CS 4530 & CS 5500 Software Engineering

Lecture 10.4: Continuous Delivery

Jonathan Bell, John Boyland, Mitch Wand
Khoury College of Computer Sciences

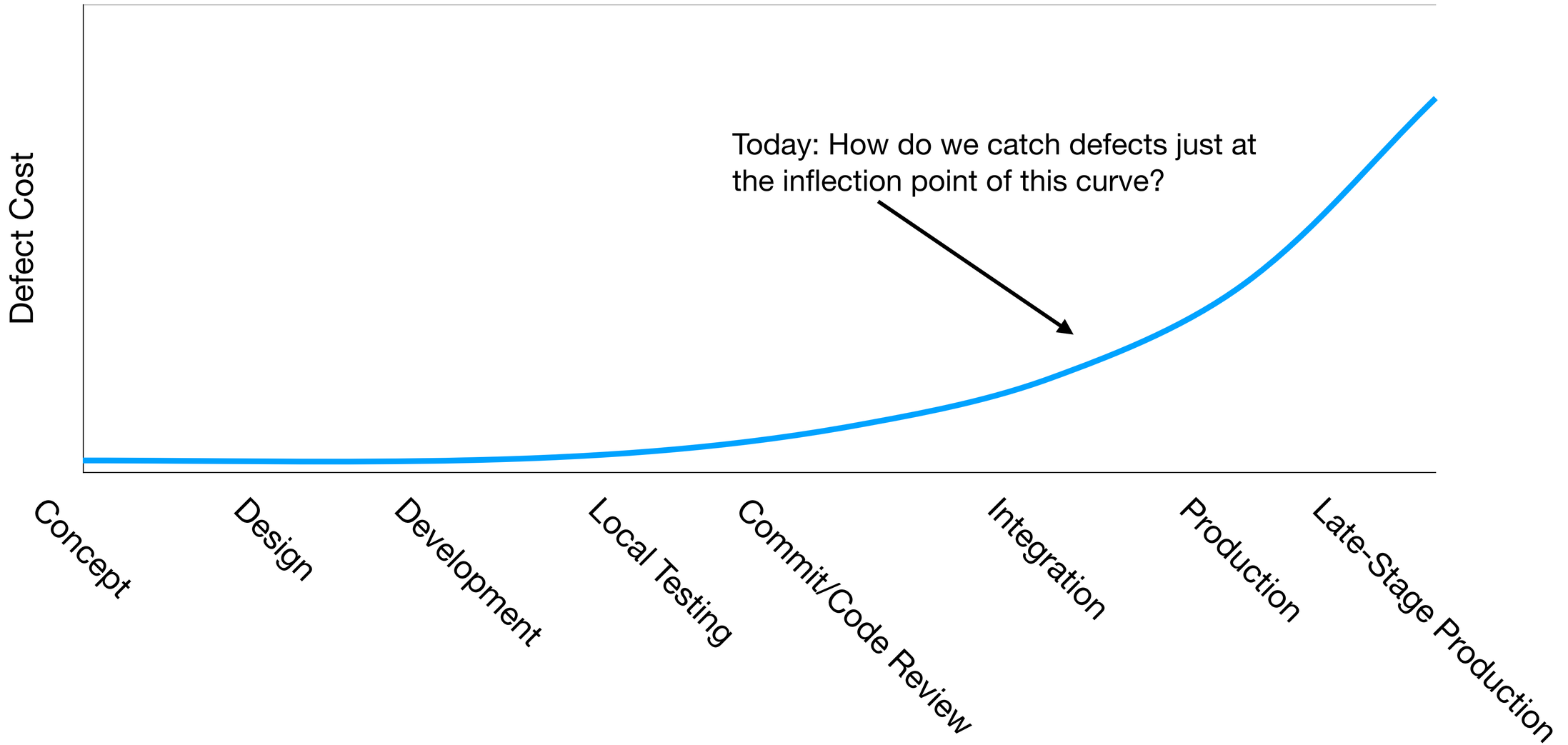
Learning Objectives for this Lesson

By the end of this lesson, you should be able to…

- Describe how continuous delivery helps to catch errors sooner in a product's lifecycle
- Describe the distinction between a DevOps and "traditional" developer/operator mentality
- Describe strategies for performing quality-assurance on software as and after it is delivered

Cost to Fix a Defect Over Time

Rough estimate



Deploying New Code

The best that we can hope for?

