Devops

Rohan Padhye and **Michael Hilton**

Administrivia

Mid-semester grades

Midterm

HW3 – Please schedule interviews soon

institute for SOFTWARE RESEARCH

institute for SOFTWARE RESEARCH

Retrospectives

Retrospectives

“the purpose of the Sprint Retrospective is to plan ways to increase quality and effectiveness.” –Scrum.org

We often use three questions:

What should we:

- Start doing?
- Stop doing?
- Keep doing?

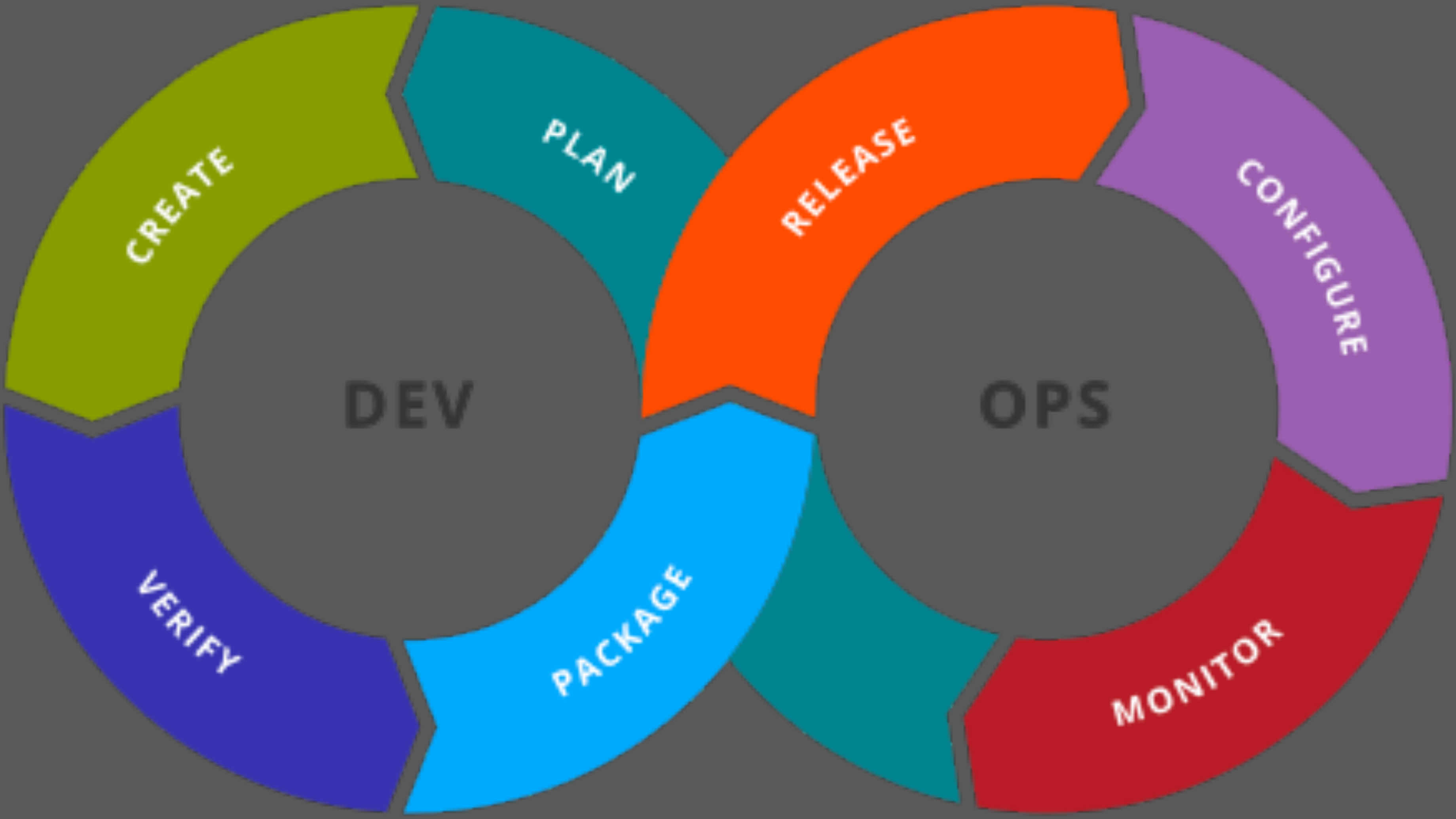
Devops

Learning Goals

Articulate the various purposes of a design document.

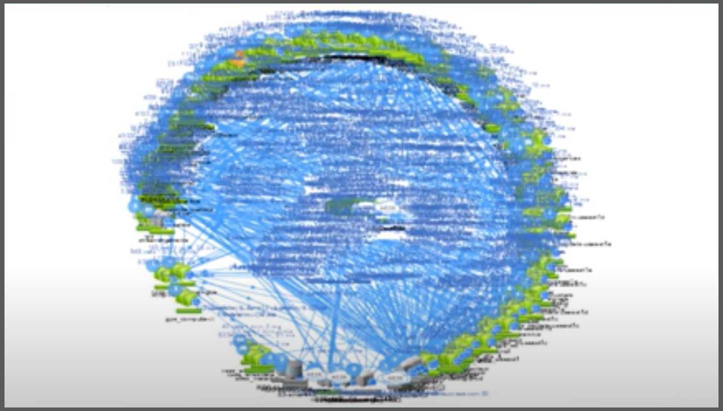
Use design documentation to ensure that the correct thing is being implemented.

Write useful, clear, high-quality design documentation.


CREATE
PLAN
RELEASE
CONFIGURE
DEV
OPS
VERIFY
PACKAGE
MONITOR

Netflix

Netflix: Microservice Architecture



- 100s of microservices
- 1,000s of production changes per day
- 10,000s of virtual machines
- 100,000s of customer interactions per second
- 1,000,000s of metrics per minute (actually, 2 million)

- 81.5 million customers

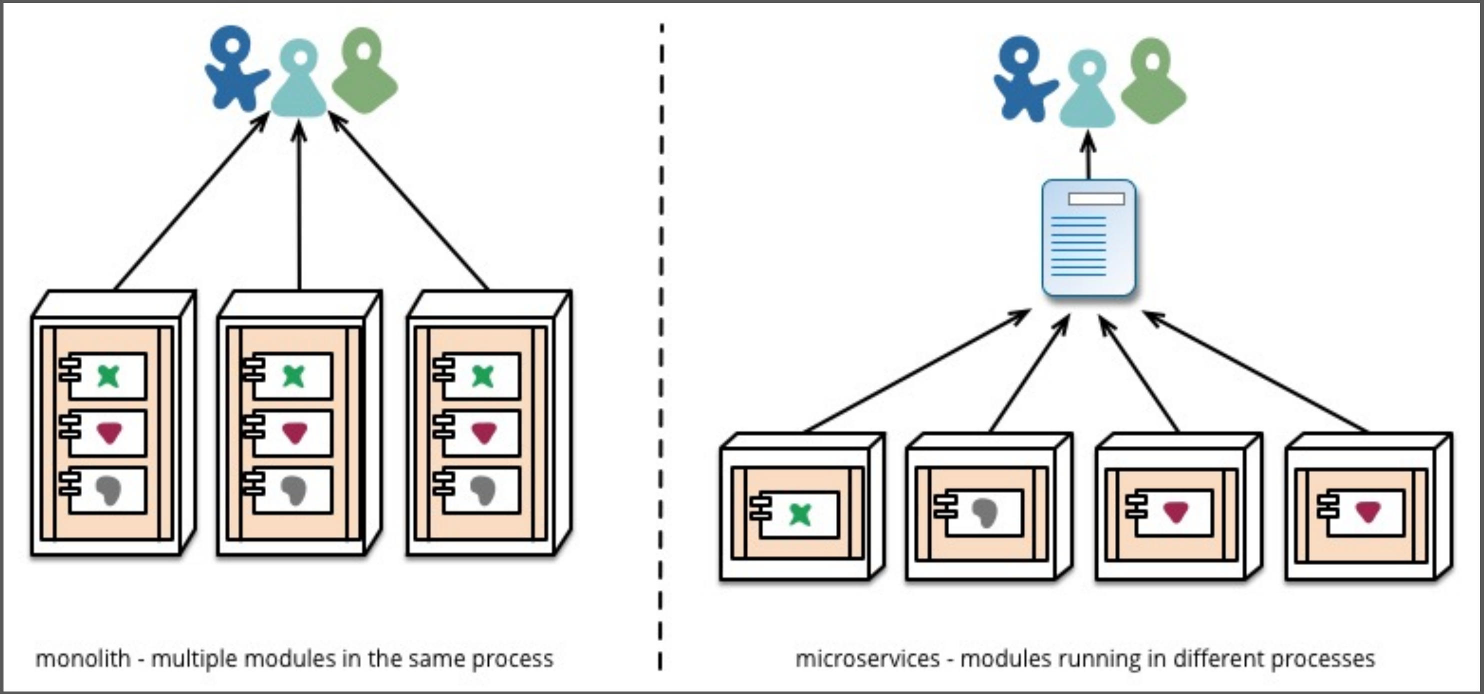
- 10s of operations engineers
- no single engineer knows the entire application

Activity
What were some of the challenges of
running a microservice architecture of this scale?

as of 2018, reference: https://www.youtube.com/watch?v=UTKIT6STSVM

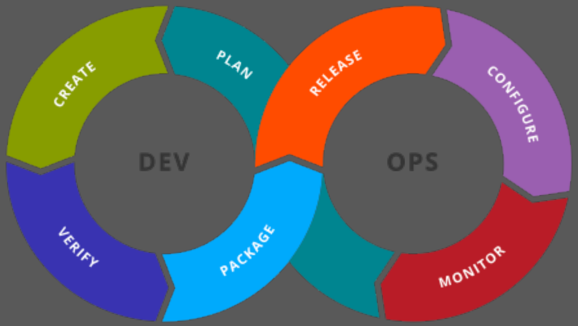
Brainstorm: Microservices

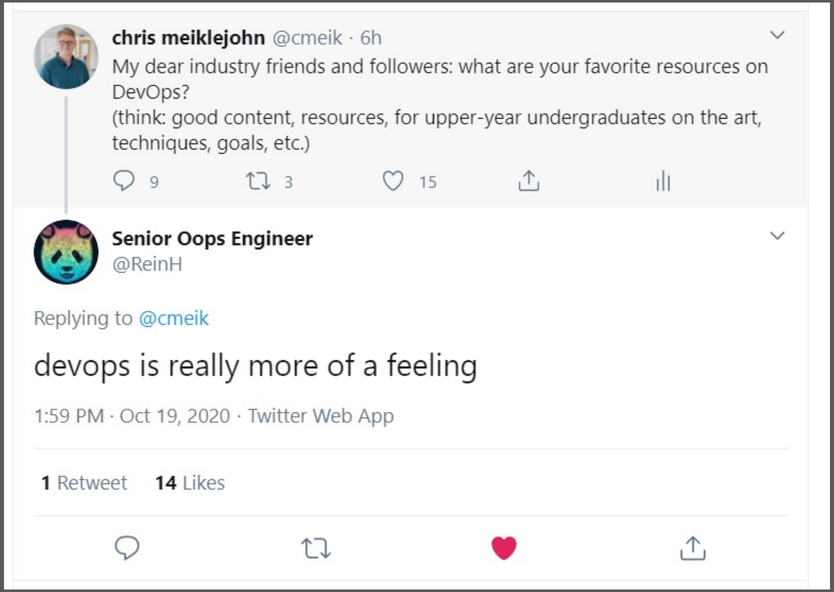
Deployment and Evolution


monolith - multiple modules in the same process
microservices - modules running in different processes

