DevOps

Michael Hilton Claire Le Goues

Christopher Meiklejohn

October 20, 2020

Administrativia: Midterm Exam

- Midterm Thursday, October 22nd **released at 00:00 US/Eastern**
  - ***Please sleep!***
  - Released at this time due to not all students being in US/Eastern. (released Noon in Singapore)
  - Available on Gradescope
  - **Upload to Gradescope by Thursday, October 22nd at 23:59 US/Eastern**

- Download PDF from Gradescope, submit in **one** of two ways:
  - Upload document, print, scan, photo with answers;
    - (e.g., annotate PDF, print/scan/photo upload, upload Word/PDF doc of just answers)
  - Answer on Gradescope directly through the UI.

- Questions on Midterm Exam
  - Instructors will be available on Zoom during lecture slot to answer questions.
  - Available on Slack throughout the day, **DM all three instructors** in order to ensure prompt answers from who is available.

    Hours: 8:00 AM US/Eastern – 10:00 PM US/Eastern

Homework 2: Grade Distribution

- Released Homework 2 (A, C, D)
  - **A:** median: 90%, mean: 91%
  - **C:** median: 95%, mean: 87%
  - **D:** median: 100%, mean: 92%

- Homework B is halfway graded, will be done shortly.

- Midsemester grades will be released Monday
  - Will include both midterm and Homework 2B.

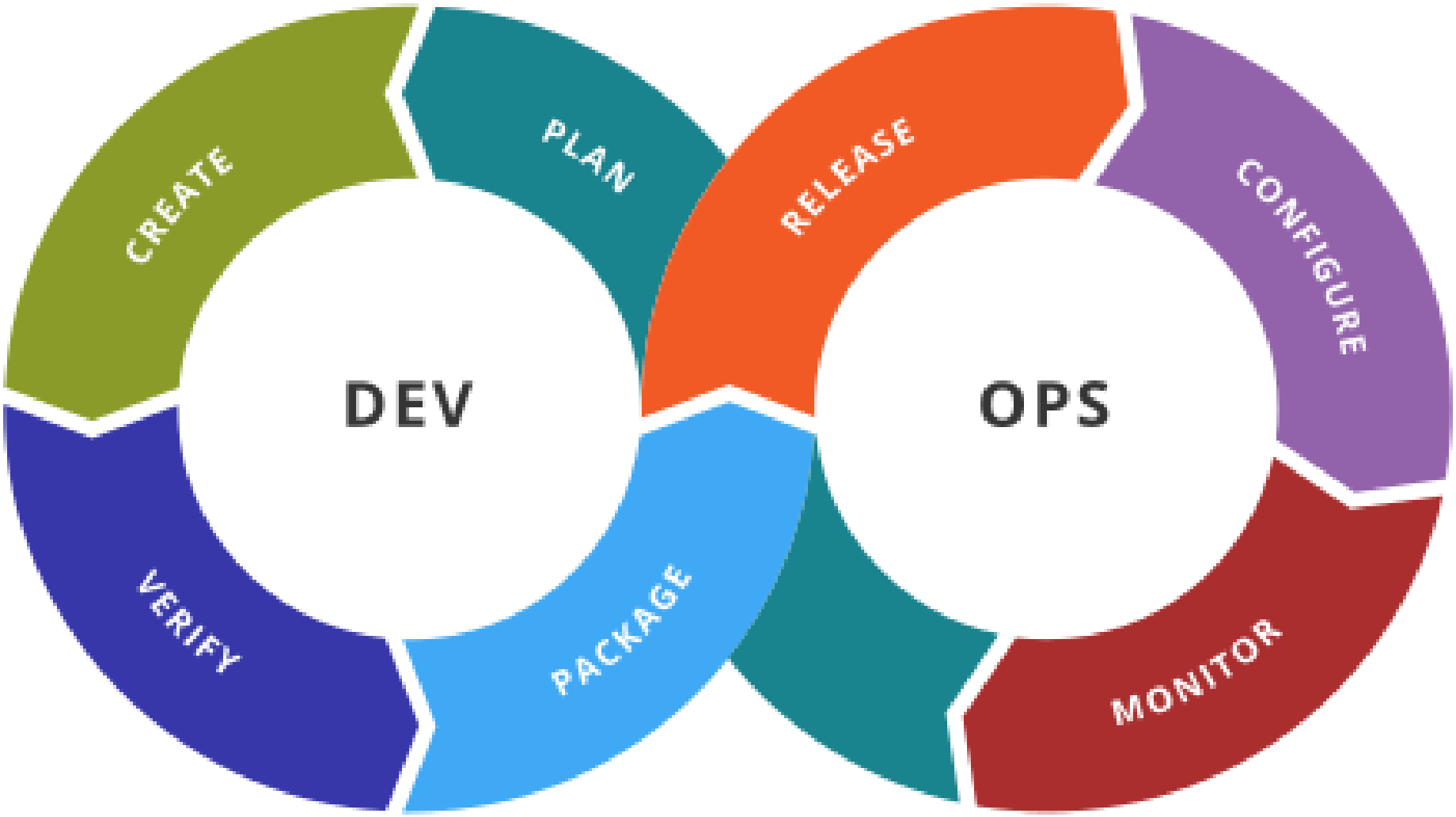
Homework 2: Observations

- Overall, high quality submissions for Homework 2.

- Feedback on schedules:
  - Should contain:
    - Description
    - Who it was assigned to
    - Estimated size from before work begins
    - Actual time took when work completed
    - Who actually completed the work (it's OK to reassign, just track this!)
  - Difficult to capture this in prose, use spreadsheet, Trello, GitHub projects, etc.
- Reflections
  - Discussing *why* things happened is better than listing *what* happened (this is in the schedule!)

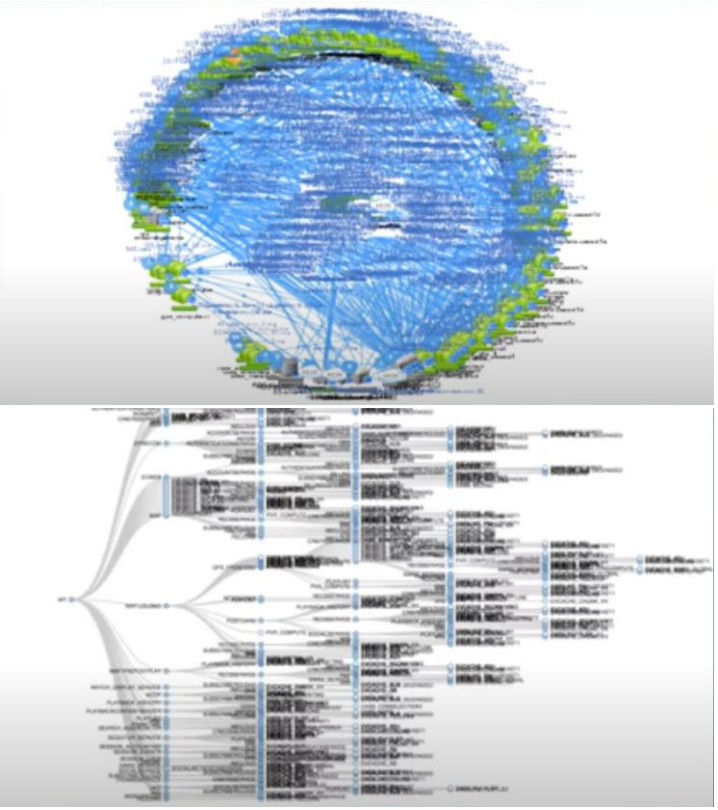
Learning Goals

- Identify the benefits to adopting a DevOps mentality within your organization
- Understand the stages in a typical DevOps pipeline
- Identify the benefits of each stage of the pipeline and how they relate to improving both velocity and code quality