“If stuff blows up it affects a very small percentage of people”

Instagram cofounder and CTO Mike Krieger

https://www.fastcompany.com/3047642/do-the-simple-thing-first-the-engineering-behind-instagram

Continuous Delivery

“Faster is safer”: Key values of continuous delivery

- Release frequently, in small batches
- Maintain key performance indicators to evaluate the impact of updates
- Phase roll-outs
- Evaluate business impact of new features

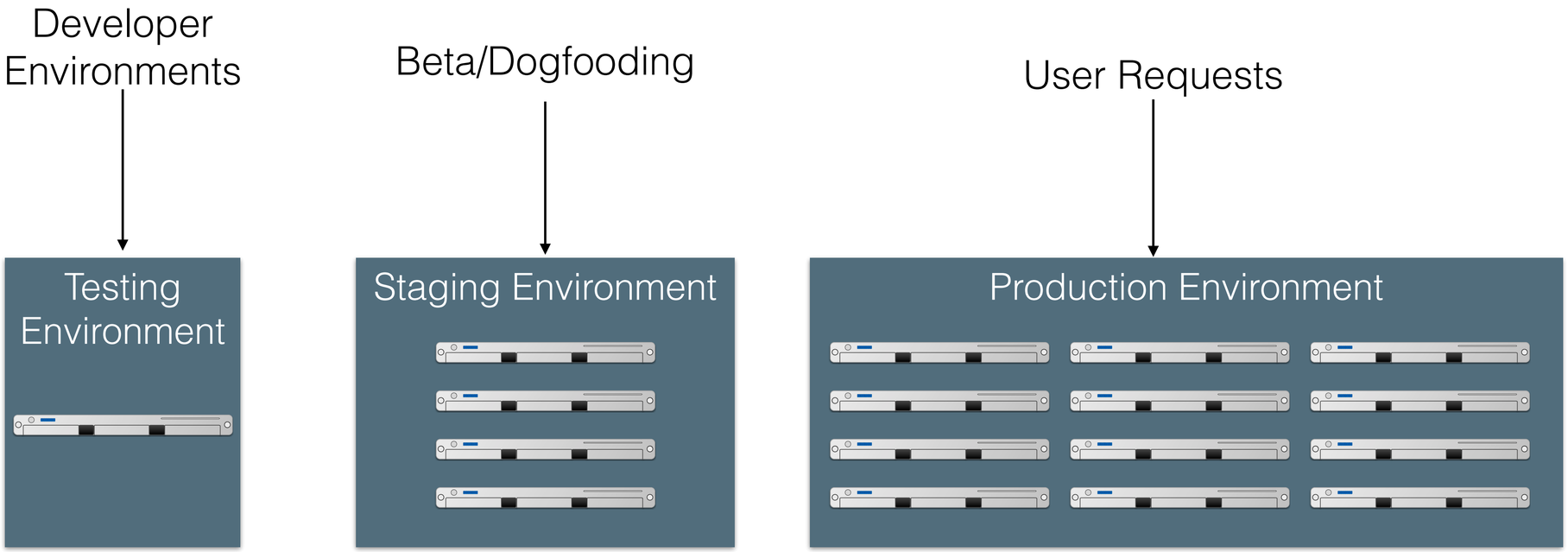
Staging Environments

Enabling Continuous Delivery

- As software gets more complex with more dependencies, it's impossible to simulate the whole thing when testing
- Idea: Deploy to a complete production-like environment, but don't have everyone use it
  - Examples:
    - “Eat your own dogfood”
    - Beta/Alpha testers
- Lower risk if a problem occurs in staging than in production

Test-Stage-Production

Continuous Delivery in Action

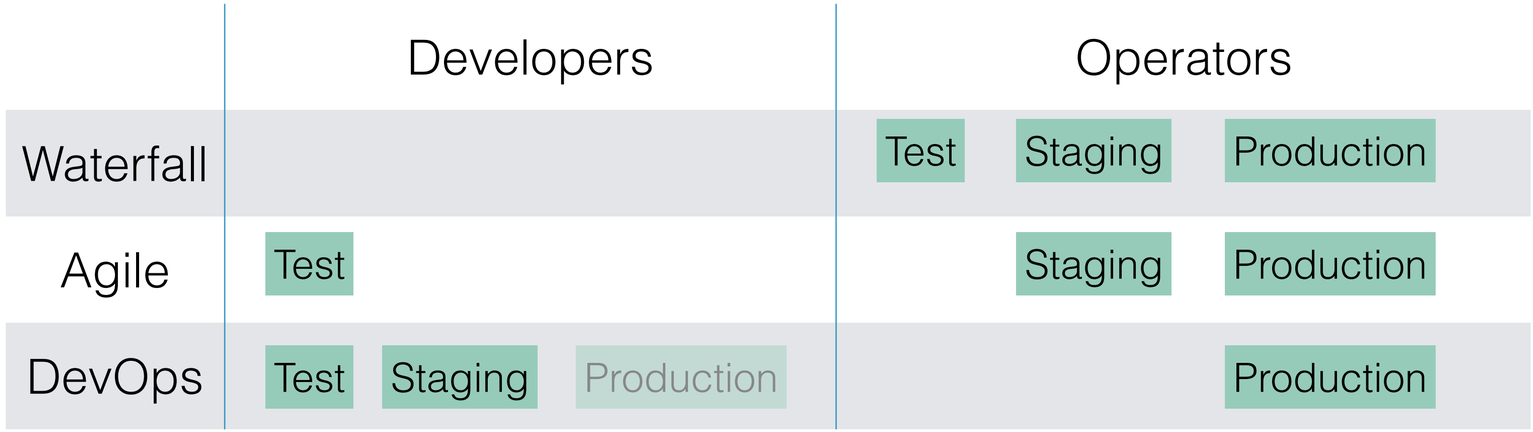


Q/A takes place in each stage (including production!)