Source: http://martinfowler.com/articles/microservices.html

What is DevOps?





chris meiklejohn @cmeik · 6h
My dear industry friends and followers: what are your favorite resources on DevOps?
(think: good content, resources, for upper-year undergraduates on the art, techniques, goals, etc.)
9 3 15

Senior Oops Engineer
@ReinH

Replying to @cmeik

devops is really more of a feeling

1:59 PM · Oct 19, 2020 · Twitter Web App

1 Retweet 14 Likes

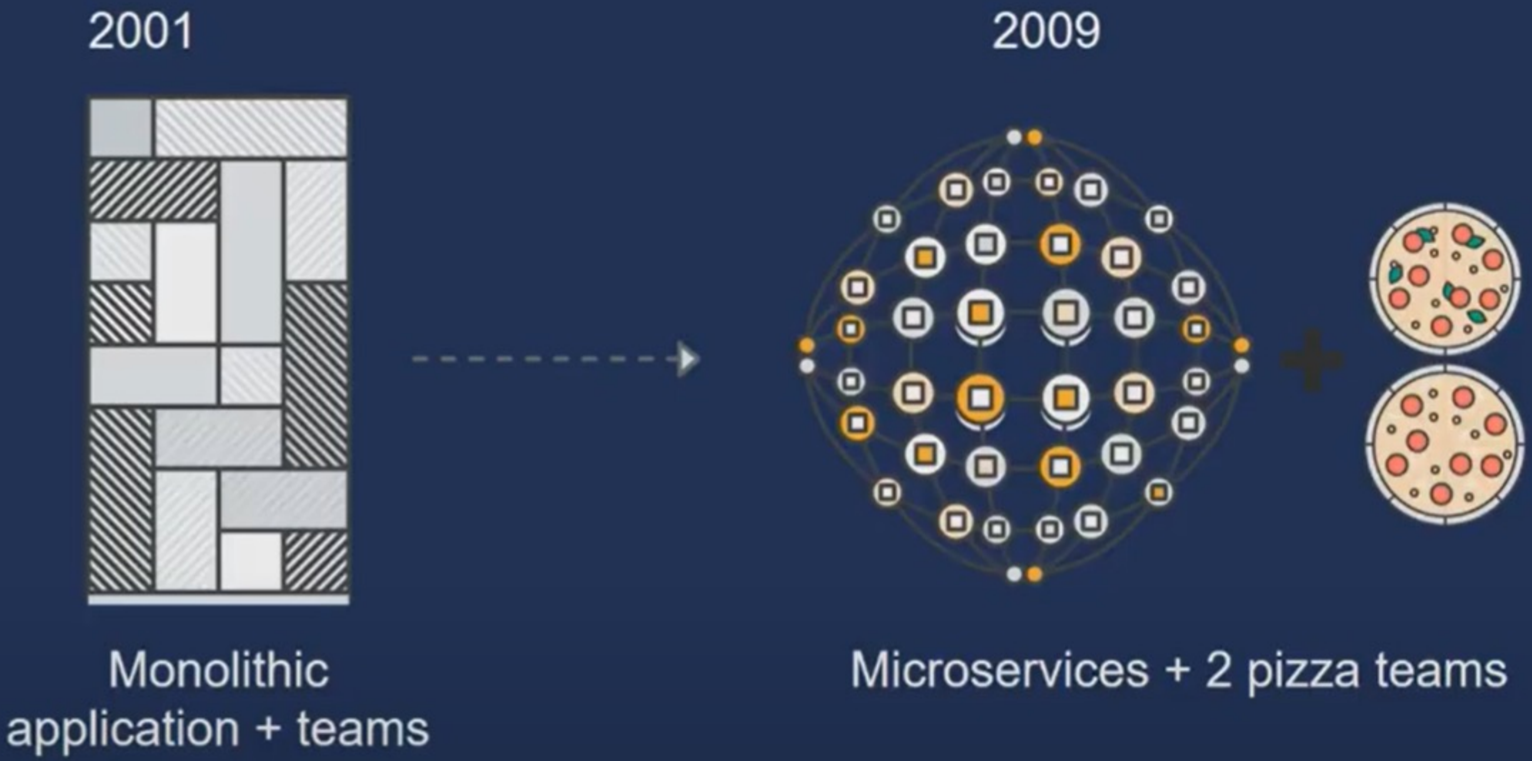
Bringing together two traditionally separate groups within software organizations
- **Development**, typically *measured on features completed*, code shipped
- **Operations**, typically *measured through stability, reliability, availability*

Benefits:
- *Increased* **Velocity**: how quickly products and applications are pushed to release
- *Increased* **Quality**: successful delivery of features and products

reference: https://www.youtube.com/watch?v=UbtB4sMaaNM

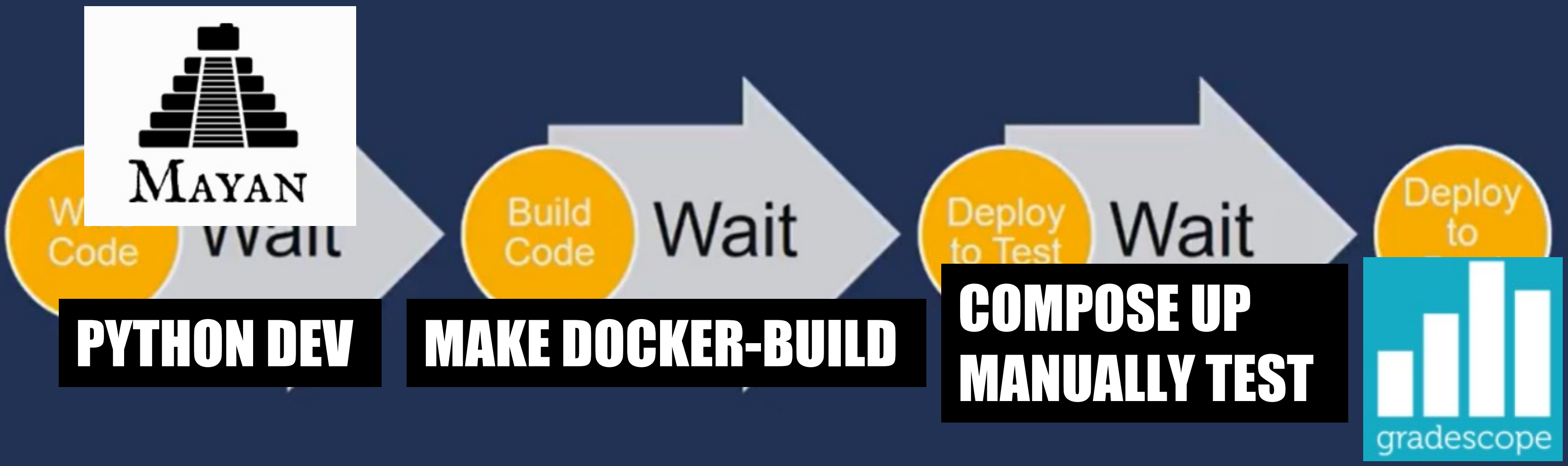
Amazon

Development transformation at Amazon: 2001-2009



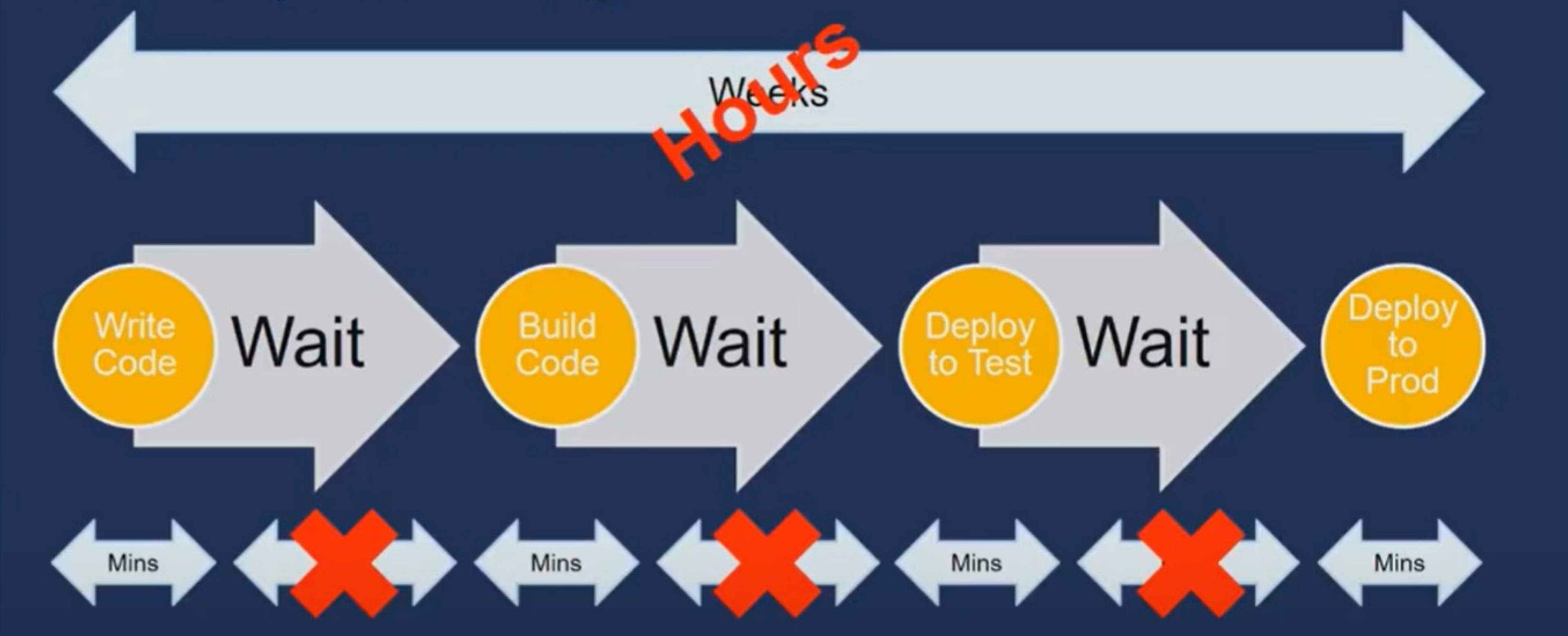
reference: https://www.youtube.com/watch?v=mBU3AJ3j1rg

We were just waiting.



reference: https://www.youtube.com/watch?v=mBU3AJ3j1rg

We were just waiting.



reference: https://www.youtube.com/watch?v=mBU3AJ3j1rg

How do we get to DevOps?