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

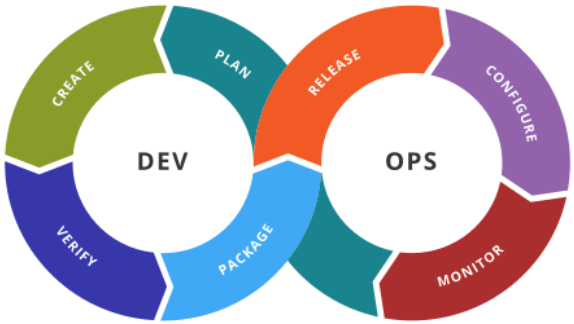
Activity

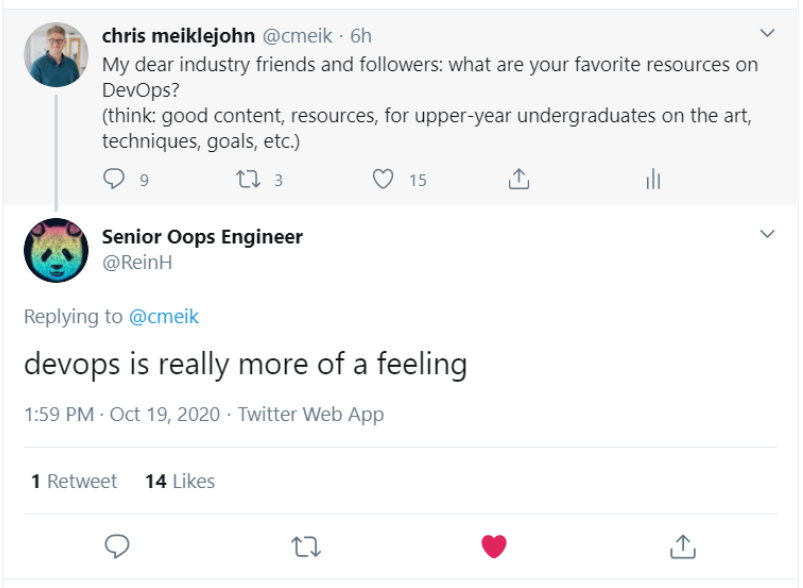
What were some of the challenges of running a microservice architecture of this scale?

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

Brainstorm: Microservices

What is DevOps?





Bringing together two traditionally separate groups within software organizations

- **Development**, typically *measured on features completed*, code shipped
- **Operations**, typically *measured through stability, reliability, availability*

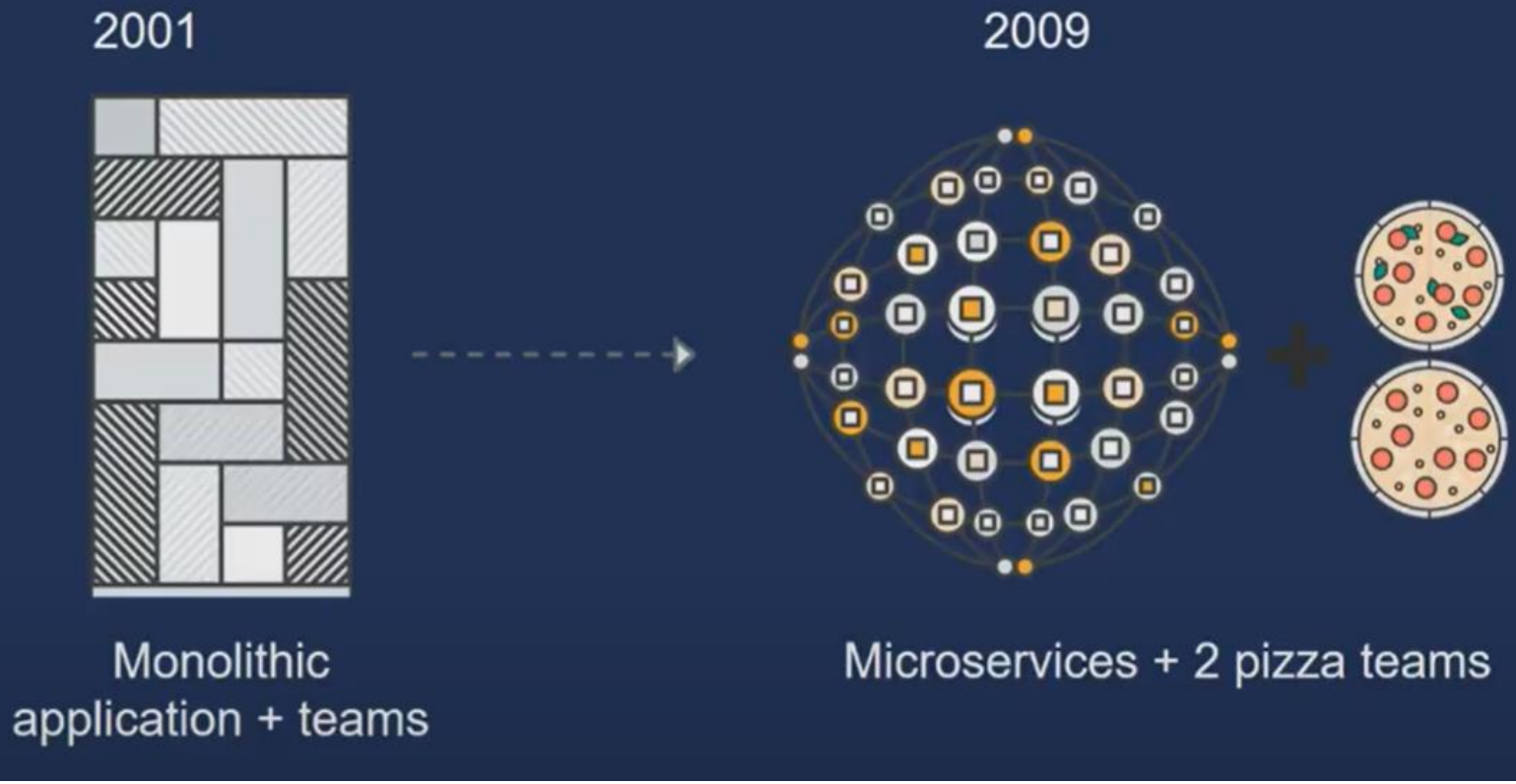
Benefits:

