How the work works

Matt Machell - Senior Technical Architect, OPG Digital / MOJ
@shuckle / eclecticdreams.com

A story of change

How you work matters
as much as the tools

Steal this, but...

The best of (digital transformation) intentions

The symptoms

- Not getting value to real people as regularly as you once did
- Slow cycles of manual testing
- User stories take more than one sprint to deliver
- Complex release process
- Bugs
- The dev team is seen as a feature farm and distanced from users

TECH
DEBT

PRODUCT ROT

DELIVERY ATROPHY

Lesson: How you talk about technical work really matters

Us. Circa 2018

Are you optimised for change?

5 minutes now or **45 minutes** every time we change something

All your risks in one basket

Aim : Small change, Often

Rule #1 : We should be able to release the current working code at any time.

Why the Fear?

Are you scared it might break?

Write better test automation.

Scared data might get messed up?

Work on your rollback strategy.

Worried about database migrations being slow?

Reengineer your database change approach

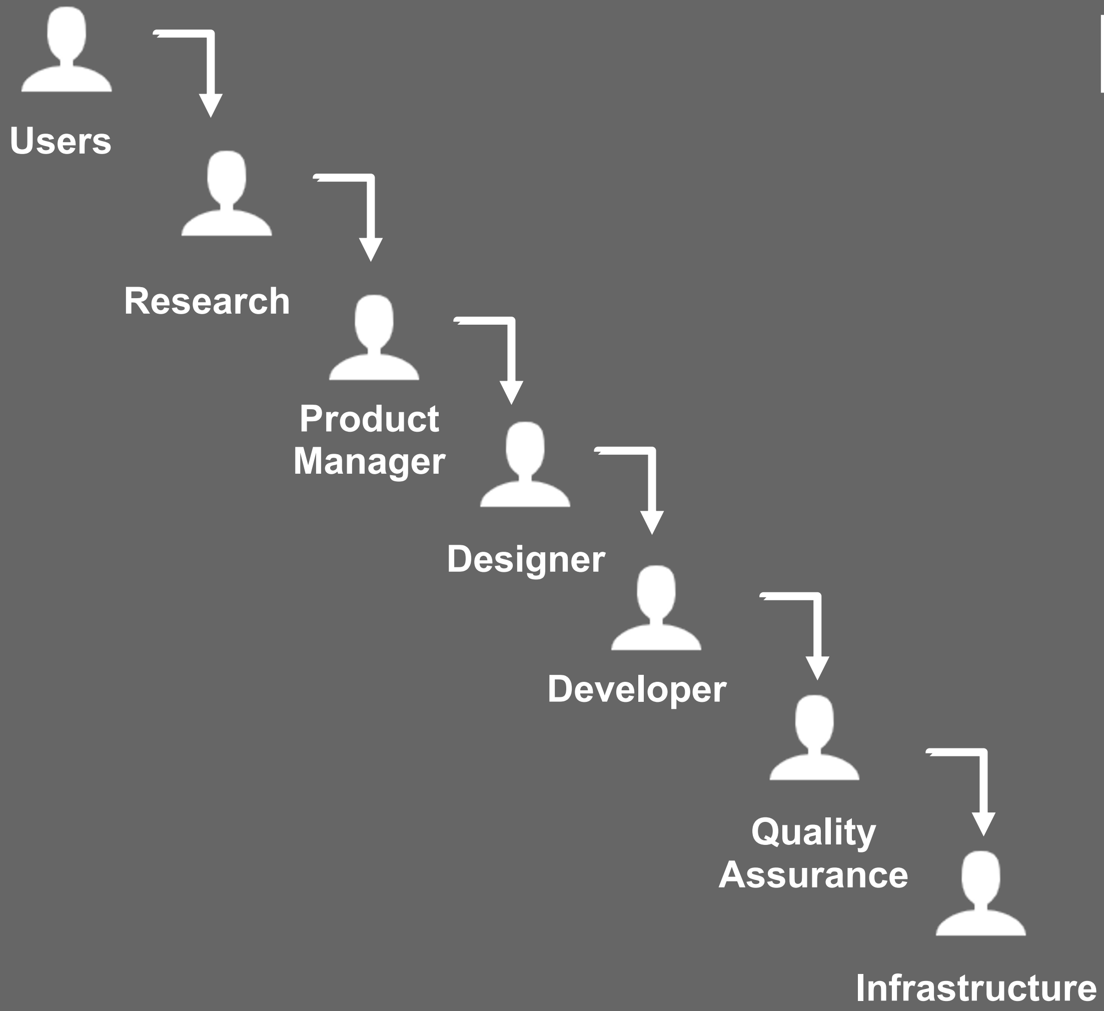
TODO LIST

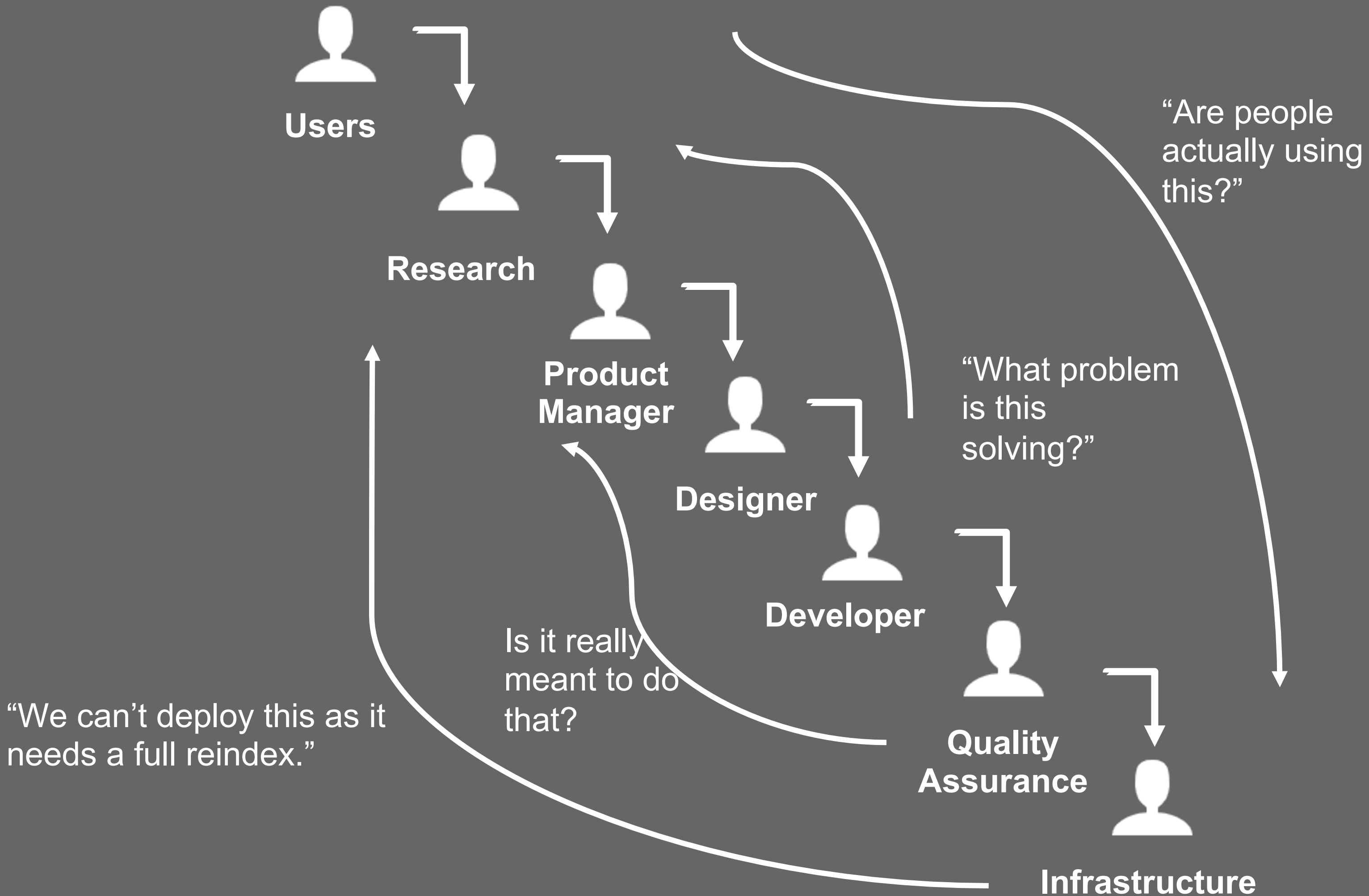
- Understand the change
- Have a consistent path to live for change
- Have automation to verify all change

