

- Introduction to DevOps
- Choose your own adventure...
  - Repository branch management
  - A Java Puzzler
- Monolithic repositories

## Google: continuous deployment, huge code base



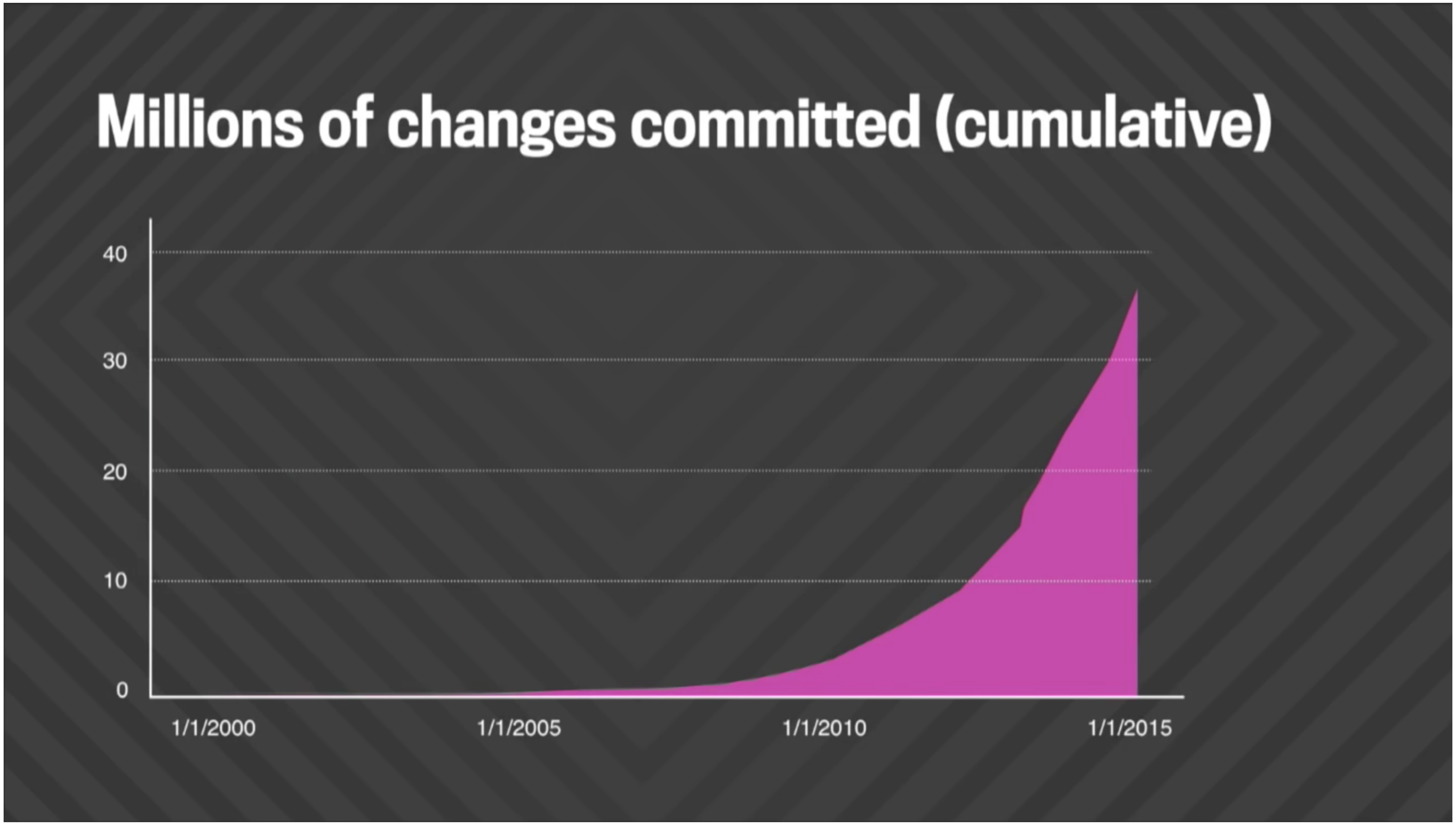
Google repository statistics

As of Jan 2015

| | |
|---|---|
| Total number of files* | 1 billion |
| Number of source files | 9 million |
| Lines of code | 2 billion |
| Depth of history | 35 million commits |
| Size of content | 86 terabytes |
| Commits per workday | 45 thousand |

*The total number of files includes source files copied into release branches, files that are deleted at the latest revision, configuration files, documentation, and supporting data files.