- *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

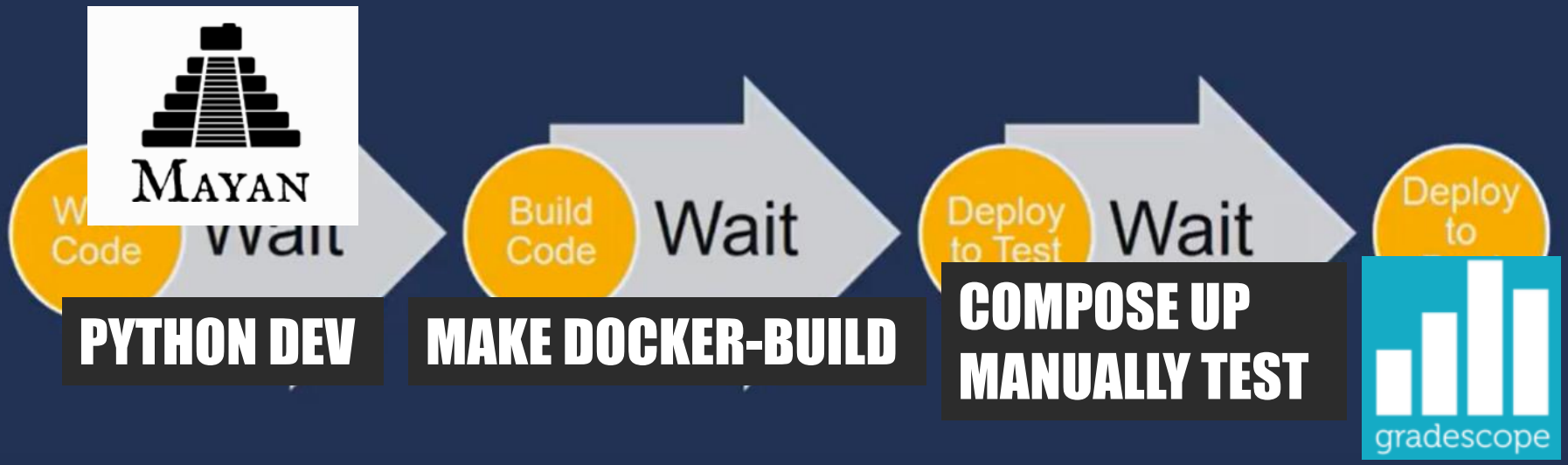
Amazon

Development transformation at Amazon: 2001-2009

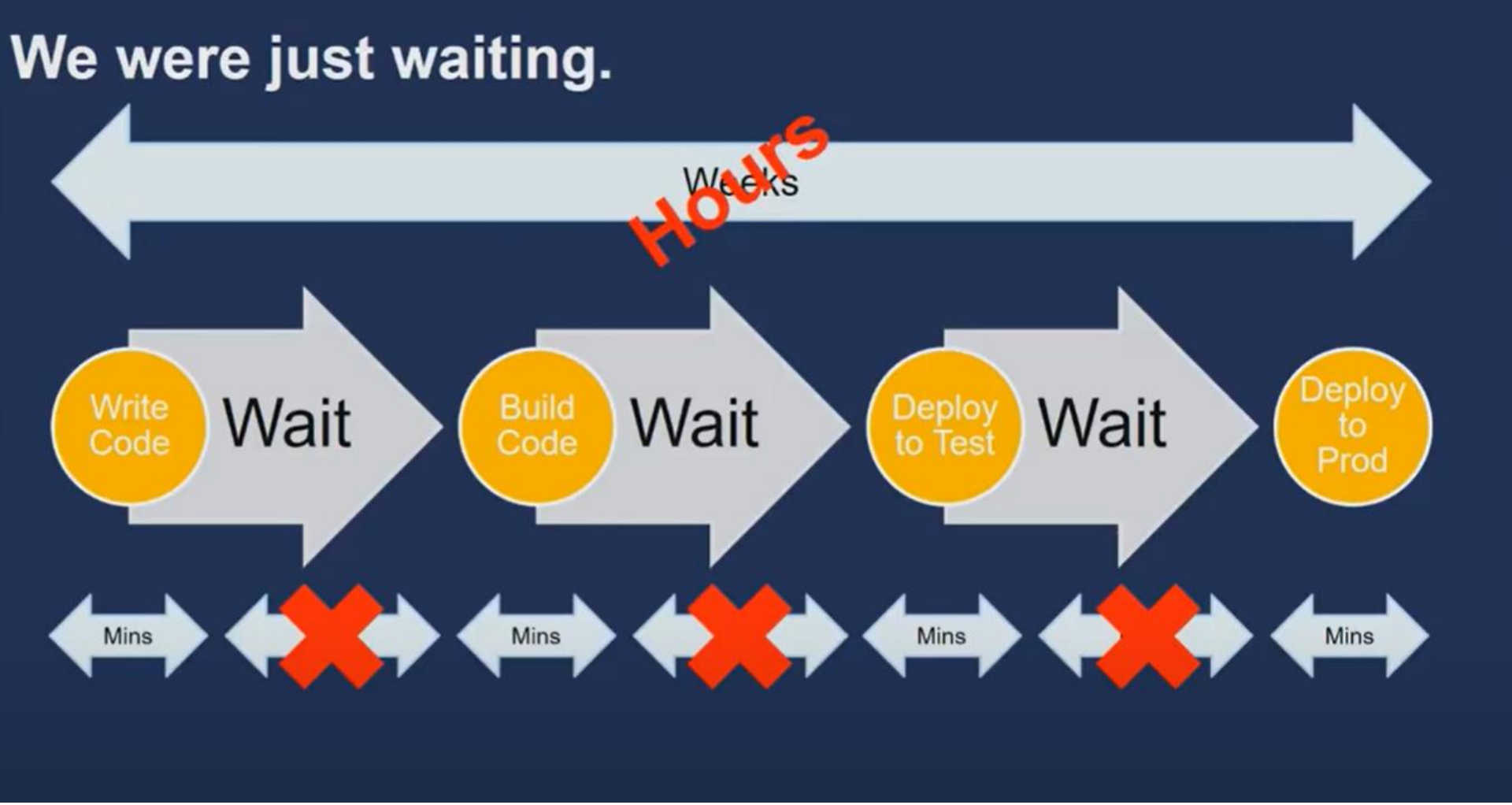


reference: https://www.youtube.com/watch?v=mBU3AJ3j1rg

We were just waiting.