Operations Responsibility

DevOps in a slide

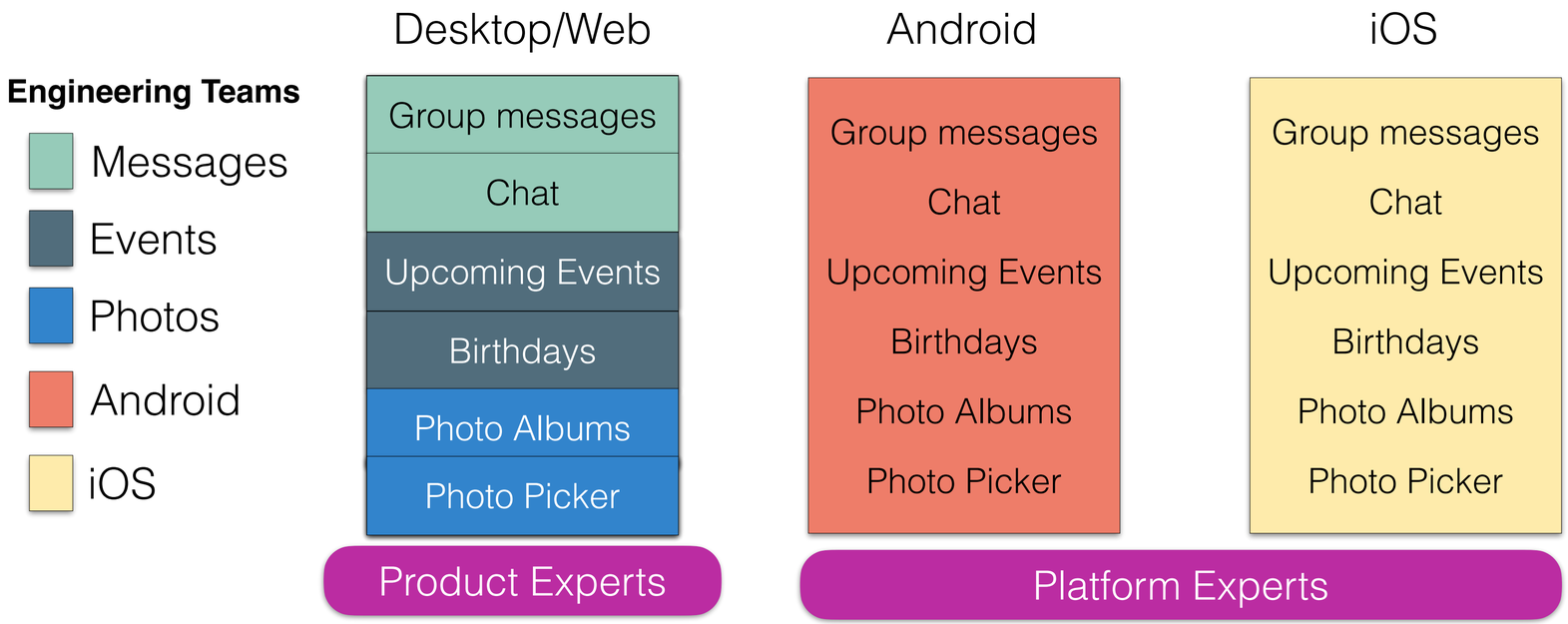
- Once we **deploy**, someone has to monitor software, make sure it's running OK, no bugs, etc
- Assume 3 environments:
  - Test, Staging, Production
- Whose job is it?



DevOps Values

One team owns changes "from cradle to grave"