Goals:

1. ***Technological:*** Automated process for moving code from dev to release.

   Starting with check-in, build, unit test, build artifact,
   integration test, load test, as moves through stage to production,
   finally, with monitoring and other telemetry.

2. ***Cultural:*** Building cohesive, multidisciplinary teams.

   Typically, developers are the "first responders" when things go bad in production.
   Sense of "ownership" by the developer all the way from inception to release.

reference: https://www.youtube.com/watch?v=UbtB4sMaaNM

What can it look like when it's done?

Netflix Spinnaker (open-source CI/CD fully automated pipeline):

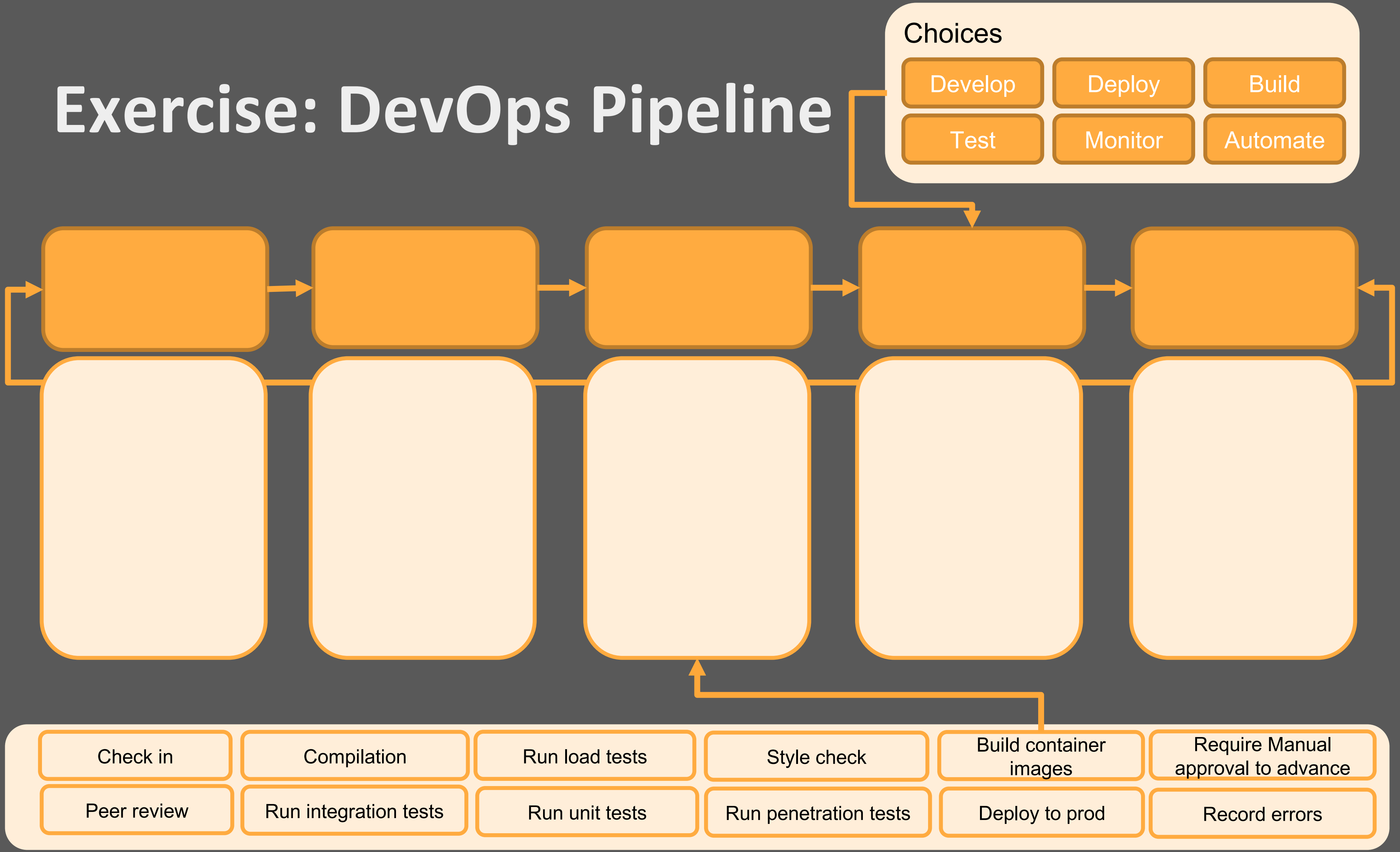
- Takes code from code repository to production.
- Allows developers to specify required tests.
- Determines where, how code should be run in system (e.g., replication, placement.)
- Supports canary deployments, traffic management.
- Just publish the repo!

reference: https://www.youtube.com/watch?v=UTKIT6STSVM

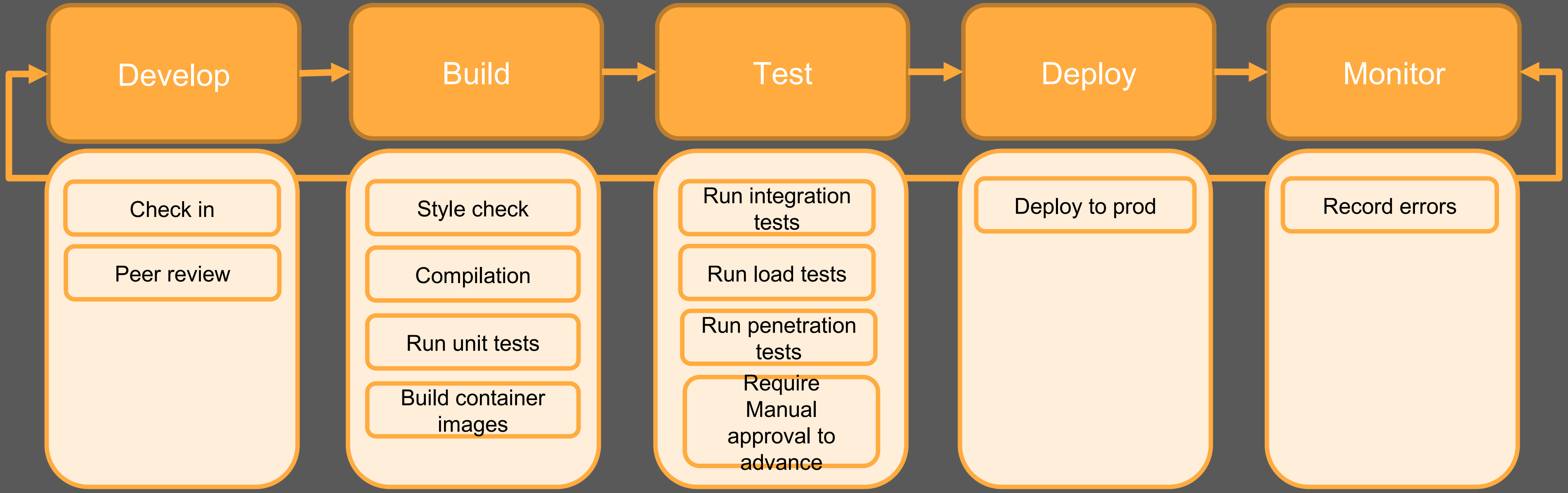
5x	440x	46x	44%
lower change failure rate	faster from commit to deploy	more frequent deployments	more time spent on new features and code

reference: Puppet State of the DevOps Report 2017

Exercise: DevOps Pipeline



A Typical DevOps Pipeline



What do we need to practice for DevOps? 1

Continuous Integration (CI)

1. Constant testing as code is checked-in/pushed to the repository (e.g., GH hooks, etc.)
2. Verify the build process works (i.e., parsing, compilation, code generation, etc.)
3. Verify unit tests pass, style checks pass, other static analysis tools.
4. Build artifacts

Continuous Delivery & Deployment (CD)

1. Moving build artifacts from test -> stage -> prod environments.
   Environments always differ! (e.g., ENV, PII, data, etc.)
2. Gate code, if necessary, from advancing without manual approval.
   Useful when initially transitioning applications into a modern DevOps pipeline.

reference: https://www.youtube.com/watch?v=mBU3AJ3j1rg

What do we need to practice for DevOps? 2

Infrastructure as Code

1. Required resources (e.g., cloud services, access policies, etc.) are created by code.
   No UI provisioning, no manual steps (avoid: easy to forget, time consuming!)
2. "Immutable Infrastructure"
   No update-in-place (e.g., SSH to server.)
   Replace with new instances, decommission old instances.
3. *Nothing to prod without it being in code, checked-in, versioned along side code!*

Observability (Monitoring, Logging, Tracing, Metrics)

1. Be able to know how your application is running in production
2. Track and analyze low-level metrics on performance, resource allocation
3. Capture high-level metrics on application behavior
   1. What's "normal"?
   2. What's abnormal?

reference: https://www.youtube.com/watch?v=mBU3AJ3j1rg

CI/CD

Continuous Integration (CI)

1. Commit and check-in code frequently (always can squash later)
2. Commits build on previous commits (know precisely where the build breaks)
3. Automated feedback and testing on commits
4. Artifact creation (e.g., container images, WAR files, etc.)
5. Ensure code, supporting infrastructure, documentation are all versioned together

Continuous Deployment (CD)

1. Artifacts automatically shipped into test, stage, production environments
2. Prevents “manual” deployment, avoids “manual” steps, early detection of problems
3. Can be tied to a “manual” promotion technique to advance through environments
4. Multi-stage deployment with automatic rollback on failure detection

reference: https://www.youtube.com/watch?v=mBU3AJ3j1rg

Deploying Code

Nightly Build

Build code and run smoke test (Microsoft 1995)

Benefits

- it minimizes integration risk
- It reduces the risk of low quality
- it supports easier defect diagnosis
- it improves morale