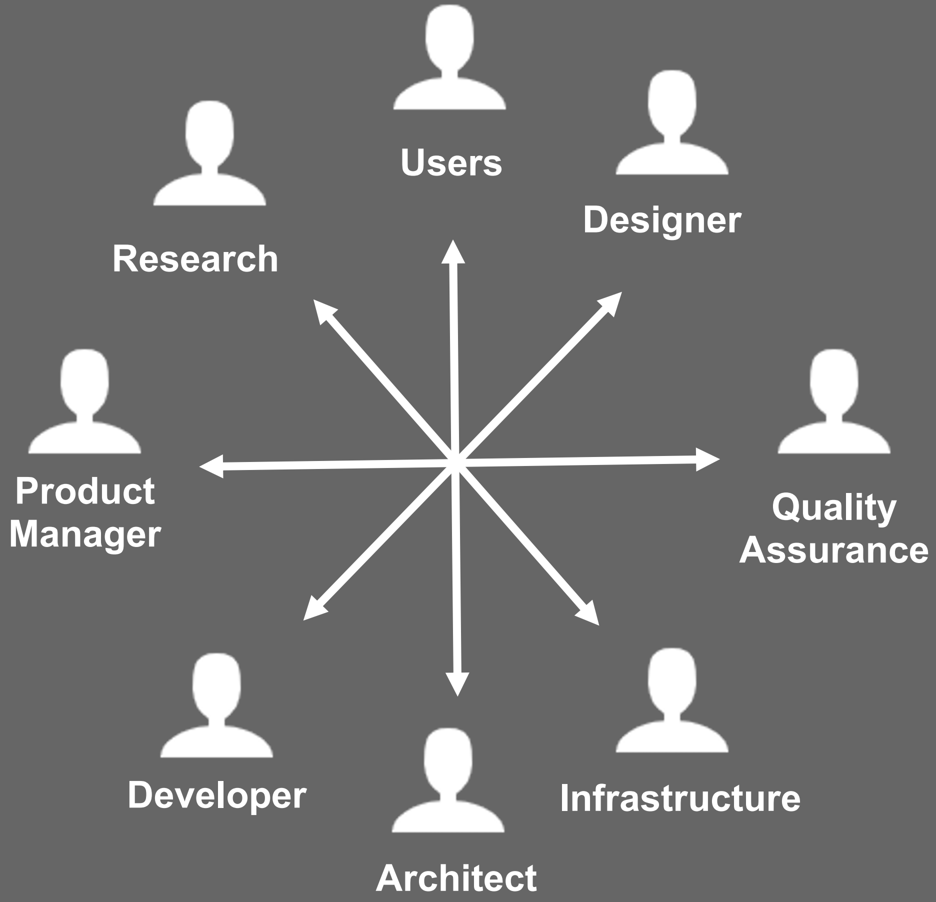
Meanwhile...

TODO: Understand the Change

Flow of work


Users
Research
Product Manager
Designer
Developer
Quality Assurance
Infrastructure