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



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
lower change failure rate

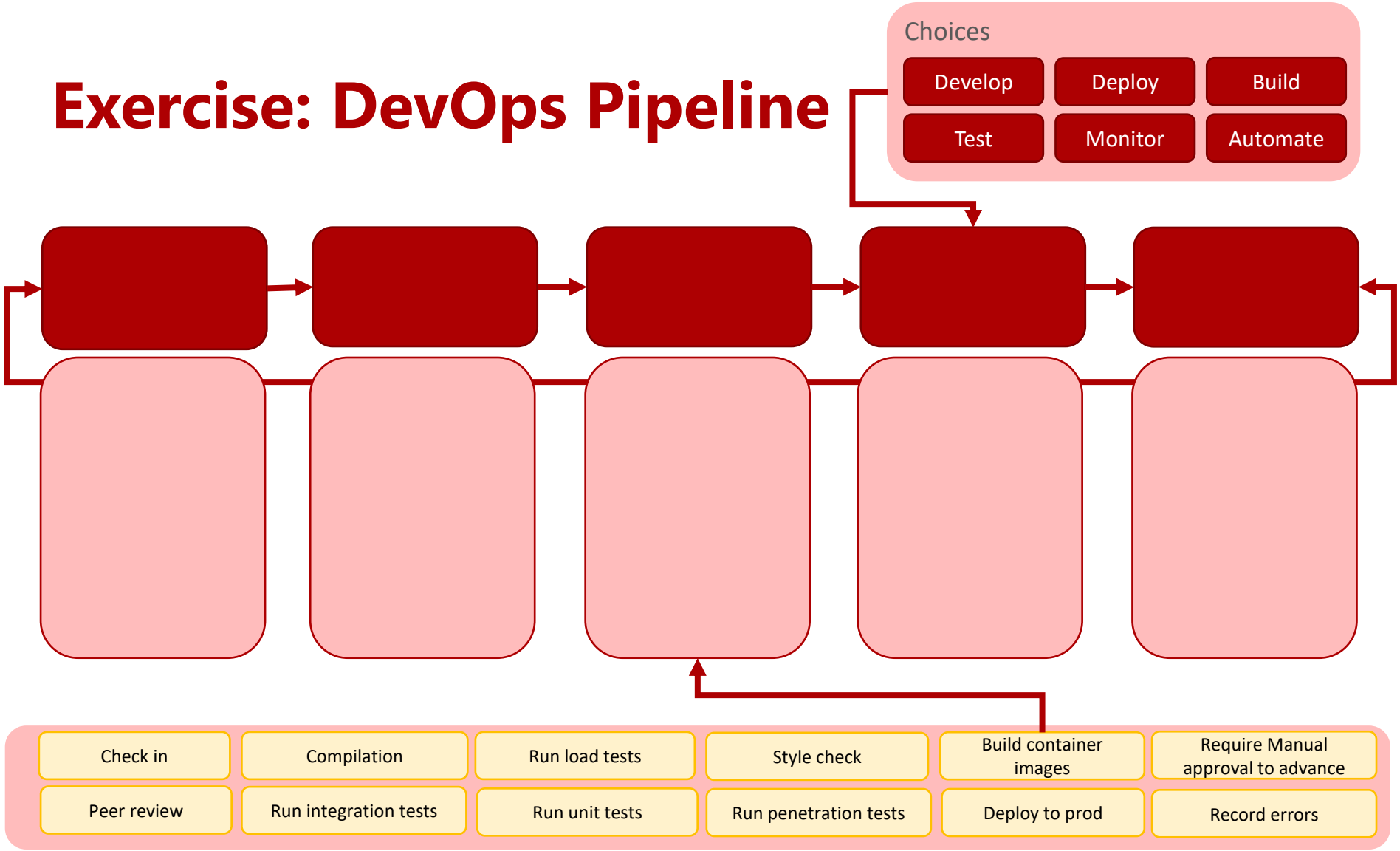
440x
faster from commit to deploy

46x
more frequent deployments

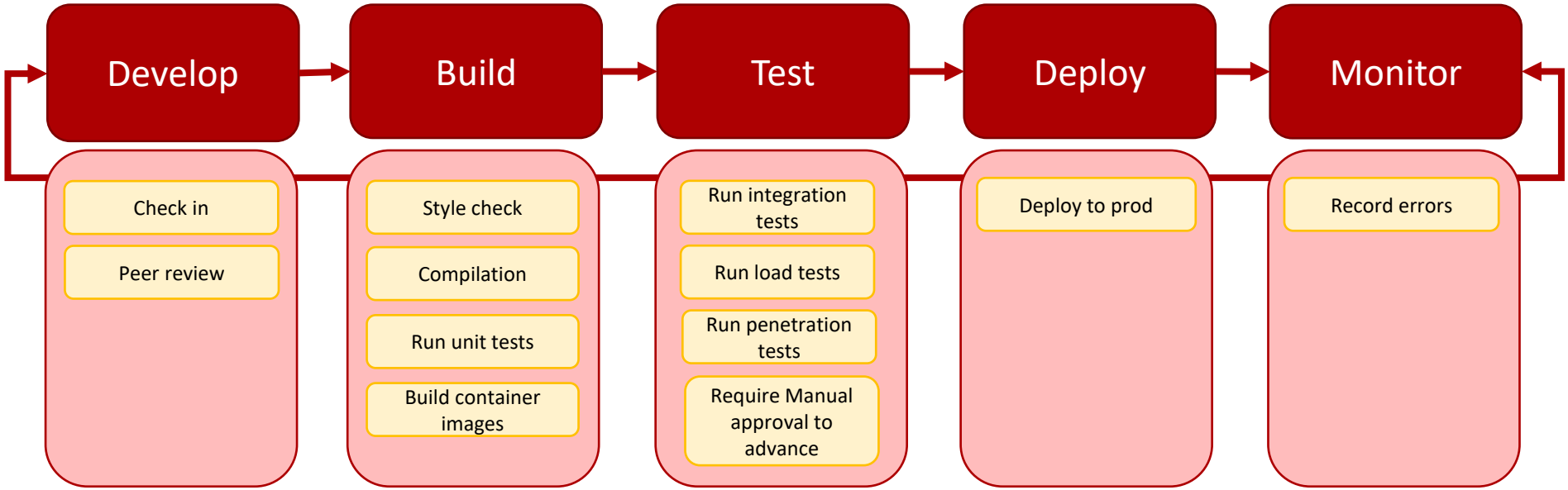
44%
more time spent on new features and code

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

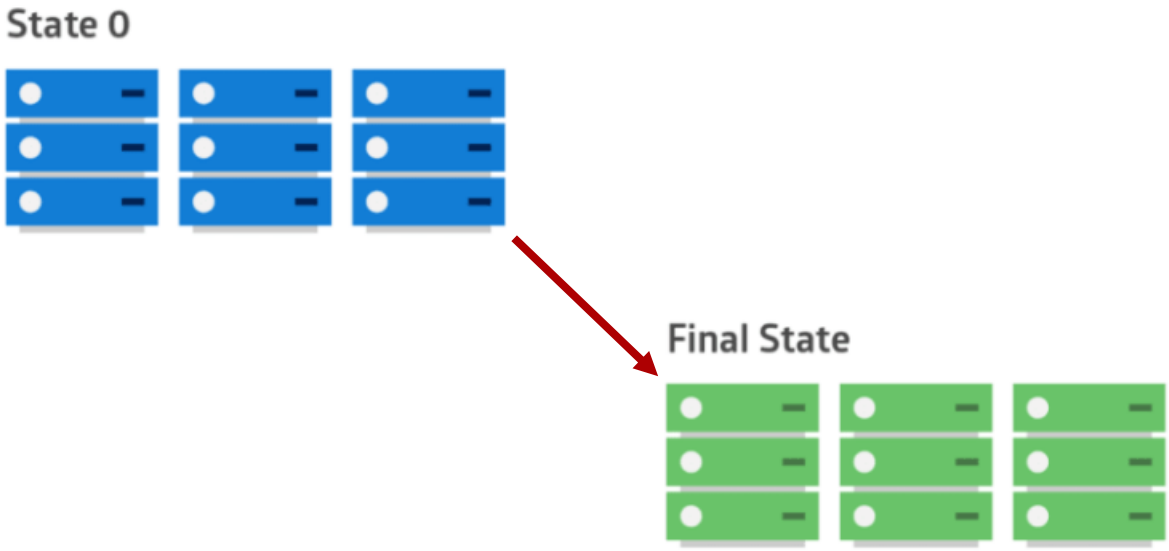
- Commits flow out to rings, de-flight if issue
- For example:
  - Ring 0 => Team
  - Ring 1 => Dogfood
  - Ring 2 => Beta
  - Ring 3 => Many
  - Ring 4 => All
- Windows 10 Insiders Program
  - Dev Channel (weekly builds of Windows 10)
  - Beta Channel (dev + validated updates by Microsoft)
  - Release 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-3a3