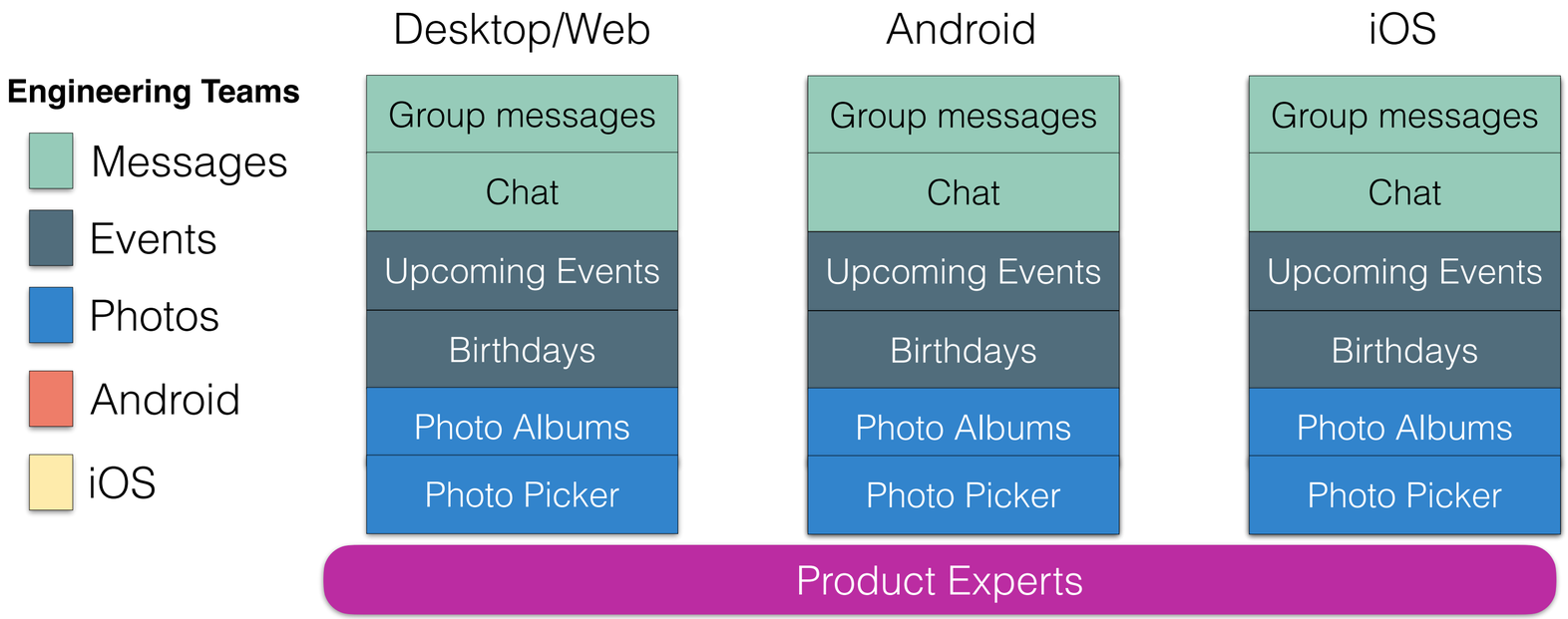
- You are the support person for your changes, regardless of platform
- Example: Facebook mobile teams (non-DevOps)



DevOps Values

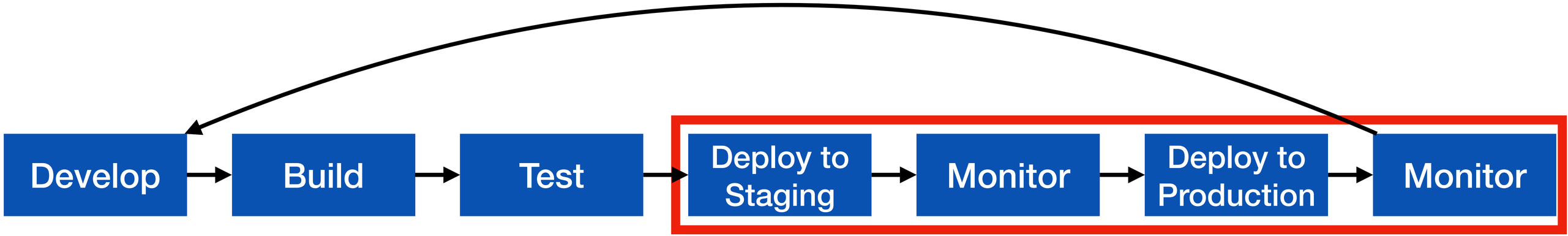
One team owns changes "from cradle to grave"

- You are the support person for your changes, regardless of platform
- Example: Facebook mobile teams (DevOps)