Ring Deployment: Microsoft

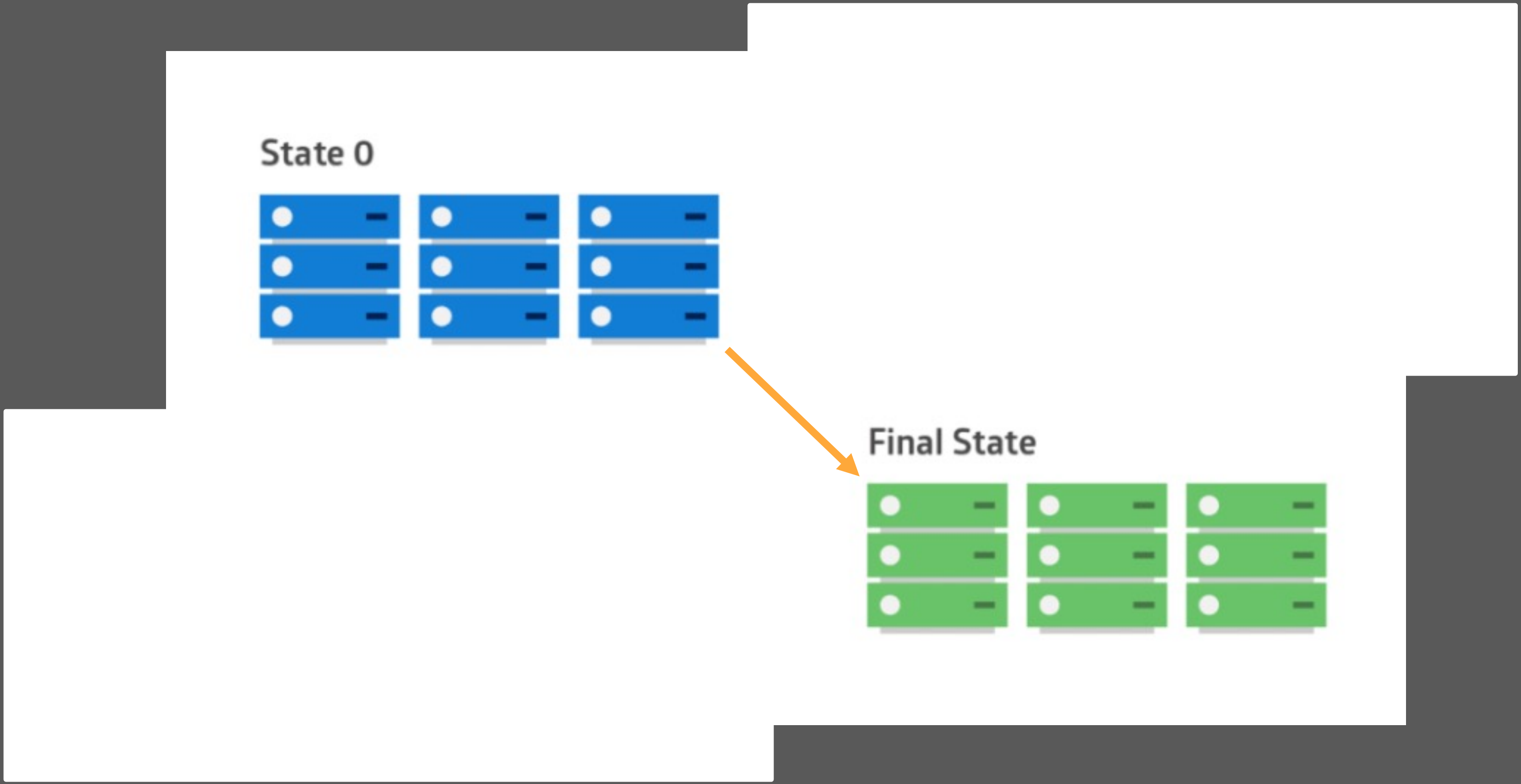
- Commits flow out to rings, de-flight if issue
- For example:
  - Ring 0 => Team
  - Ring 1 => Dogfood
  - Ring 2 => Beta
  - Ring 3 => Many
  - Ring 4 => All
- Windows 10 Insiders Program
  - Dev Channel (weekly builds of Windows 10)
  - Beta Channel (dev + validated updates by Microsoft)
  - Release Preview Channel (highest quality, validated updates)

Rapid Release/Mozilla

If deployment requires on-prem deployment, say a web browser

- There are four channels: *Nightly*, Alpha, Beta, Release Candidate
- Code flows every 2 weeks to next channel, unless fast tracked by release engineer.
- Involve corporate customer specific testing in testing (Practice also used by IBM, Redhat)
- same for Windows Edge browser Insiders Program:
  - Canary: nightly builds
  - Dev: weekly builds
  - Beta: 6 weeks

“Big bang” deployments



reference: https://dev.to/mostlyjason/intro-to-deployment-strategies-blue-green-canary-and-more-3a

Fast to Deploy, Slow to Release

Chuck Rossi at Facebook: *“Get your s*** in, fix it in production”*

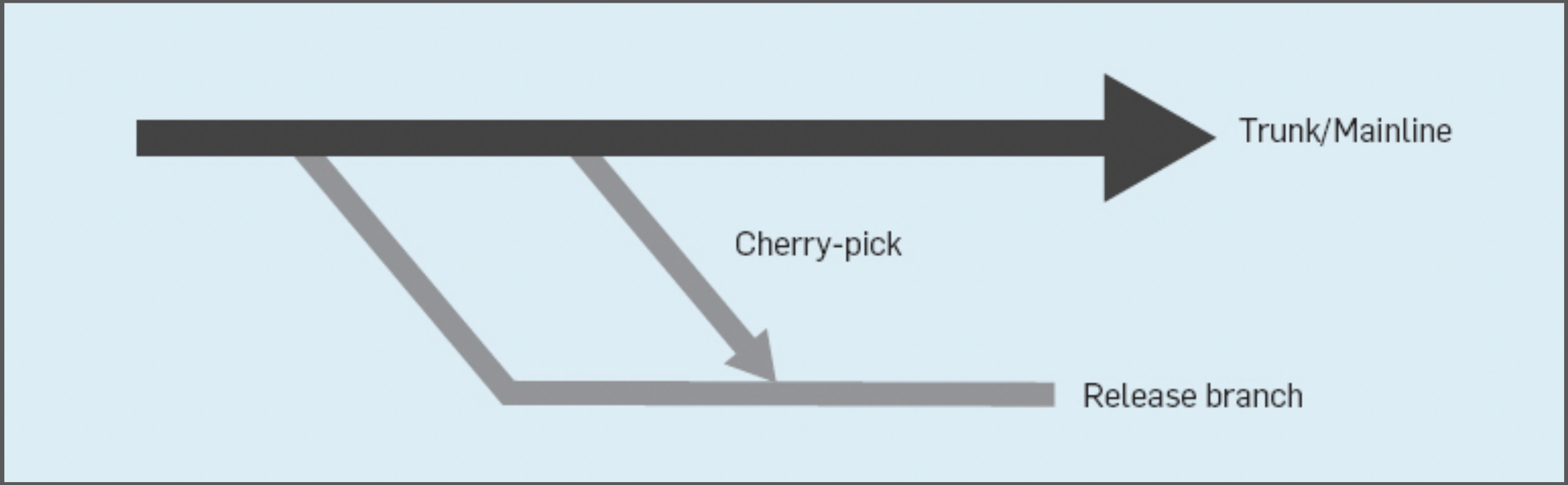
Dark Launches at Instagram

Early: Integrate as soon as possible. Find bugs early. Code can run in production about 6 months before being publicly announced.

Often: Reduce friction. Try things out. See what works. Push small changes just to gather metrics, feasibility testing. Large changes just slow down the team. Do dark launches, to see what performance is in production, can scale up and down. *"Shadow infrastructure" is too expensive, just do in production.*

Incremental: Deploy in increments. Contain risk. Pinpoint issues.

Facebook process (until 2016)