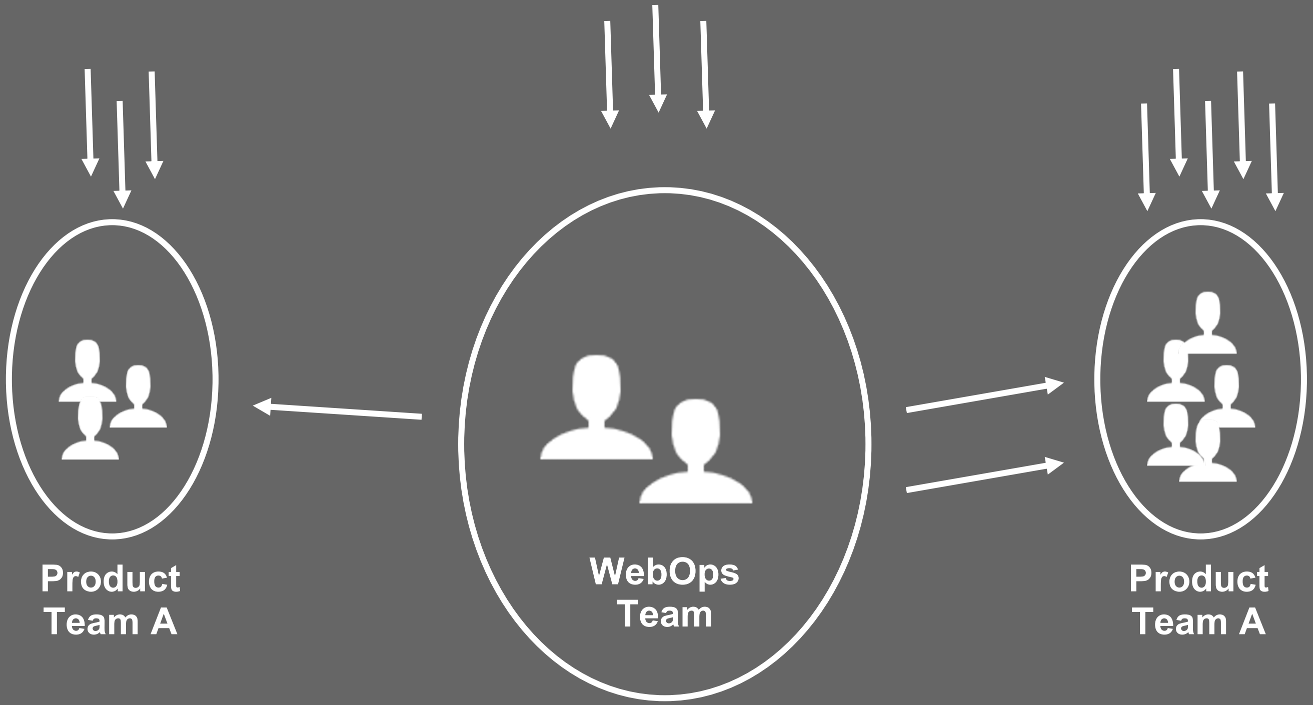

Users
Research
Product Manager
Designer
Developer
Quality Assurance
Infrastructure
“Are people actually using this?”
“What problem is this solving?”
Is it really meant to do that?
“We can’t deploy this as it needs a full reindex.”


Users
Designer
Research
Product Manager
Quality Assurance
Developer
Architect
Infrastructure