## Exponential growth?



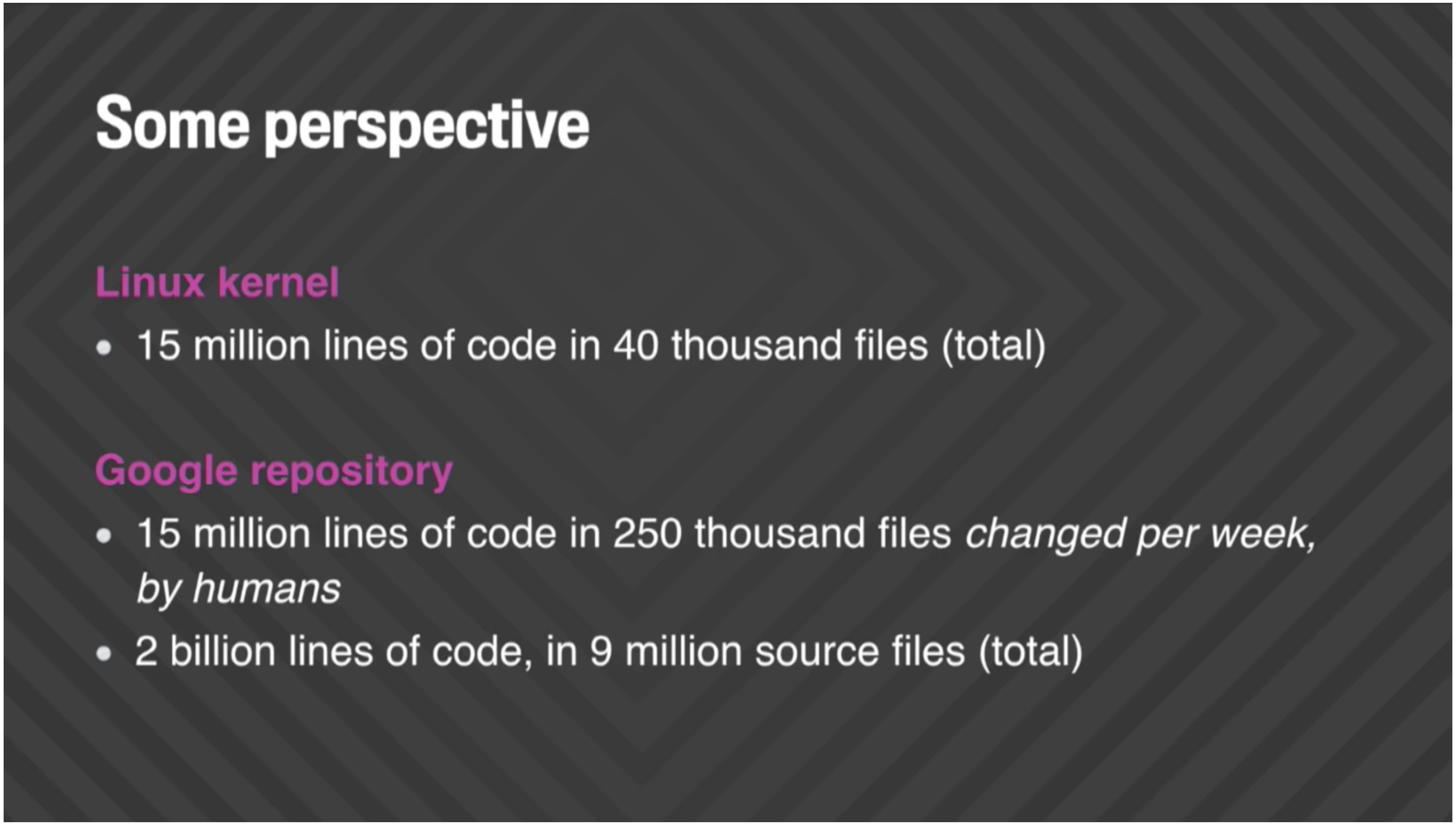
2016 numbers

## Google Speed and Scale

- >30,000 developers in 40+ offices
- 13,000+ projects under active development
- 30k submissions per day (1 every 3 seconds)

- All builds from source
- 30+ sustained code changes per minute with 90+ peaks
- 50% of code changes monthly
- 150+ million test cases / day, > 150 years of test / day
- Supports continuous deployment for all Google teams!