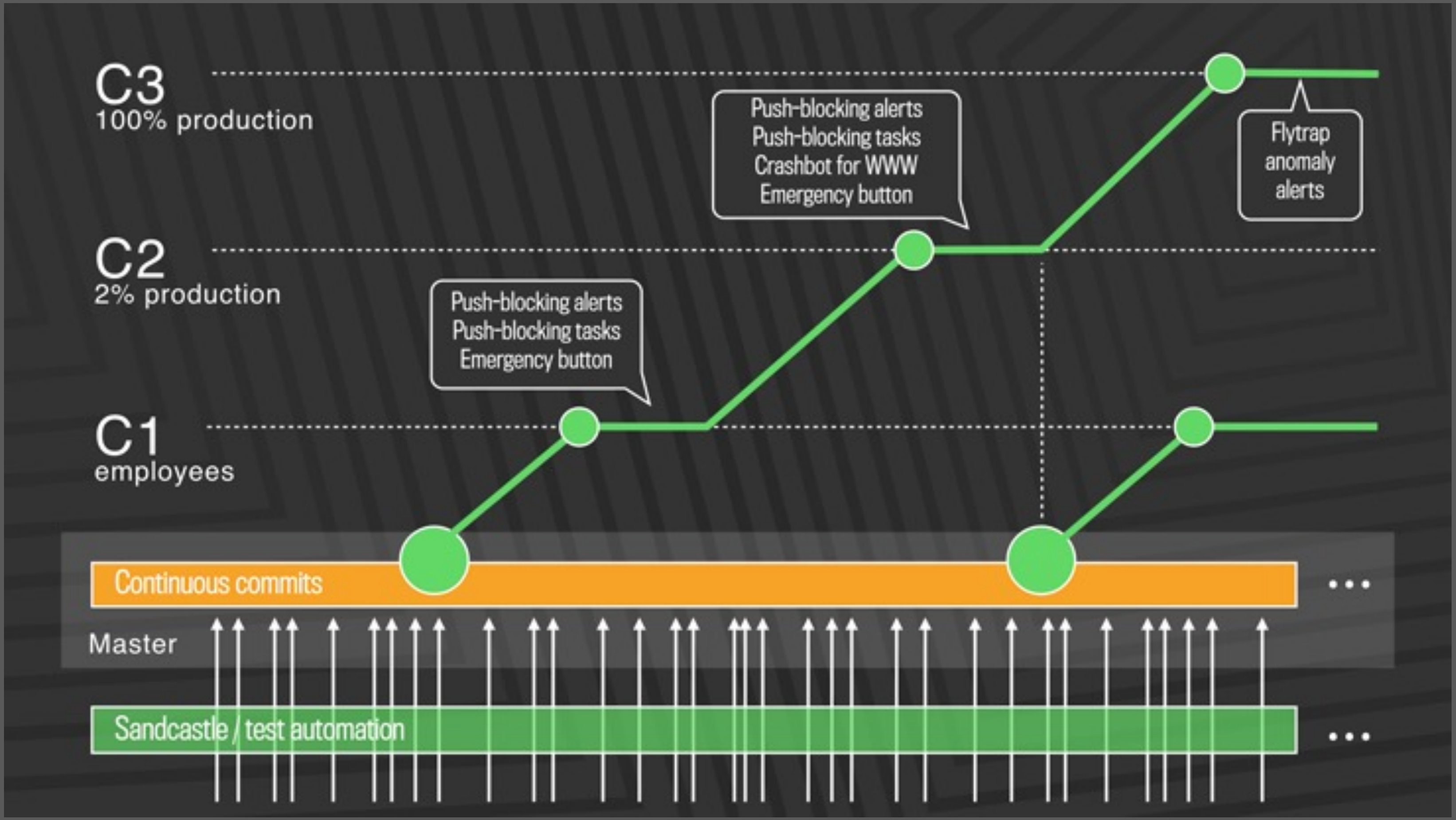


Release is cut Sunday 6pm

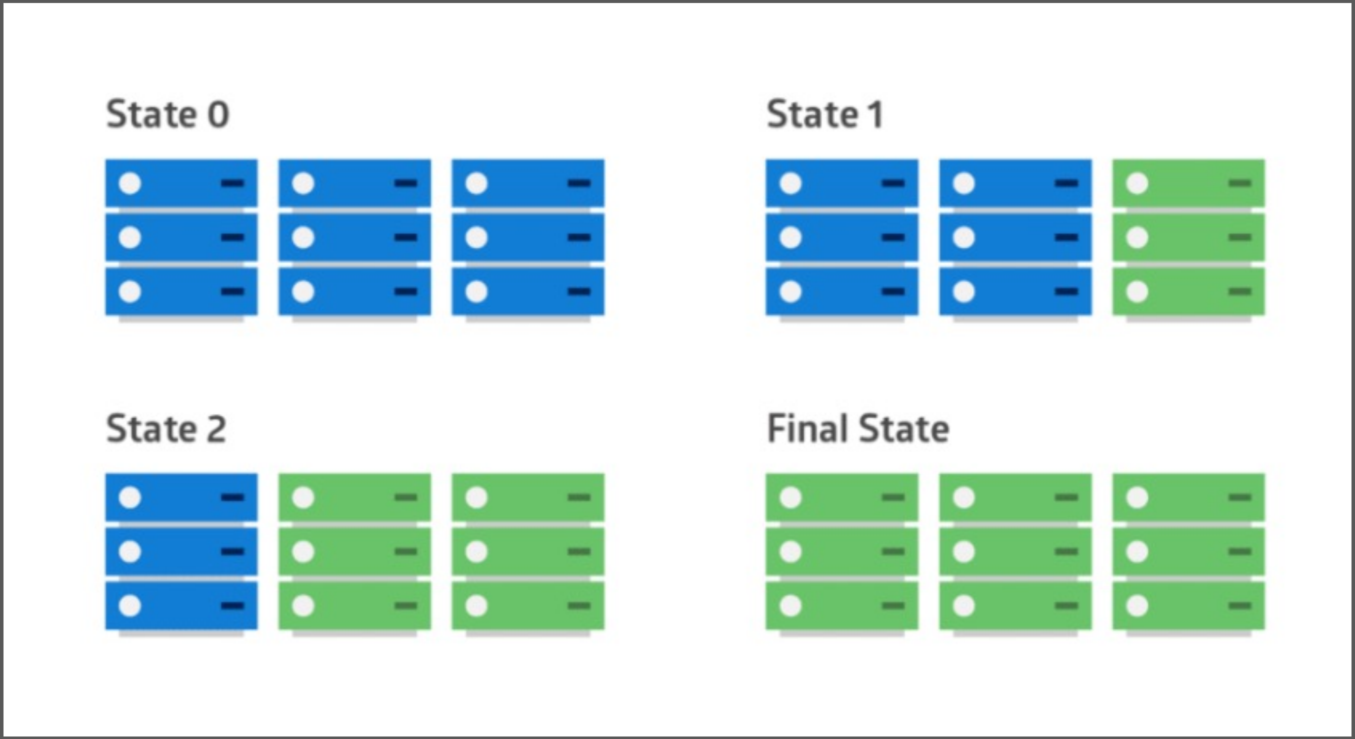
Stabilize until Tuesday, canaries, release. Tuesday push is 12,000 diffs.

Cherry pick: Push 3 times a day (Wed-Fri) 300-700 cherry picks / day.

Facebook quasi-continuous release

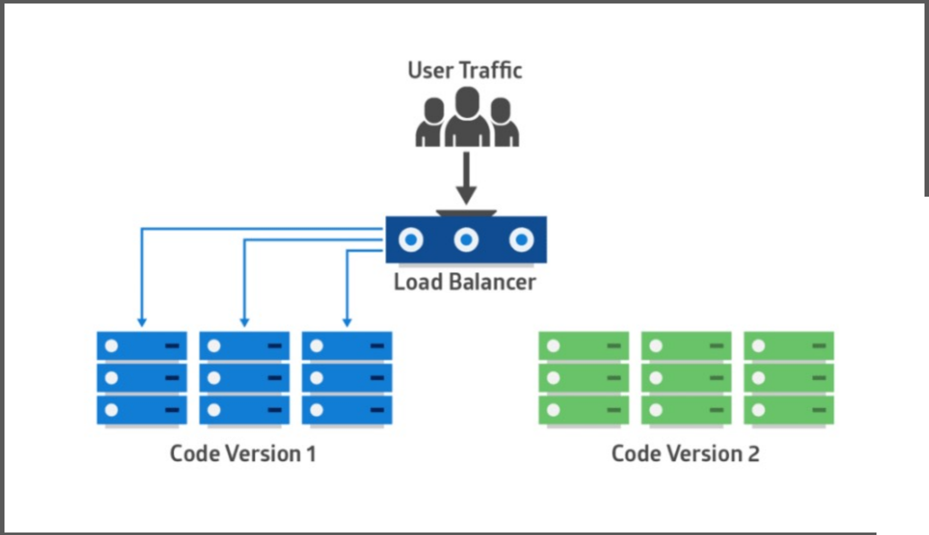


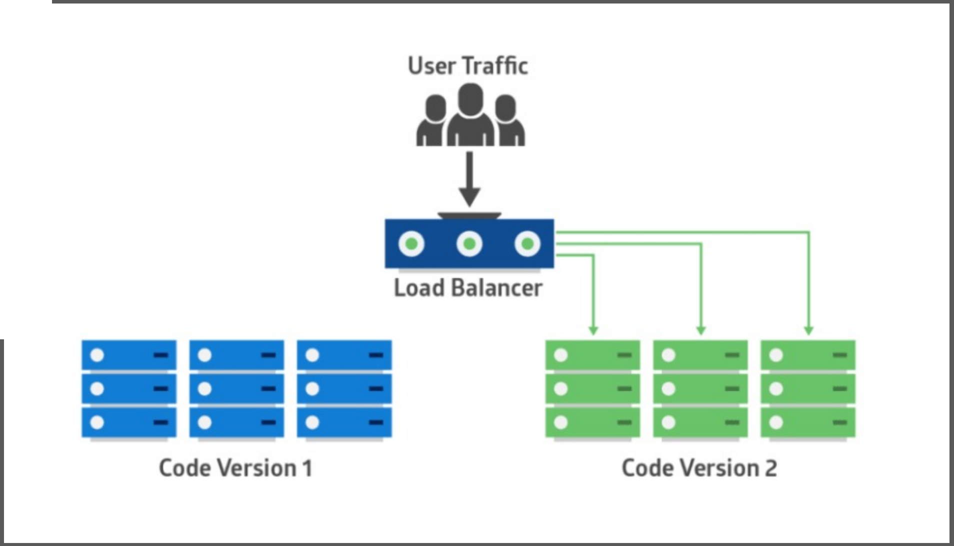
Rolling deployments



reference: https://dev.to/mostlyjason/intro-to-deployment-strategies-blue-green-canary-and-more-3