Deployment Pipeline

With staging environment



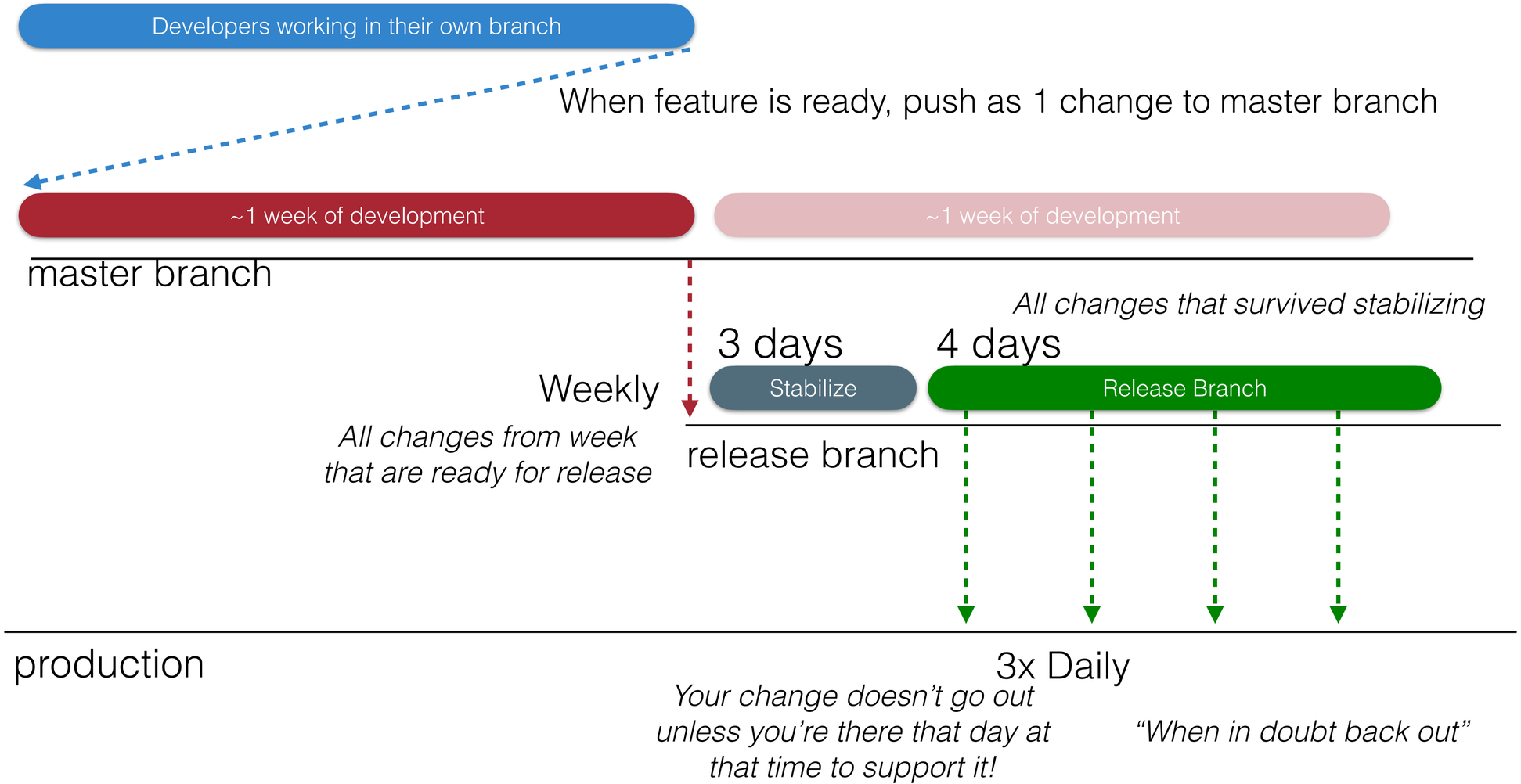
Release Pipelines

How quickly is my change deployed?

- Even if you are deploying every day, you still have some latency
- A new feature I develop today won't be released today
- But, a new feature I develop today can begin the **release pipeline** today (minimizes risk)
- **Release Engineer**: gatekeeper who decides when something is ready to go out, oversees the actual deployment process

Deployment Example: Facebook.com

Pre-2016



Deployment Example: Facebook.com

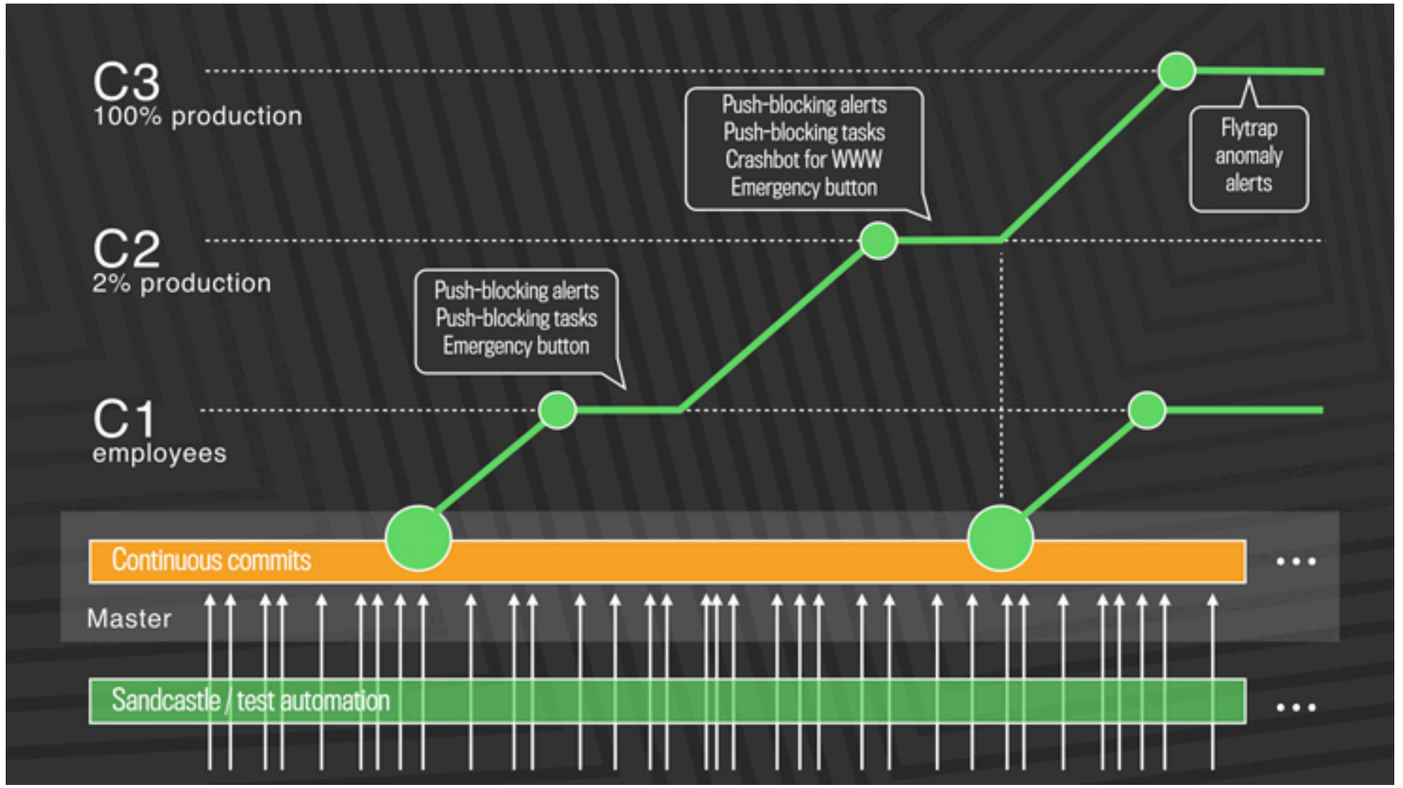
Chuck Rossi, Director Software Infrastructure & Release Engineering @ Facebook