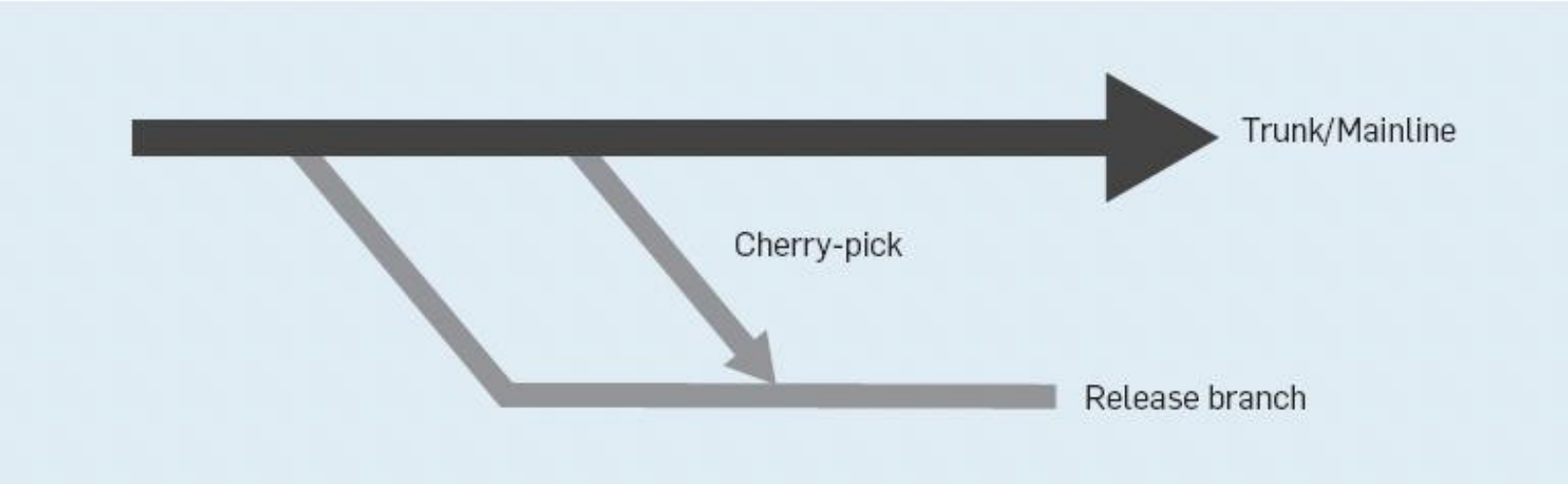
Fast to Deploy, Slow to Release

- Chuck Rossi at Facebook: *"Get your s*** in, fix it in production"*

Dark Launches at Instagram

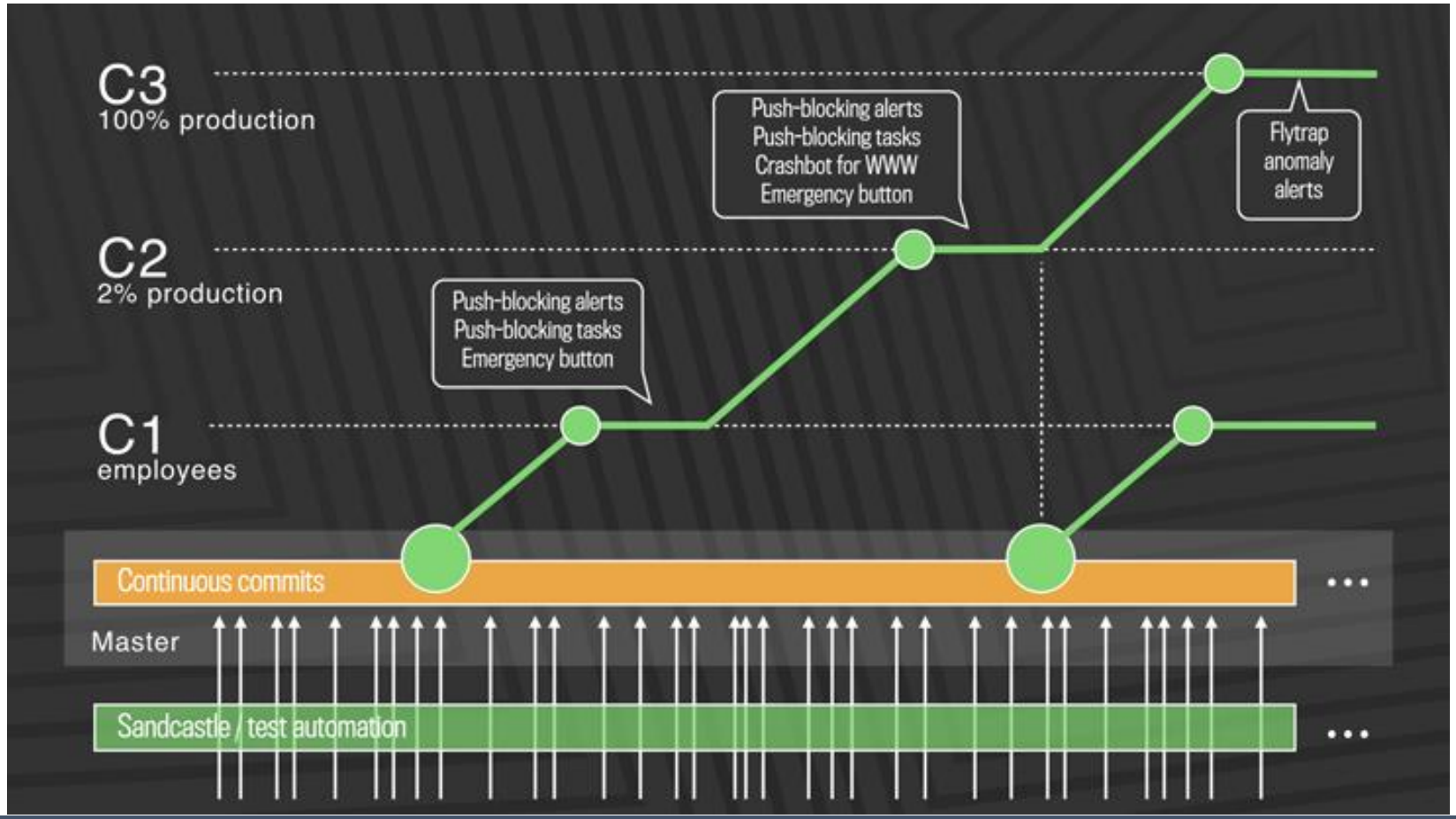
- **Early**: Integrate as soon as possible. Find bugs early. Code can run in production about 6 months before being publicly announced.
- **Often**: Reduce friction. Try things out. See what works. Push small changes just to gather metrics, feasibility testing. Large changes just slow down the team. 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.

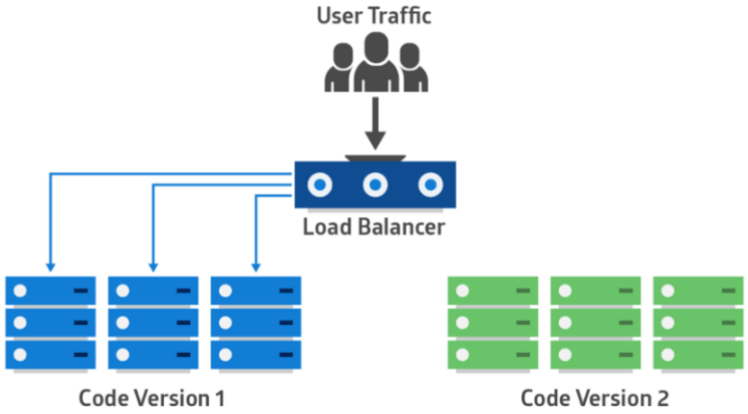
Facebook quasi-continuous release

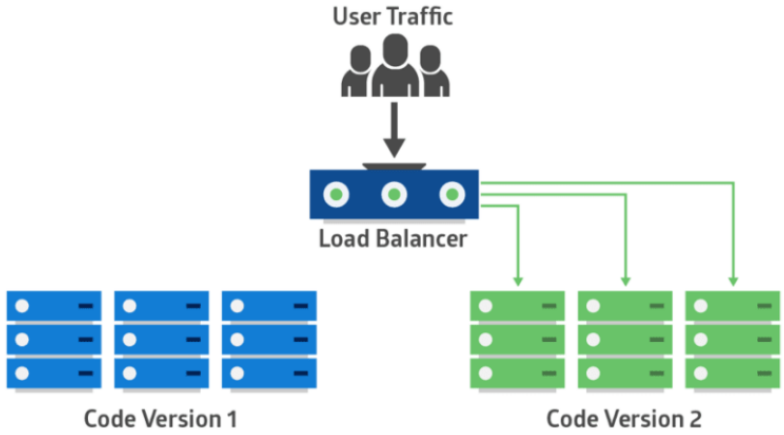


Rolling deployments

reference: https://dev.to/mostlyjason/intro-to-deployment-strategies-blue-green-canary-and-more-3a3