Aim : Pair early, pair often

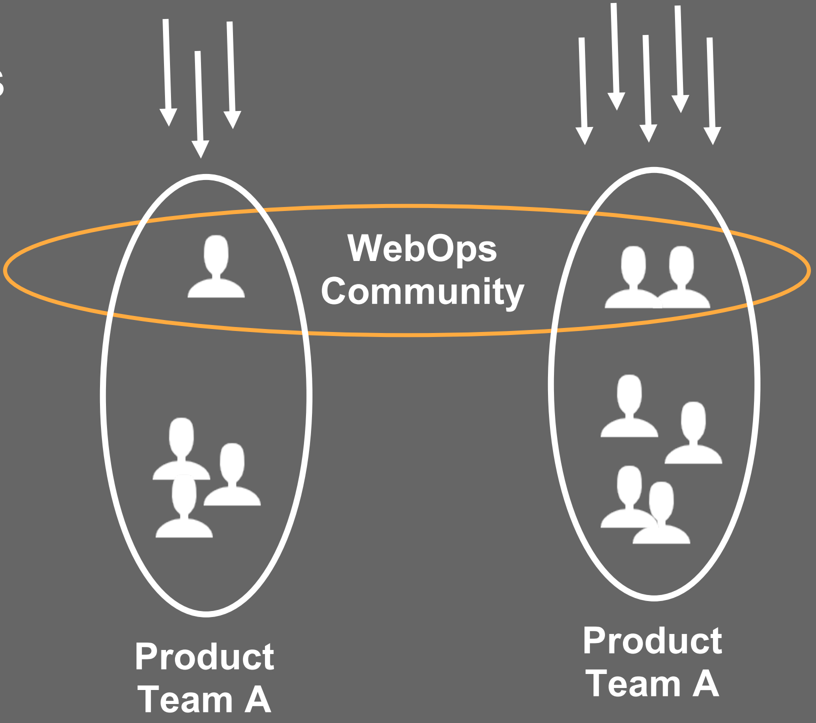
Flow of work to teams

Work priorities



Flow of work to teams

Work priorities

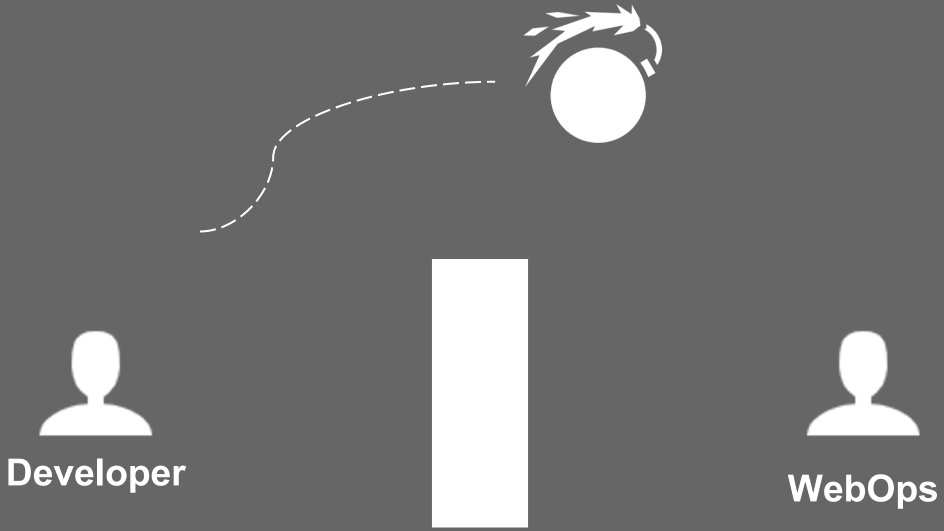


https://web.devopstopologies.com/

Lesson: Communities of Practice are the secret sauce for organisational change

https://emilywebber.co.uk/building-successful-communities-of-practice/

Own the change out to live

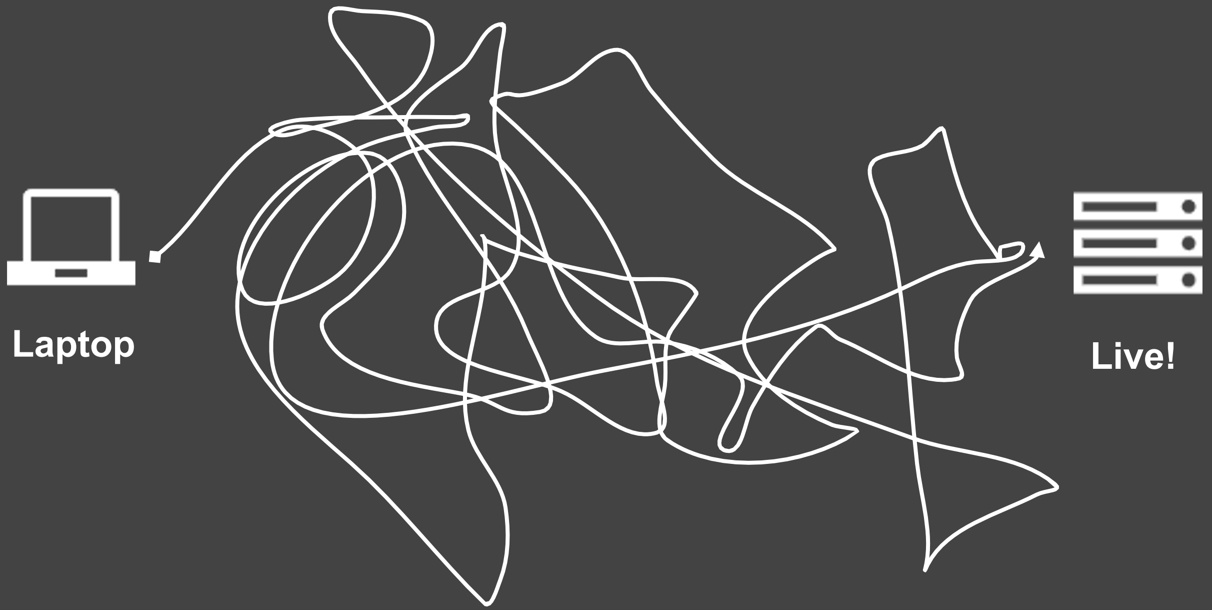


Refine it like you mean it

- Do we know really know what it should do?
- Can we write that as test automation?
- Can we add in a metric to measure its success?
- What observability is needed?
- What is the riskiest thing about it?

Rule #2 : Collaborate early, collaborate often

TODO: Have a consistent path to live...