“Our main goal was to make sure that the new system made people’s experience better — or at the very least, didn’t make it worse. After almost exactly a year of planning and development, over the course of three days in April 2017 **we enabled 100 percent of our production web servers to run code deployed directly from master**.”

“Rapid release at massive scale” https://engineering.fb.com/2017/08/31/web/rapid-release-at-massive-scale/

Deployment Example: Facebook.com

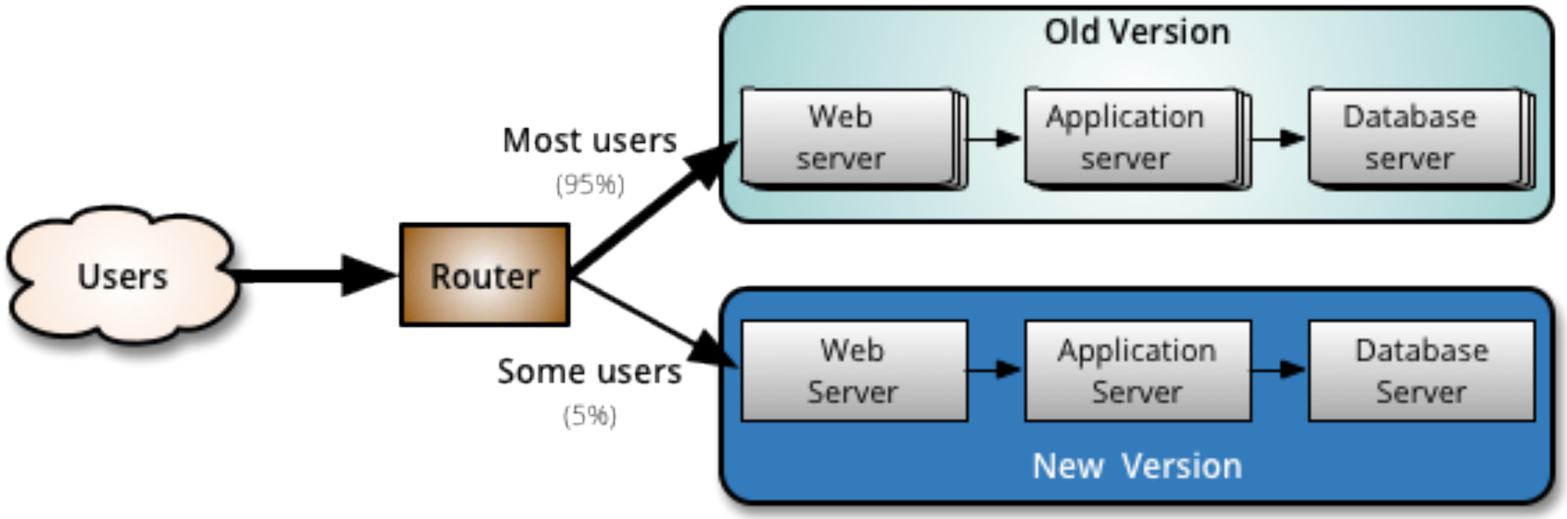
Post-2016: Truly continuous releases from master branch