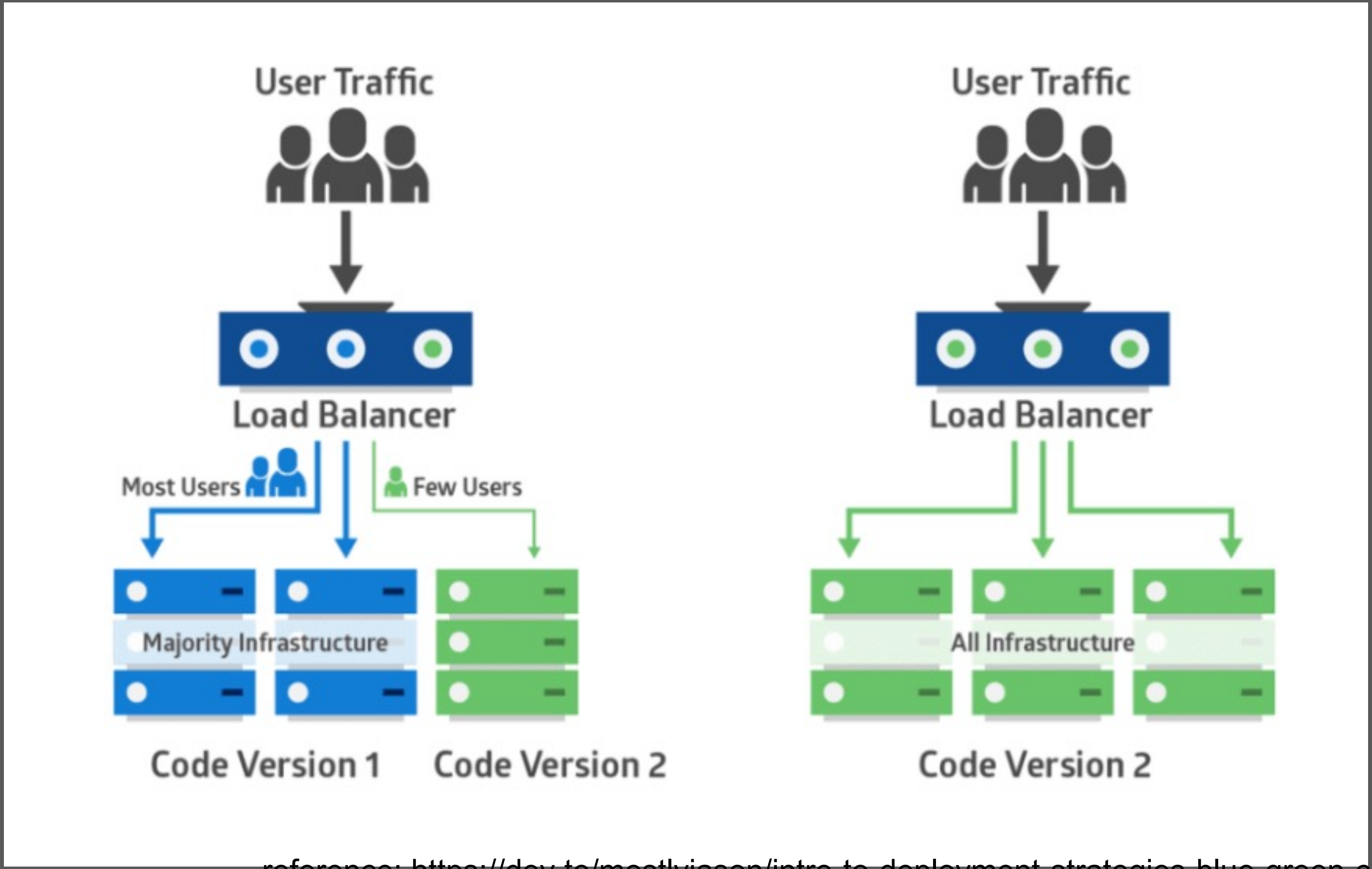
Red/Black (Blue/Green) deployments





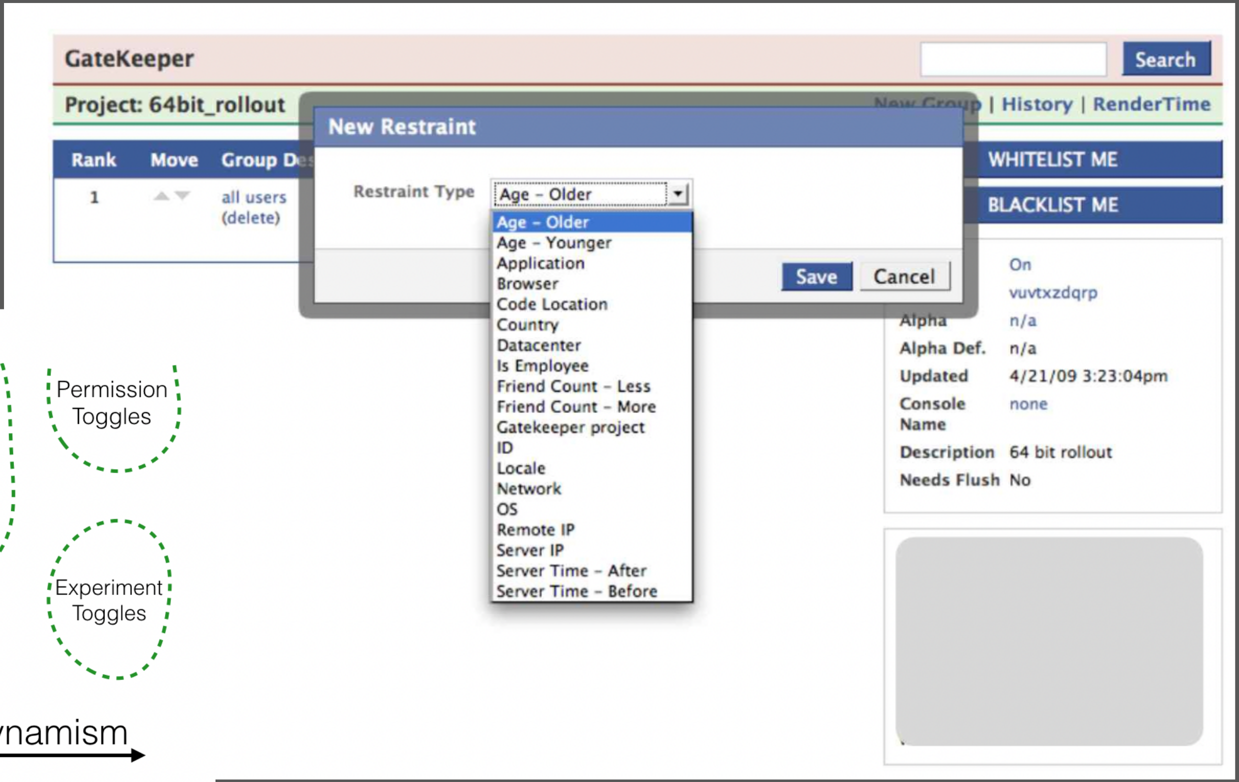
reference: https://dev.to/mostlyjason/intro-to-deployment-strategies-blue-green-canary-and-more-3a

Canary deployments



reference: https://dev.to/mostlyjason/intro-to-deployment-strategies-blue-green-canary-and-more-3a

Feature flags



GateKeeper Search

Project: 64bit_rollout | History | RenderTime

Rank	Move	Group De
1		all users (delete)

New Restraint

Restraint Type: Age – Older

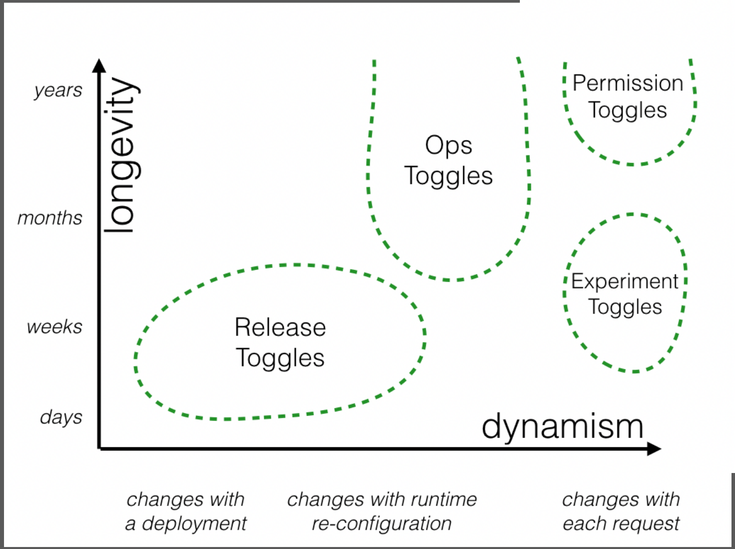
Age – Older
Age – Younger
Application
Browser
Code Location
Country
Datacenter
Is Employee
Friend Count – Less
Friend Count – More
Gatekeeper project
ID
Locale
Network
OS
Remote IP
Server IP
Server Time – After
Server Time – Before

Save Cancel

WHITELIST ME

BLACKLIST ME

	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



Monitoring Production

What is Observability?

"As a philosophy, ***observability*** is our ability as developers to know and discover what is going on in our systems. In practice, it means adding telemetry to our systems in order to measure change and track workflows."

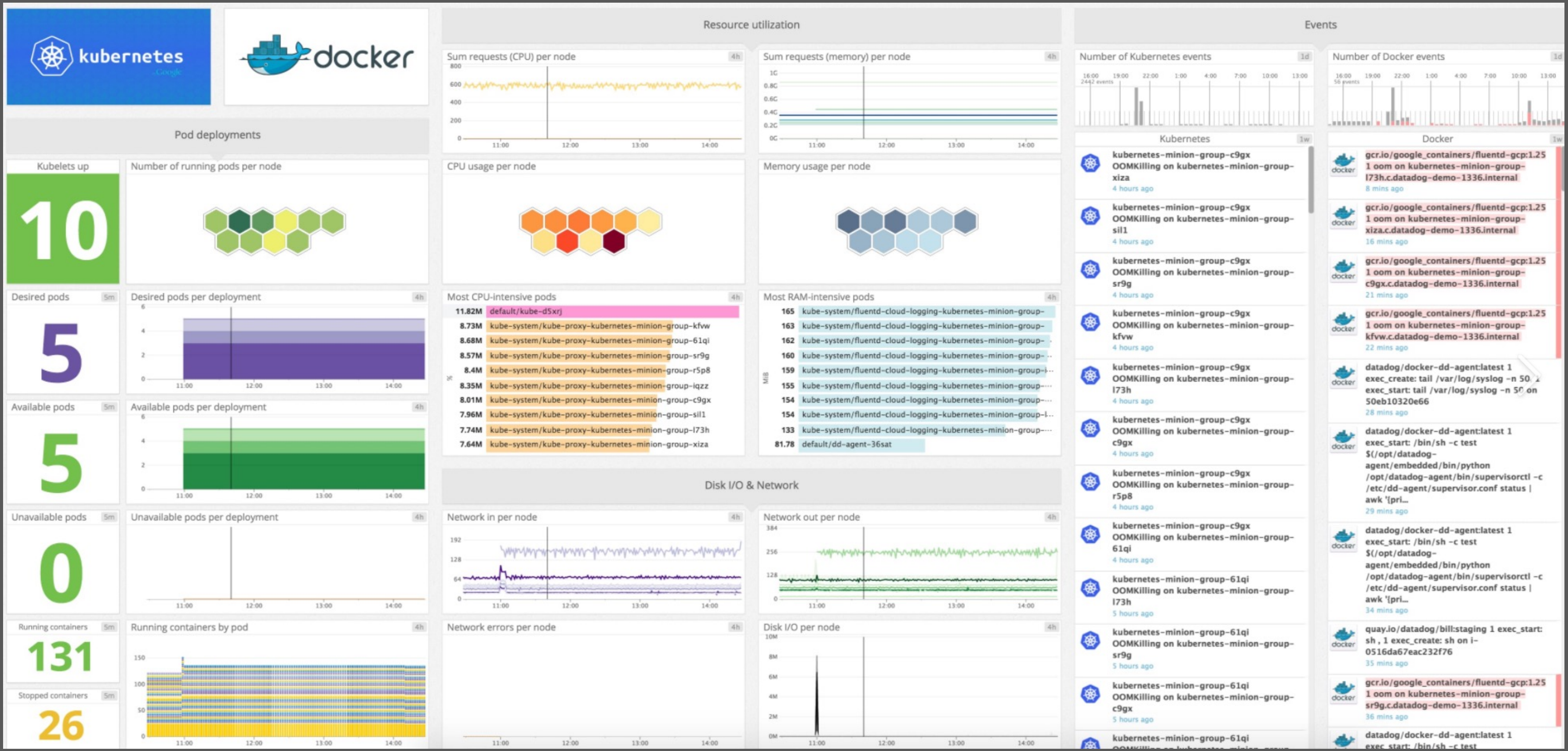
The New Stack, "What is observability?" 28 Feb 2020
https://thenewstack.io/what-is-observability/

Observability: Dashboards

1. What's happening now?
2. What does "normal" behavior look like?
3. What does it look like when something's gone (or is going) wrong?
4. Can I correlate events to changes in the actual graphs?

reference: https://www.youtube.com/watch?v=mBU3AJ3j1rg

Observability: Dashboard Example



Most CPU-intensive pods

11.82M	default/kube-d5xrj
8.73M	kube-system/kube-proxy-kubernetes-minion-group-kfvw
8.68M	kube-system/kube-proxy-kubernetes-minion-group-61qi
8.57M	kube-system/kube-proxy-kubernetes-minion-group-sr9g
8.4M	kube-system/kube-proxy-kubernetes-minion-group-r5p8
8.35M	kube-system/kube-proxy-kubernetes-minion-group-iqzz
8.01M	kube-system/kube-proxy-kubernetes-minion-group-c9gx
7.96M	kube-system/kube-proxy-kubernetes-minion-group-sil1
7.74M	kube-system/kube-proxy-kubernetes-minion-group-l73h
7.64M	kube-system/kube-proxy-kubernetes-minion-group-xiza