# Google code base vs. Linux kernel code base



## Some perspective

**Linux kernel**

- 15 million lines of code in 40 thousand files (total)

**Google repository**

- 15 million lines of code in 250 thousand files *changed per week, by humans*
- 2 billion lines of code, in 9 million source files (total)

## Managing a huge monorepo

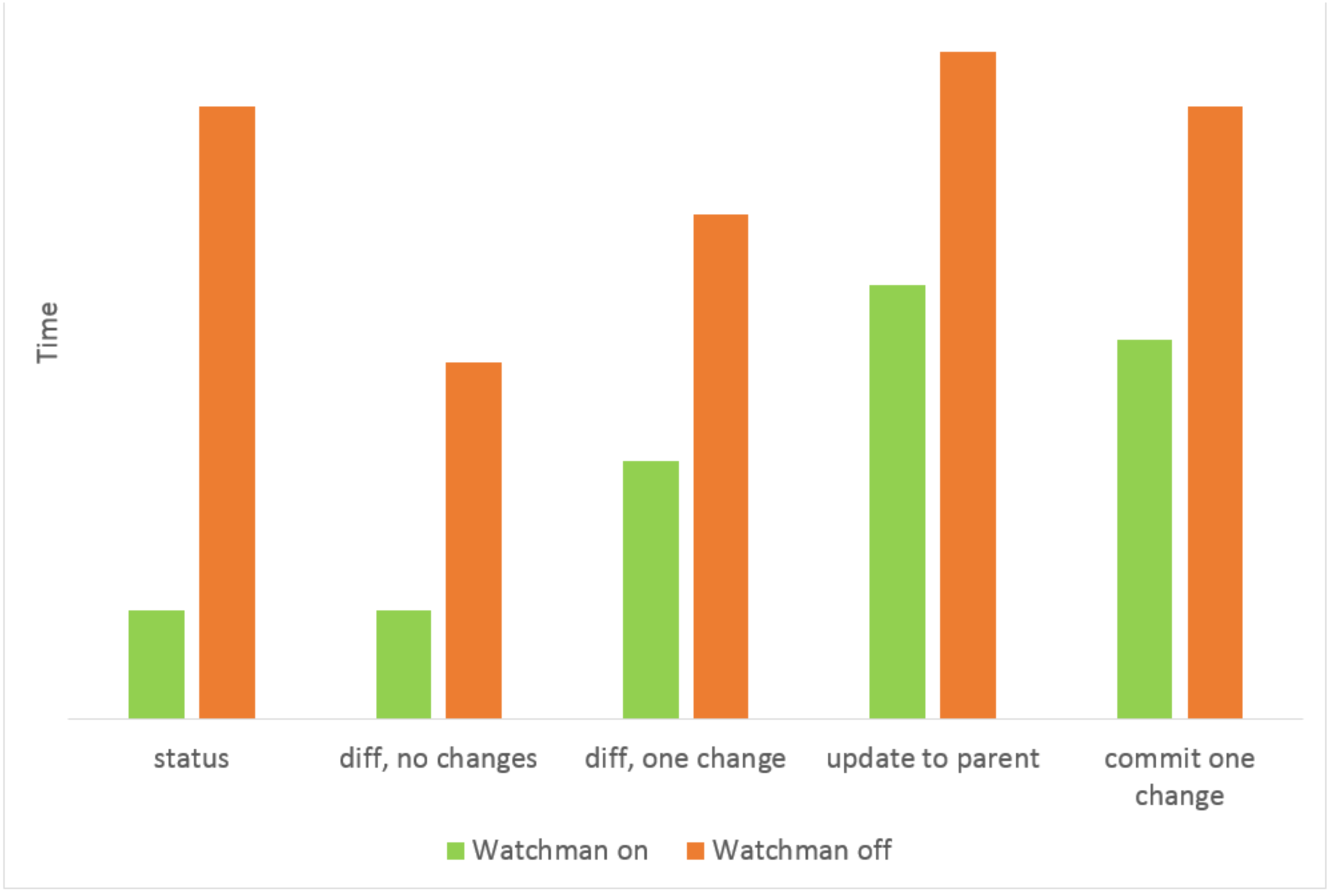
- Automated testing…
- Lots of automation…
- Smart tooling...