https://engineering.fb.com/2017/08/31/web/rapid-release-at-massive-scale/

A/B Deployments with Canaries

Mitigating risk in continuous delivery



Monitor both:
But minimize impact of problems in new version

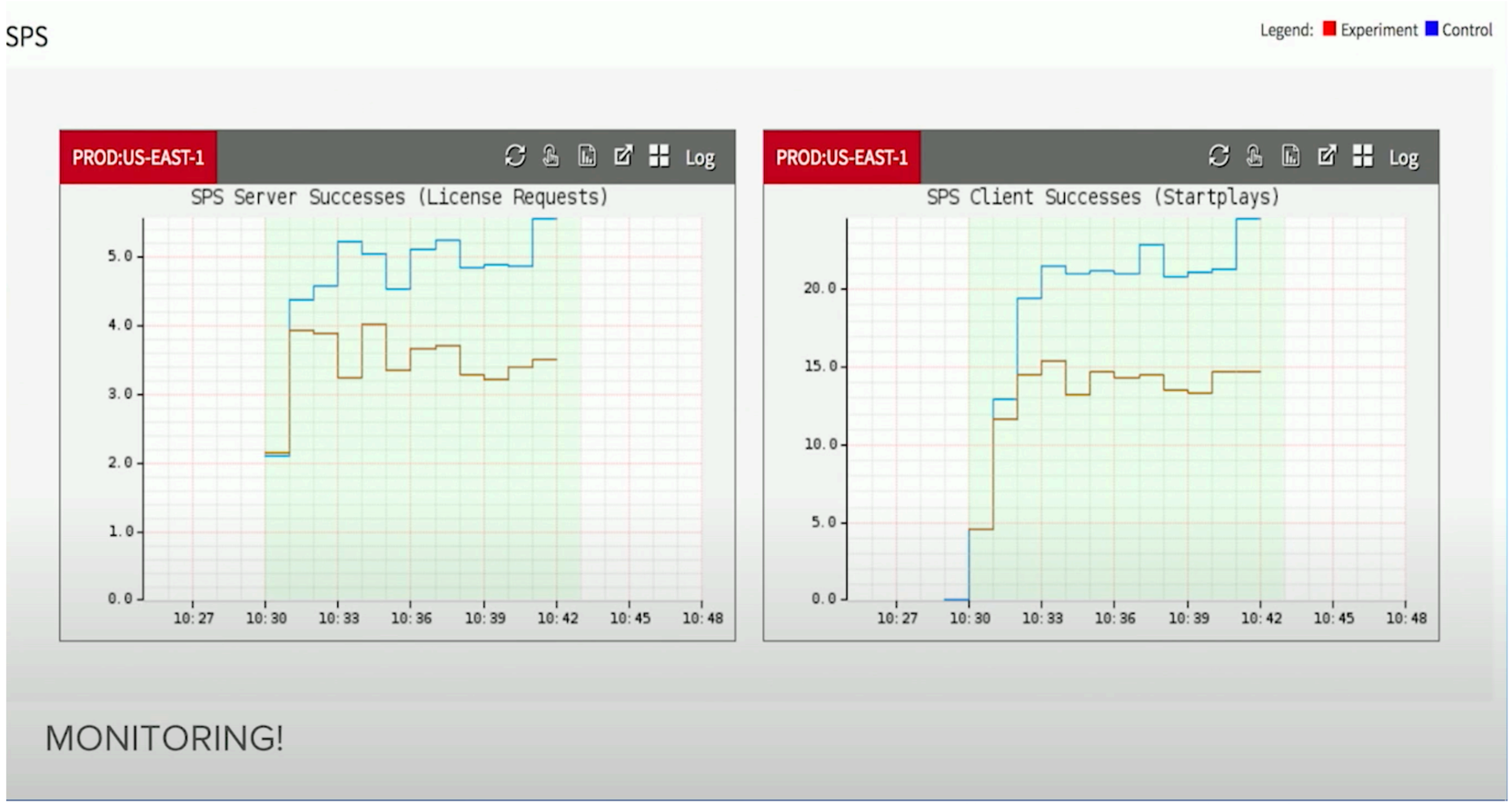
Monitoring

The last step in continuous deployment: track metrics

- Hardware
  - Voltages, temperatures, fan speeds, component health
- OS
  - Memory usage, swap usage, disk space, CPU load
- Middleware
  - Memory, thread/db connection pools, connections, response time
- Applications
  - Business transactions, conversion rate, status of 3rd party components

Monitoring

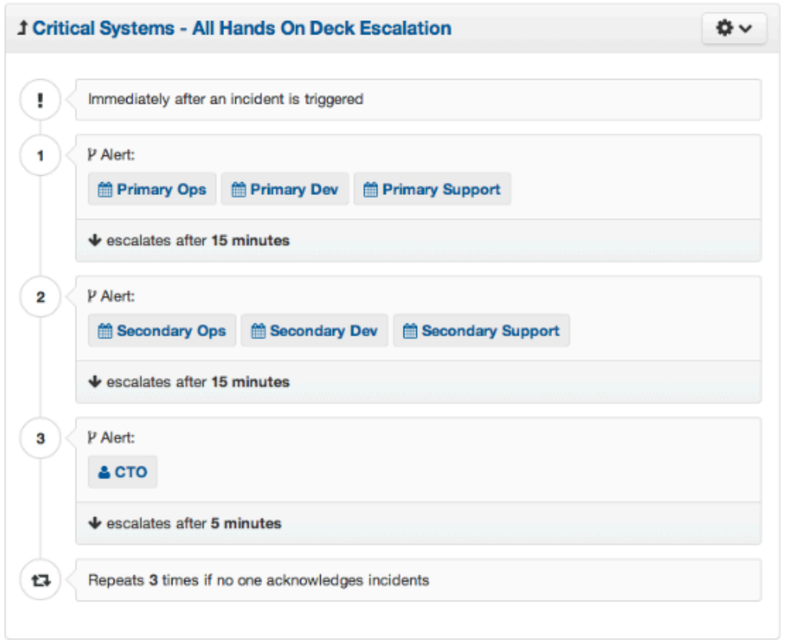
Automatically detecting irregular behavior at Netflix