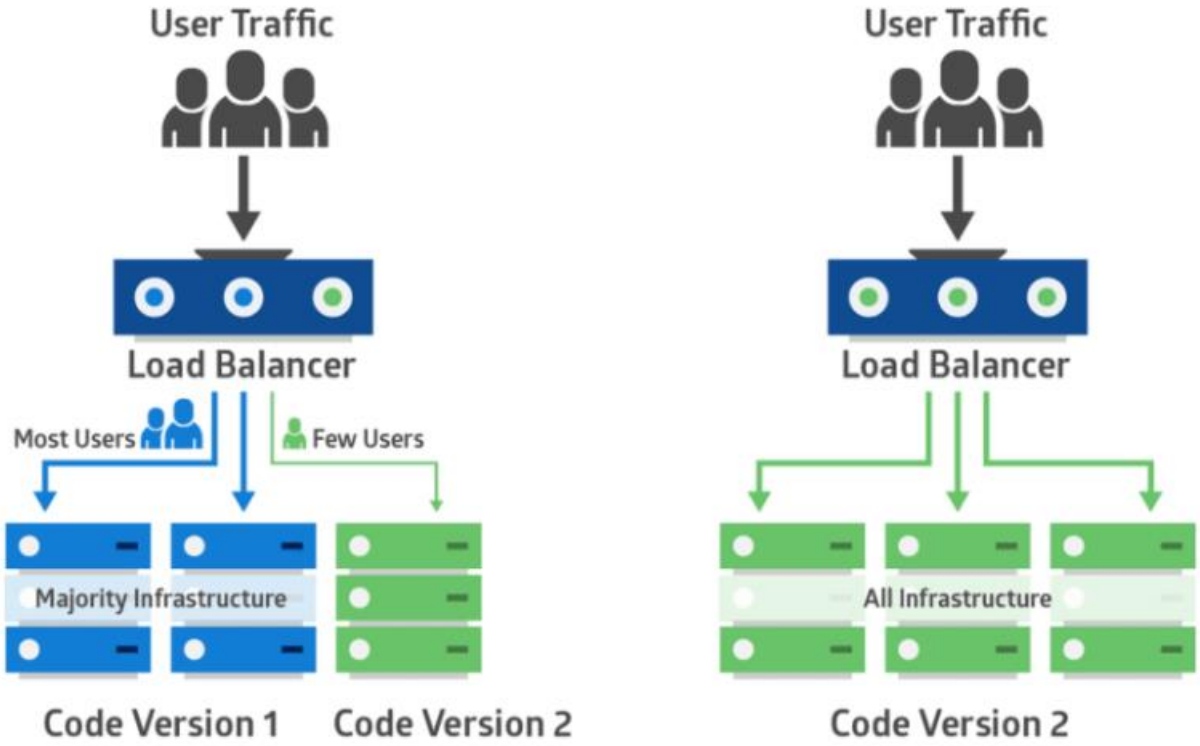
Red/Black (Blue/Green) deployments





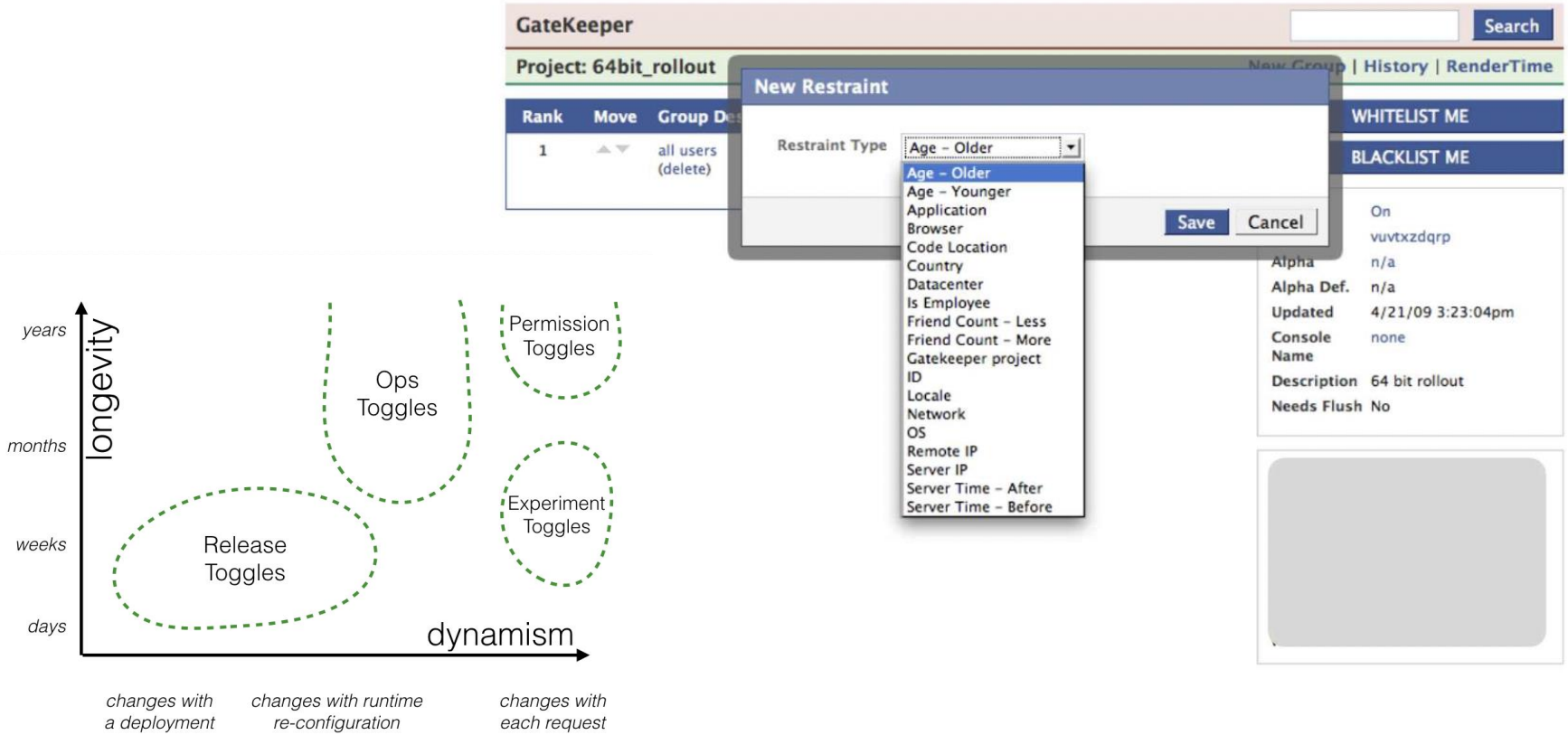
reference: https://dev.to/mostlyjason/intro-to-deployment-strategies-blue-green-canary-and-more-3a3

Canary deployments



reference: https://dev.to/mostlyjason/intro-to-deployment-strategies-blue-green-canary-and-more-3a3

Feature flags



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."

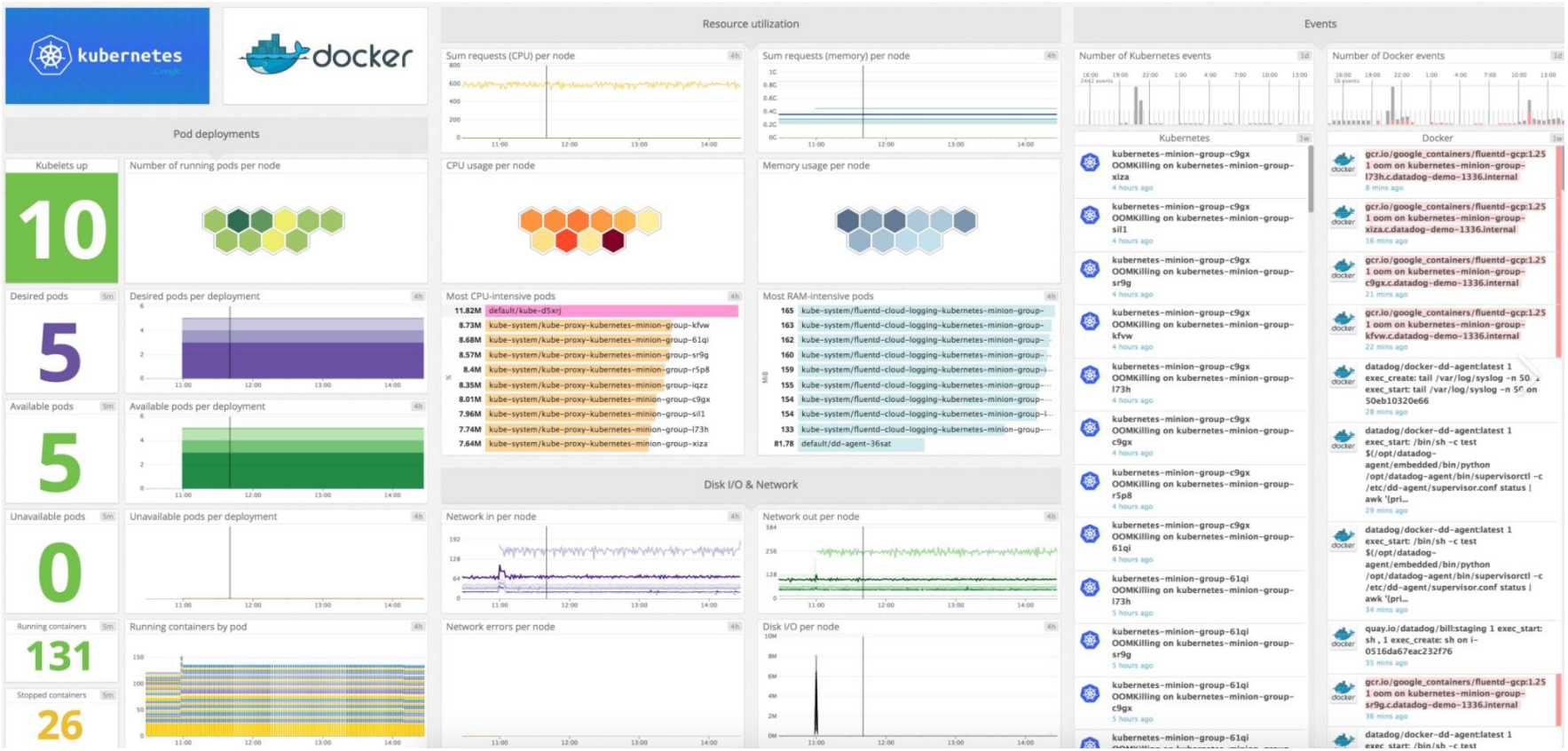
The New Stack, "What is observability?" 28 Feb 2020
https://thenewstack.io/what-is-observability/

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?