## Version control for a monorepo

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

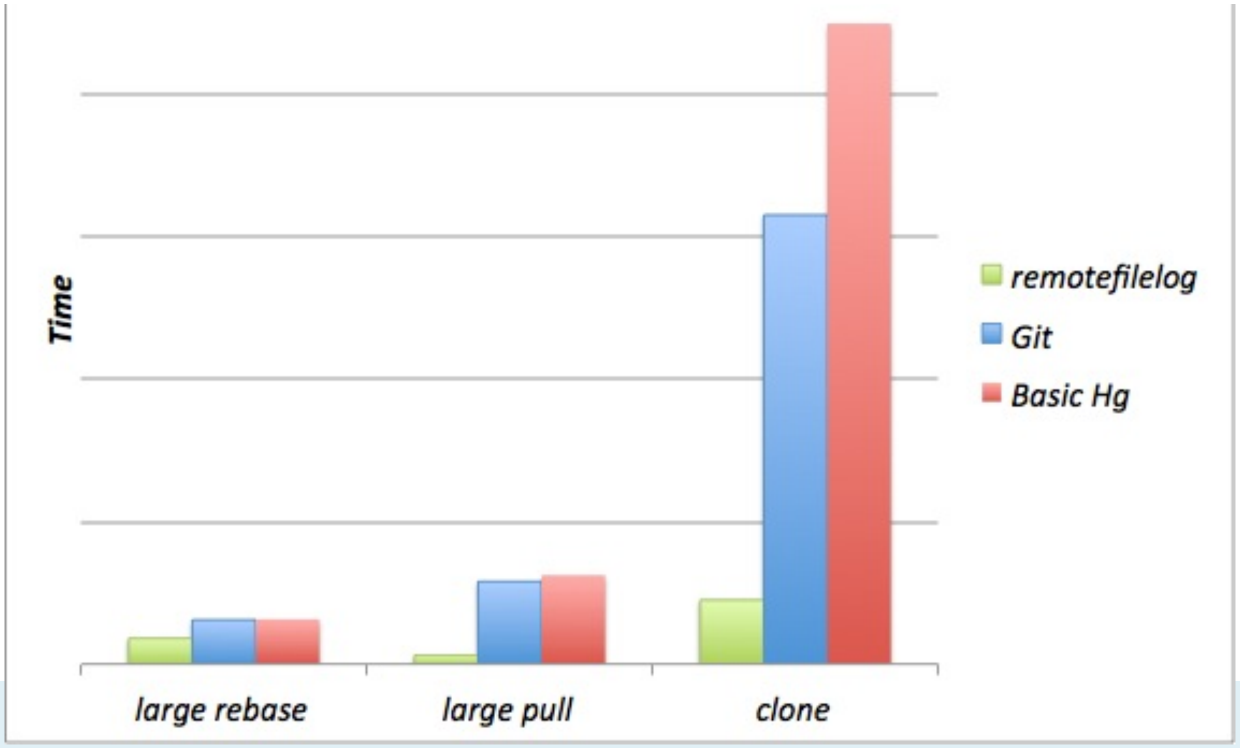
# Version control for a monorepo

- Use build system's file monitor, Watchman, to see which files have changed → **5x faster "status" command**



# Version control for a monorepo

- Sparse checkouts → **10x faster clones and pulls**
  - `clone` and `pull` download only the commit metadata, omit the files
  - When a user performs an operation that needs the contents of files (such as `checkout`), download the file contents on demand



## Summary

- DevOps brings development and operations together
  - Well-attuned to a modern development process
- Monorepos provide convenience, can reduce developer effort
  - …at the expense of requiring custom tooling