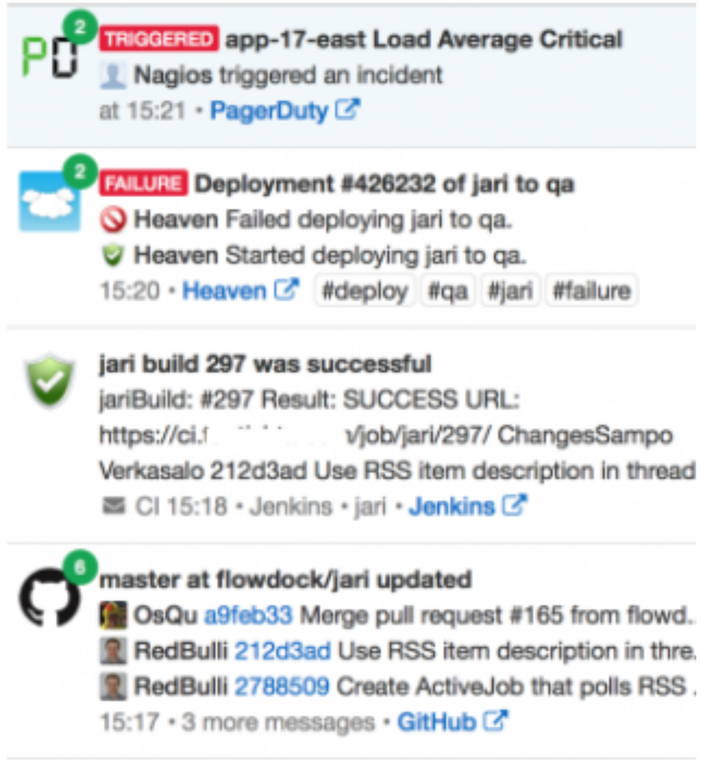
https://www.youtube.com/watch?v=qyzymLlj9ag

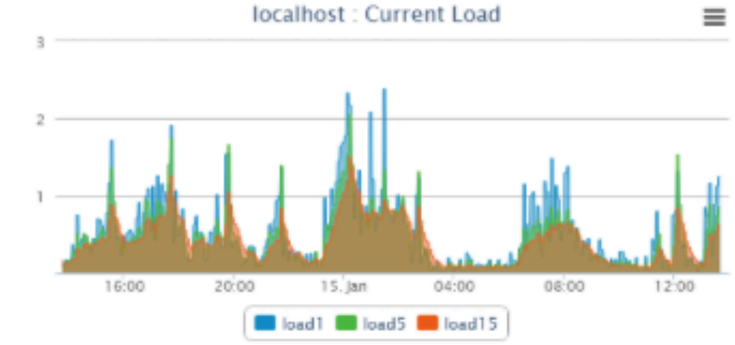
When things go wrong

- Automated monitoring systems can notify “on-call” staff of a problem
- Triage & escalation



Monitoring Dashboards





Host Status Summary

Up	Down	Unreachable	Pending
33	4	0	0
Unhandled	**Problems**		**All**
4	4		37

Last Updated: 2014-01-15 13:45:04

Service Status Summary

Ok	Warning	Unknown	Critical	Pending
279	5	1	47	1
Unhandled	**Problems**		**All**	
51	53		333	

Last Updated: 2014-01-15 13:45:04

Disk Usage

Host	Service	% Utilization	Details
localhost	Root Partition	75.0%	DISK OK - free space: / 1516 MB (21% inode=83%):
ScottsServer	Drive D: Disk Usage	69.4%	D:\ - total: 232.88 Gb - used: 161.71 Gb (69%) - free 71.17 Gb (31%)
ScottsServer	Drive C: Disk Usage	29.1%	C:\ - total: 452.96 Gb - used: 131.70 Gb (29%) - free 321.26 Gb (71%)
vs1.nagios.com	/ Disk Usage	27.6%	DISK OK - free space: / 56392 MB (71% inode=97%):
exchange.nagios.org	/ Disk Usage	6.6%	DISK OK - free space: / 73482 MB (93% inode=98%):

Last Updated: 2014-01-15 13:44:06

Status Grid

Hosts	Services
192.168.5.200	
192.168.5.33	
192.168.5.41	
abc.com	
APC-UPS-27	
arcade.nagios.local	
Business Process	
conference.nagios.lo	
exchange.nagios.org	
firewall	
gateway.nagios.local	
google.com	

This work is licensed under a Creative Commons Attribution-ShareAlike license