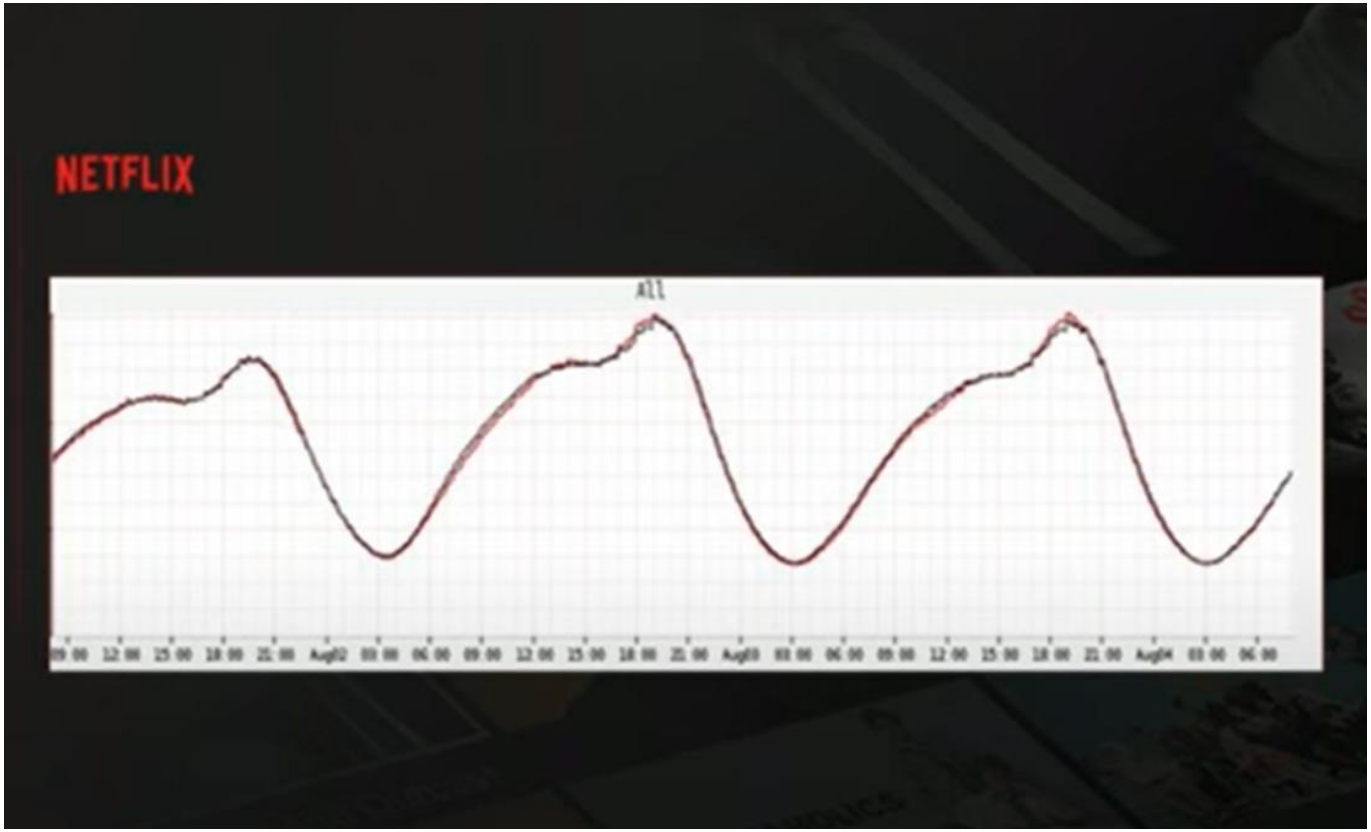
reference: https://www.youtube.com/watch?v=mBU3AJ3j1rg

Observability: Dashboard Example



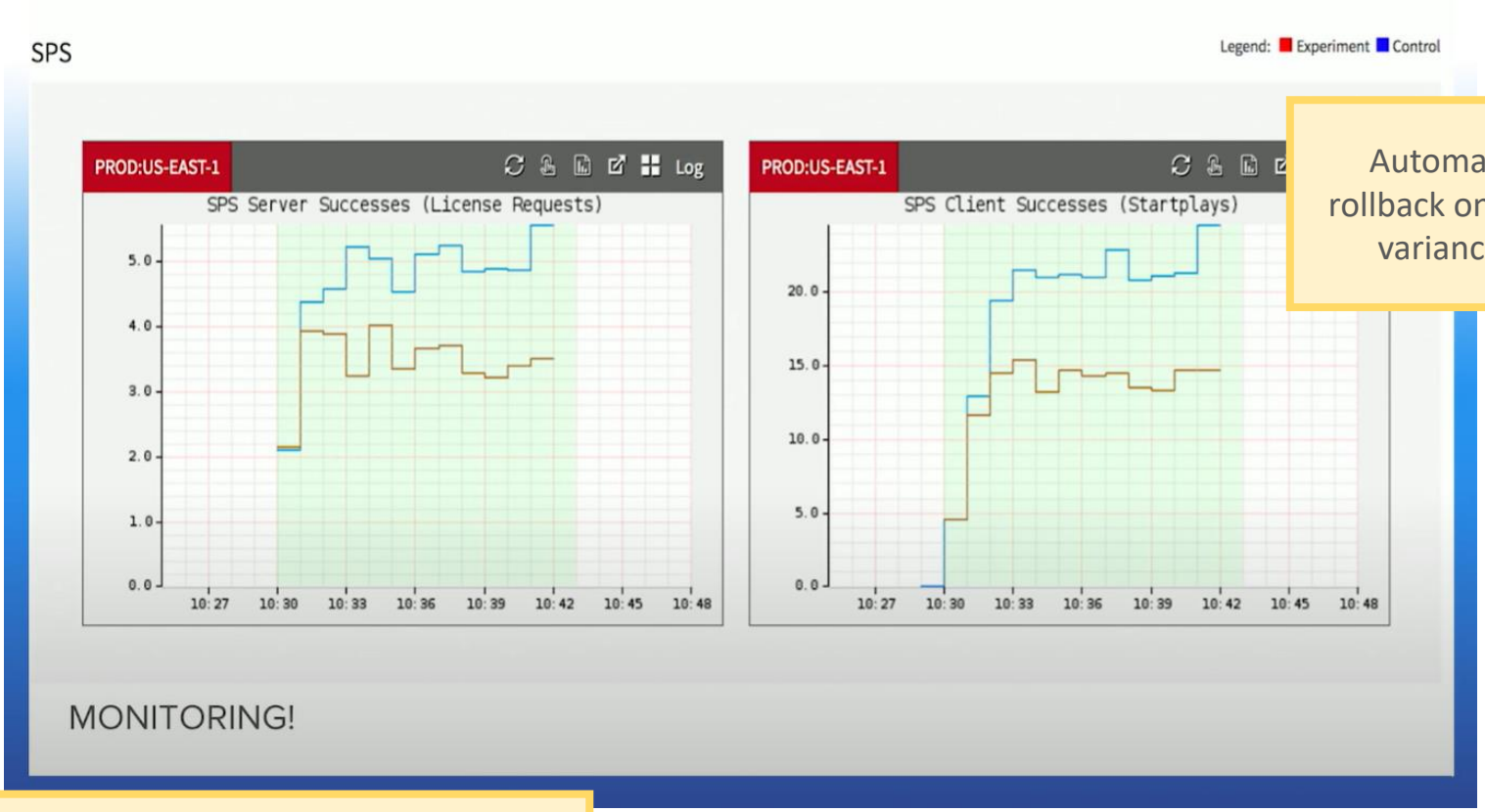
reference: https://datadog-prod.imgix.net/img/blog/monitoring-kubernetes-with-datadog/kubernetes-dashboard.png?fit=max