Most RAM-intensive pods

165	kube-system/fluentd-cloud-logging-kubernetes-minion-group-
163	kube-system/fluentd-cloud-logging-kubernetes-minion-group-
162	kube-system/fluentd-cloud-logging-kubernetes-minion-group-
160	kube-system/fluentd-cloud-logging-kubernetes-minion-group-…
159	kube-system/fluentd-cloud-logging-kubernetes-minion-group-i…
155	kube-system/fluentd-cloud-logging-kubernetes-minion-group-…
154	kube-system/fluentd-cloud-logging-kubernetes-minion-group-…
154	kube-system/fluentd-cloud-logging-kubernetes-minion-group-l…
133	kube-system/fluentd-cloud-logging-kubernetes-minion-group-…
81.78	default/dd-agent-36sat

Kubernetes

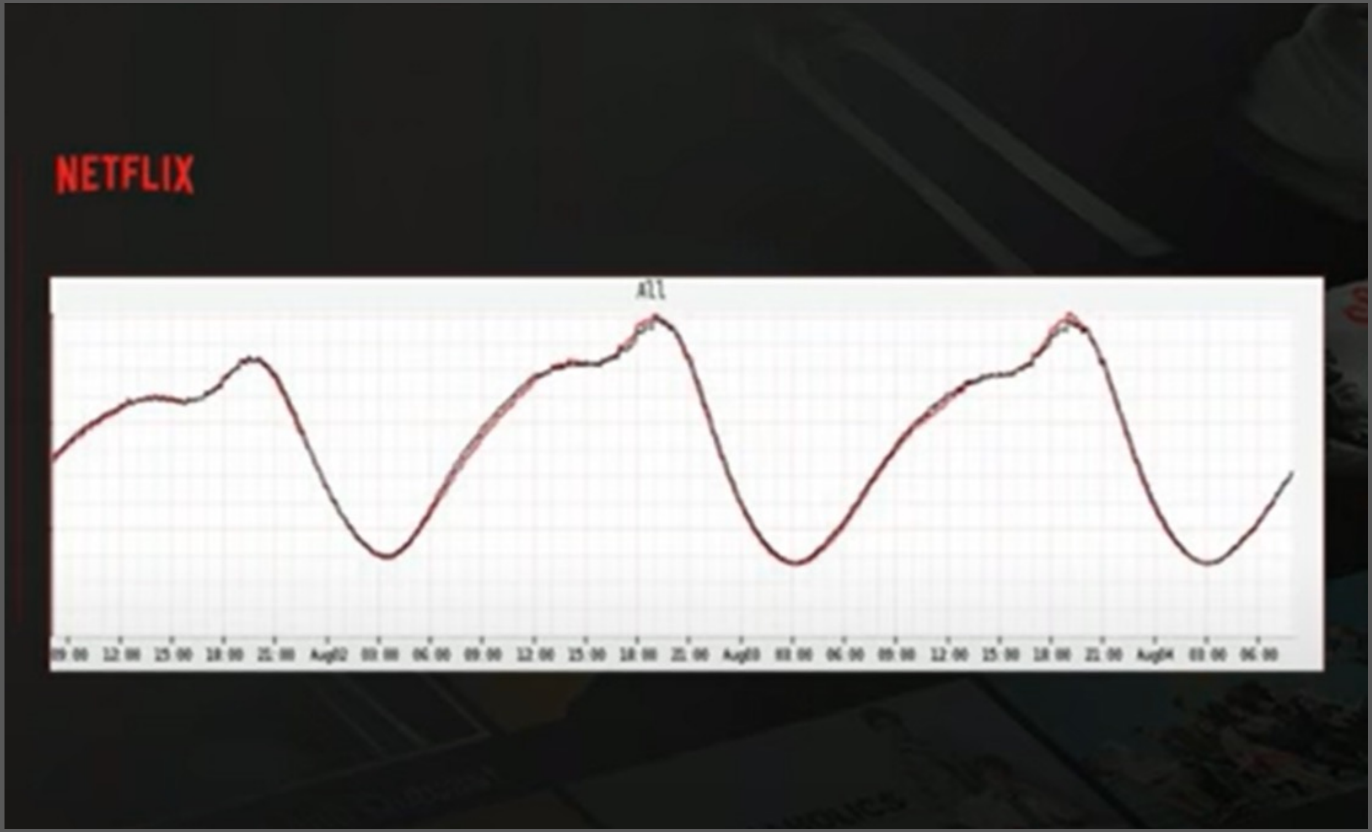
```
kubernetes-minion-group-c9gx
OOMKilling on kubernetes-minion-group-xiza
4 hours ago
kubernetes-minion-group-c9gx
OOMKilling on kubernetes-minion-group-sil1
4 hours ago
kubernetes-minion-group-c9gx
OOMKilling on kubernetes-minion-group-sr9g
4 hours ago
kubernetes-minion-group-c9gx
OOMKilling on kubernetes-minion-group-kfvw
4 hours ago
kubernetes-minion-group-c9gx
OOMKilling on kubernetes-minion-group-l73h
4 hours ago
kubernetes-minion-group-c9gx
OOMKilling on kubernetes-minion-group-c9gx
4 hours ago
kubernetes-minion-group-c9gx
OOMKilling on kubernetes-minion-group-r5p8
4 hours ago
kubernetes-minion-group-c9gx
OOMKilling on kubernetes-minion-group-61qi
4 hours ago
kubernetes-minion-group-61qi
OOMKilling on kubernetes-minion-group-l73h
5 hours ago
kubernetes-minion-group-61qi
OOMKilling on kubernetes-minion-group-sr9g
5 hours ago
kubernetes-minion-group-61qi
OOMKilling on kubernetes-minion-group-c9gx
5 hours ago
kubernetes-minion-group-61qi
```

Docker

```
gcr.io/google_containers/fluentd-gcp:1.25 1 oom on kubernetes-minion-group-l73h.c.datadog-demo-1336.internal
8 mins ago
gcr.io/google_containers/fluentd-gcp:1.25 1 oom on kubernetes-minion-group-xiza.c.datadog-demo-1336.internal
16 mins ago
gcr.io/google_containers/fluentd-gcp:1.25 1 oom on kubernetes-minion-group-c9gx.c.datadog-demo-1336.internal
21 mins ago
gcr.io/google_containers/fluentd-gcp:1.25 1 oom on kubernetes-minion-group-kfvw.c.datadog-demo-1336.internal
22 mins ago
datadog/docker-dd-agent:latest 1 exec_create: tail /var/log/syslog -n 50, 1 exec_start: tail /var/log/syslog -n 50 on 50eb10320e66
28 mins ago
datadog/docker-dd-agent:latest 1 exec_start: /bin/sh -c test $(/opt/datadog-agent/embedded/bin/python /opt/datadog-agent/bin/supervisorctl -c /etc/dd-agent/supervisor.conf status | awk '{pri…
29 mins ago
datadog/docker-dd-agent:latest 1 exec_start: /bin/sh -c test $(/opt/datadog-agent/embedded/bin/python /opt/datadog-agent/bin/supervisorctl -c /etc/dd-agent/supervisor.conf status | awk '{pri…
34 mins ago
quay.io/datadog/bill:staging 1 exec_start: sh , 1 exec_create: sh on i-0516da67eac232f76
35 mins ago
gcr.io/google_containers/fluentd-gcp:1.25 1 oom on kubernetes-minion-group-sr9g.c.datadog-demo-1336.internal
36 mins ago
datadog/docker-dd-agent:latest 1
exec_start: /bin/sh -c test
```

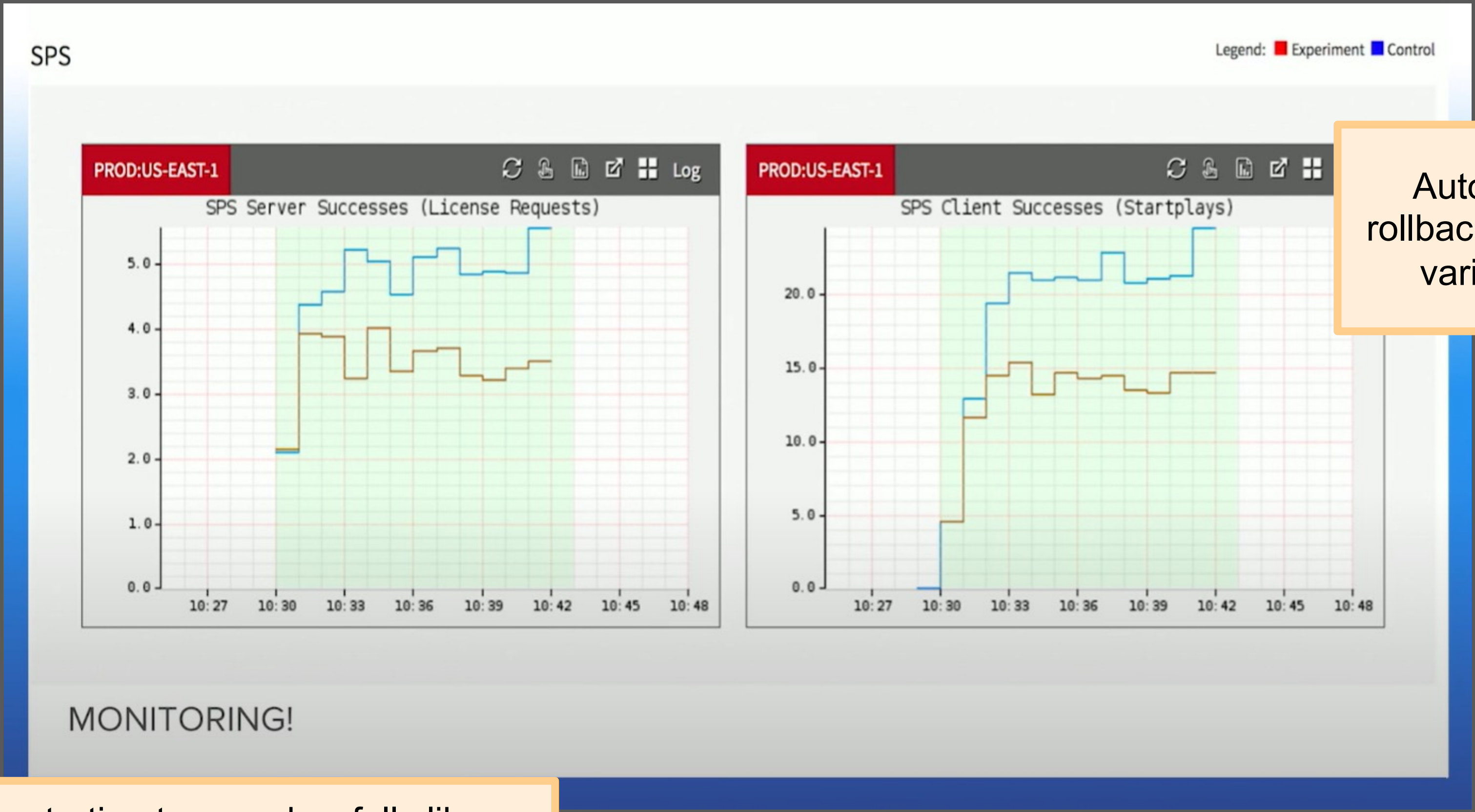
reference: https://datadog-prod.imgix.net/img/blog/monitoring-kubernetes-with-datadog/kubernetes-dashboard.png?fit=max