Observability: Defining "Normal"



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

Observability: When things aren't "Normal"

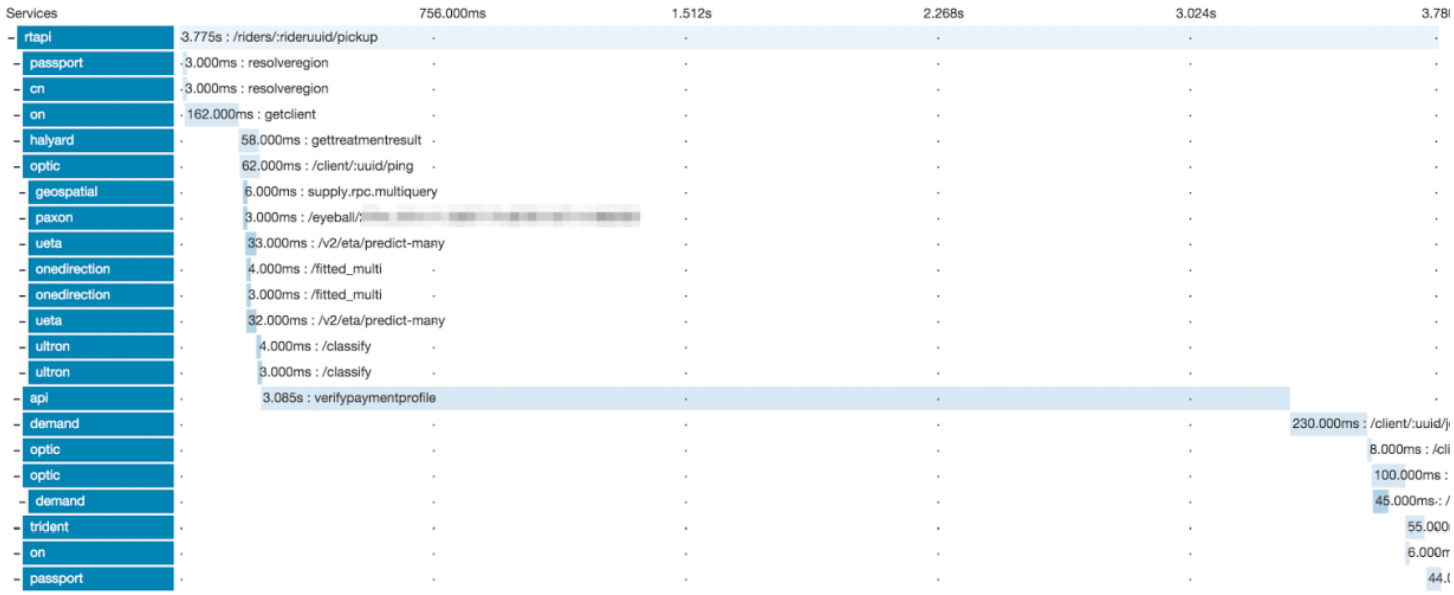


Automatic rollback on high variance!

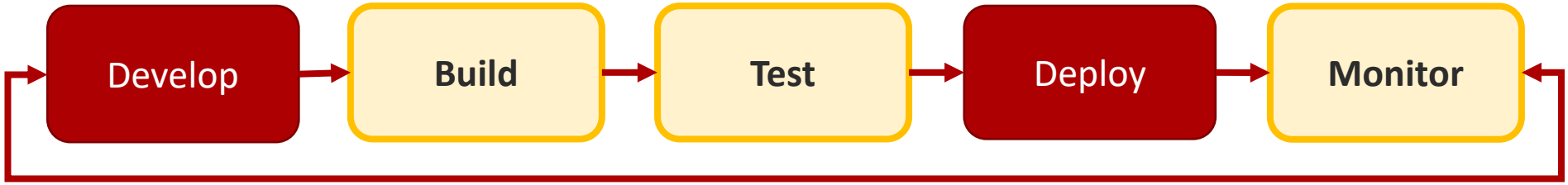
This is starting to sound awfully like a *quality attribute….*

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

Observability: Distributed Tracing



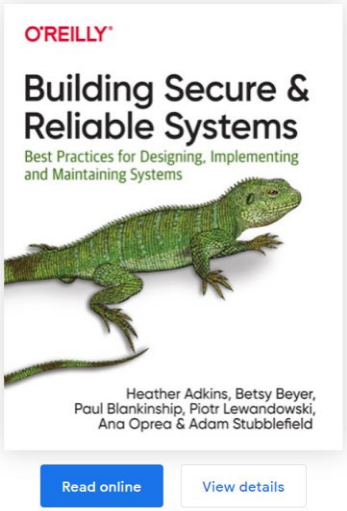
Homework 2A: Testing Plan

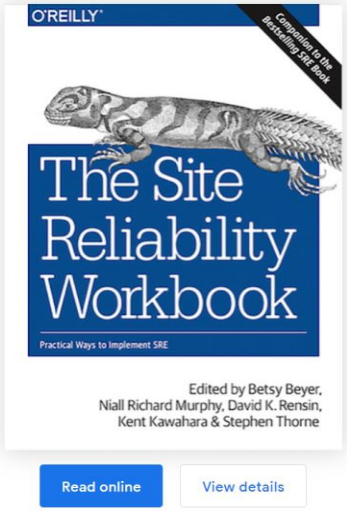


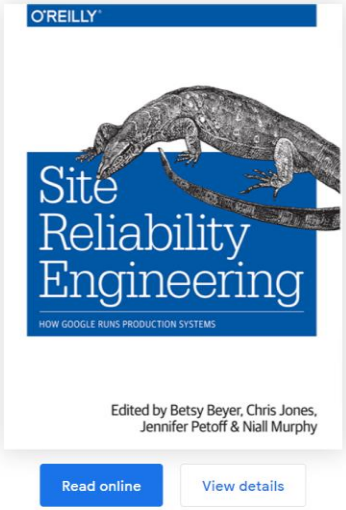
In your homework teams, come up with
one concrete task for the build, test, and monitor stages
to verify that your feature works as designed once deployed to production.

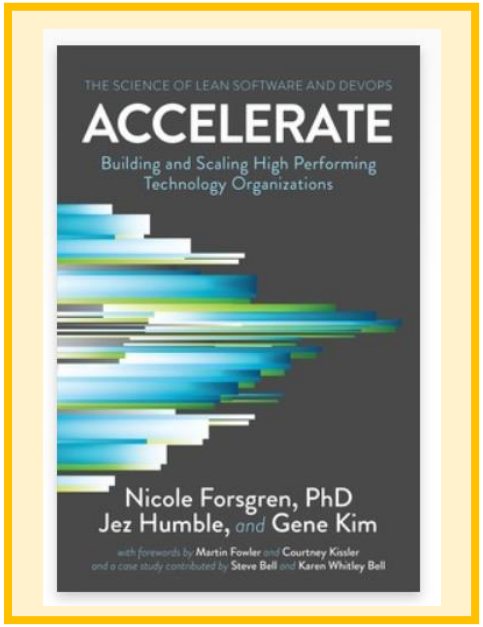
DevOps: More Resources

SRE Books









All available online from CMU Libraries!

Next Week: QA

Questions?