Observability: Defining “Normal”



reference: https://www.youtube.com/watch?v=vq4QZ4_YDok

Observability: When things aren't "Normal"

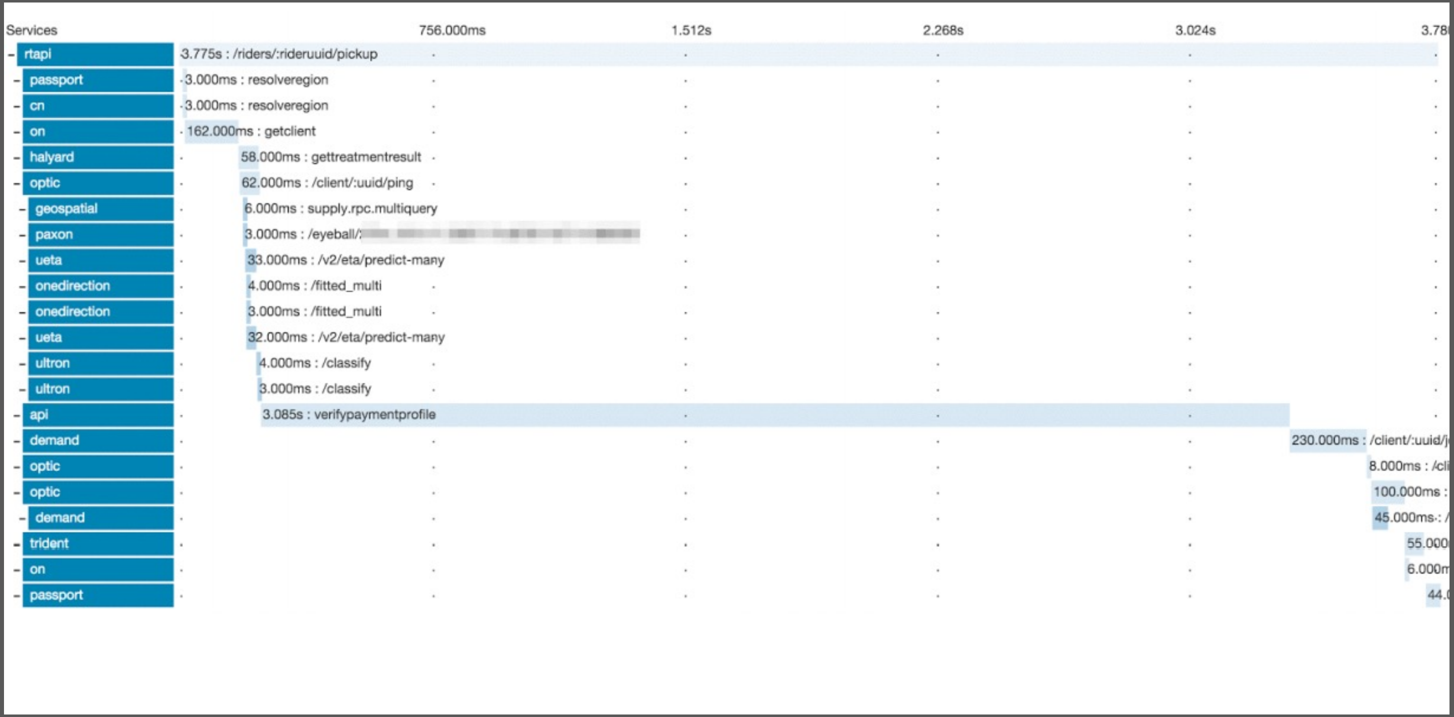


Automatic rollback on high variance!

This is starting to sound awfully like a *quality attribute….*

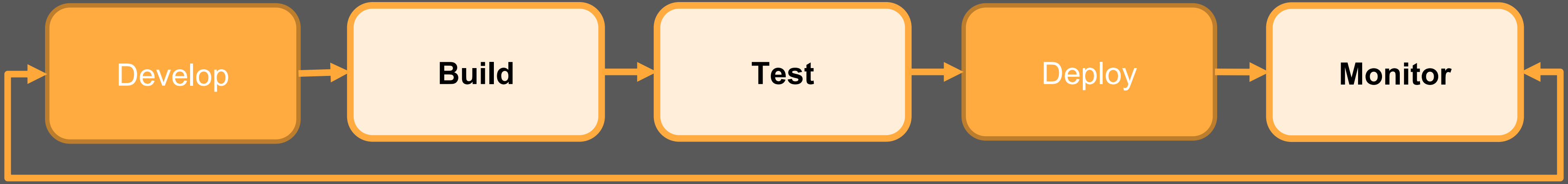
reference: https://www.youtube.com/watch?v=qyzymLlj9ag

Observability: Distributed Tracing



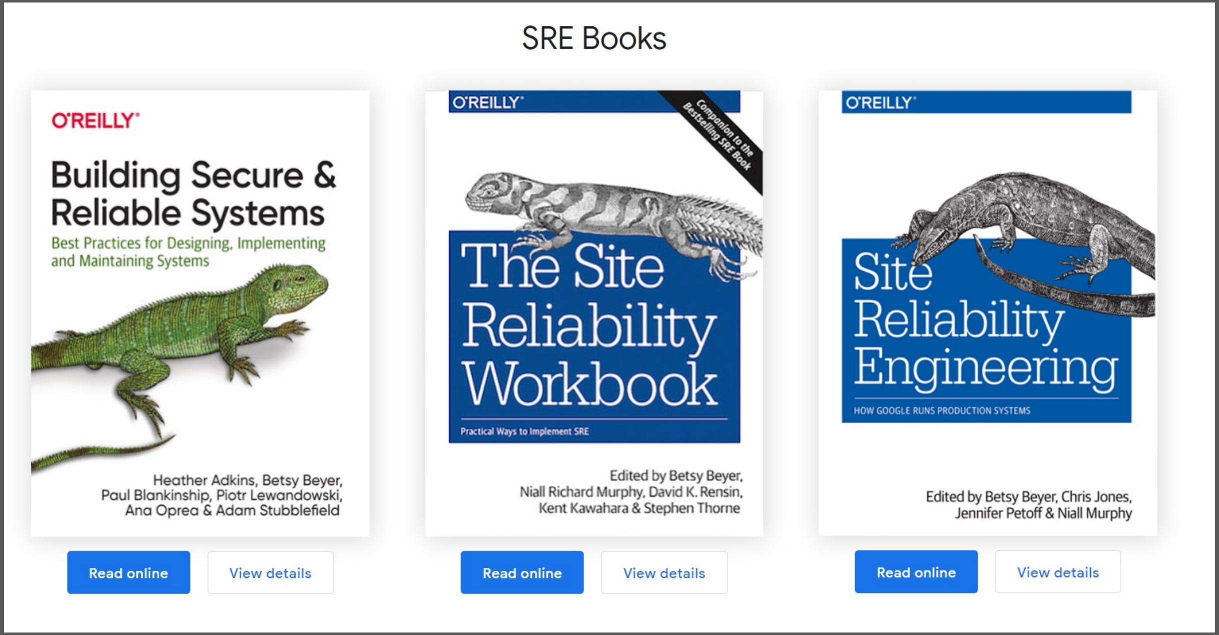
Services	756.000ms 1.512s 2.268s 3.024s 3.78
rtapi	3.775s : /riders/:rideruuid/pickup
passport	3.000ms : resolveregion
cn	3.000ms : resolveregion
on	162.000ms : getclient
halyard	58.000ms : gettreatmentresult
optic	62.000ms : /client/:uuid/ping
geospatial	6.000ms : supply.rpc.multiquery
paxon	3.000ms : /eyeball/[redacted]
ueta	33.000ms : /v2/eta/predict-many
onedirection	4.000ms : /fitted_multi
onedirection	3.000ms : /fitted_multi
ueta	32.000ms : /v2/eta/predict-many
ultron	4.000ms : /classify
ultron	3.000ms : /classify
api	3.085s : verifypaymentprofile
demand	230.000ms : /client/:uuid/j
optic	8.000ms : /cli
optic	100.000ms :
demand	45.000ms : /
trident	55.000
on	6.000m
passport	44.

Homework 2A: Testing Plan



Working in pairs: come up with
one concrete task for the build, test, and monitor stages
to verify that your feature works as designed once deployed to production.

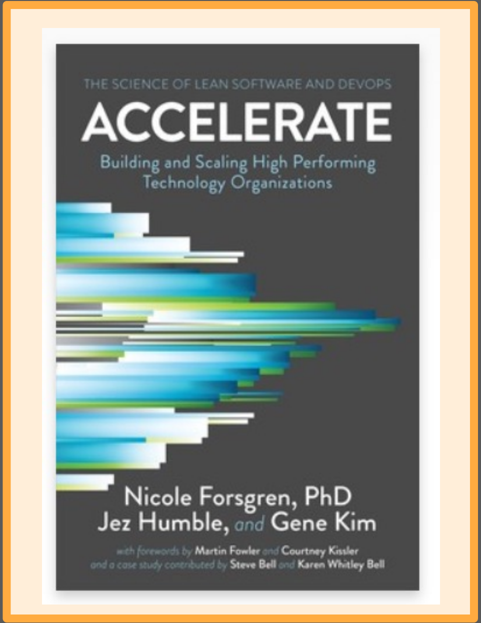
DevOps: More Resources


SRE Books

O'REILLY
Building Secure & Reliable Systems
Best Practices for Designing, Implementing and Maintaining Systems
Heather Adkins, Betsy Beyer, Paul Blankinship, Piotr Lewandowski, Ana Oprea & Adam Stubblefield

O'REILLY
Companion to the Bestselling SRE Book
The Site Reliability Workbook
Practical Ways to Implement SRE
Edited by Betsy Beyer, Niall Richard Murphy, David K. Rensin, Kent Kawahara & Stephen Thorne

O'REILLY
Site Reliability Engineering
HOW GOOGLE RUNS PRODUCTION SYSTEMS
Edited by Betsy Beyer, Chris Jones, Jennifer Petoff & Niall Murphy


THE SCIENCE OF LEAN SOFTWARE AND DEVOPS
ACCELERATE
Building and Scaling High Performing Technology Organizations
Nicole Forsgren, PhD
Jez Humble, and Gene Kim
with forewords by Martin Fowler and Courtney Kissler
and a case study contributed by Steve Bell and Karen Whitley Bell

All available online from CMU Libraries!

Next Week: teams

Questions?