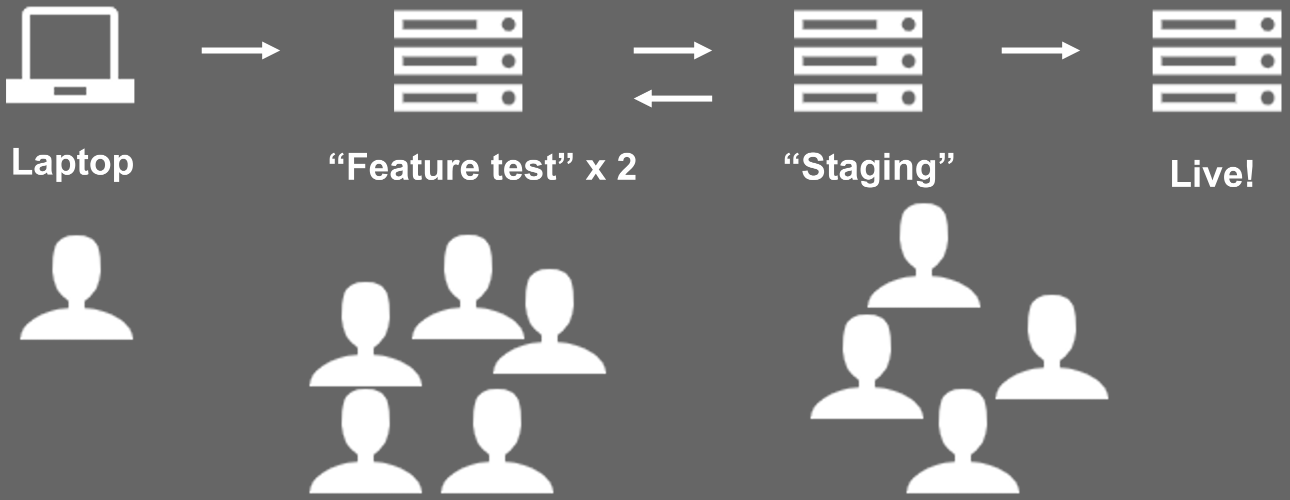


Cyclomatic Complexity of your Path to Live

- Dead ends
- Bottlenecks
- Duplications
- Manual interventions
- Loops within loops

Lesson: Design your path to live like a product

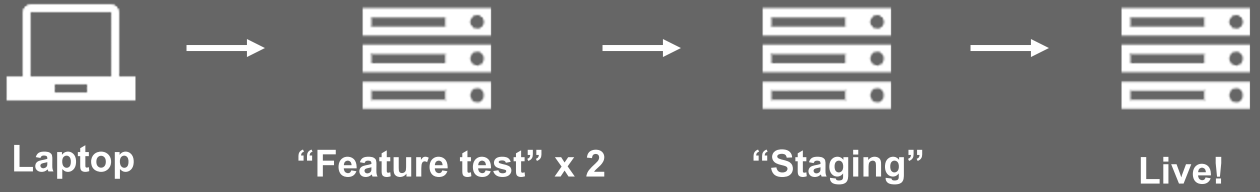
Path to live problem 1 : bottlenecks



Devs, BAs, User Researchers, Product, QA, Ops and their Dog

QA, Product, Ops, Devs

Path to live problem 2: change stacking



Change 1

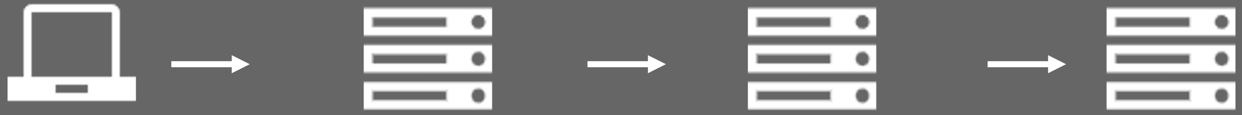
App +
Change 1

App +
Change 1 +
Change 2

App +
Change 1 +
Change 2 +
Change 3

Terraform to the rescue!

Path to live : Automate the things



Laptop

TDD

“PR-based environment” x n

Full App Automated Test

Product Demos

“Staging”

Risk based manual testing

Automated Key User journey tests

Live!

Automated Non-destructive smoke tests

Monitoring

Lesson : have regular pipeline catchups

Rule #3 : make releases easy and boring

Aside: User Stories vs Experiments

Our path to continuous delivery

- Go from fortnightly releases to weekly
- Go from weekly to every two days
- Go to once a day
- Go to twice a day
- Then just release when a story is merged.

TODO : Have automation to verify change

TEST & TRUST

Inspection does not improve the quality, nor guarantee quality. Inspection is too late. The quality, good or bad, is already in the product.

- W. Edward Demming

Identify the gatekeepers and deconstruct their world.

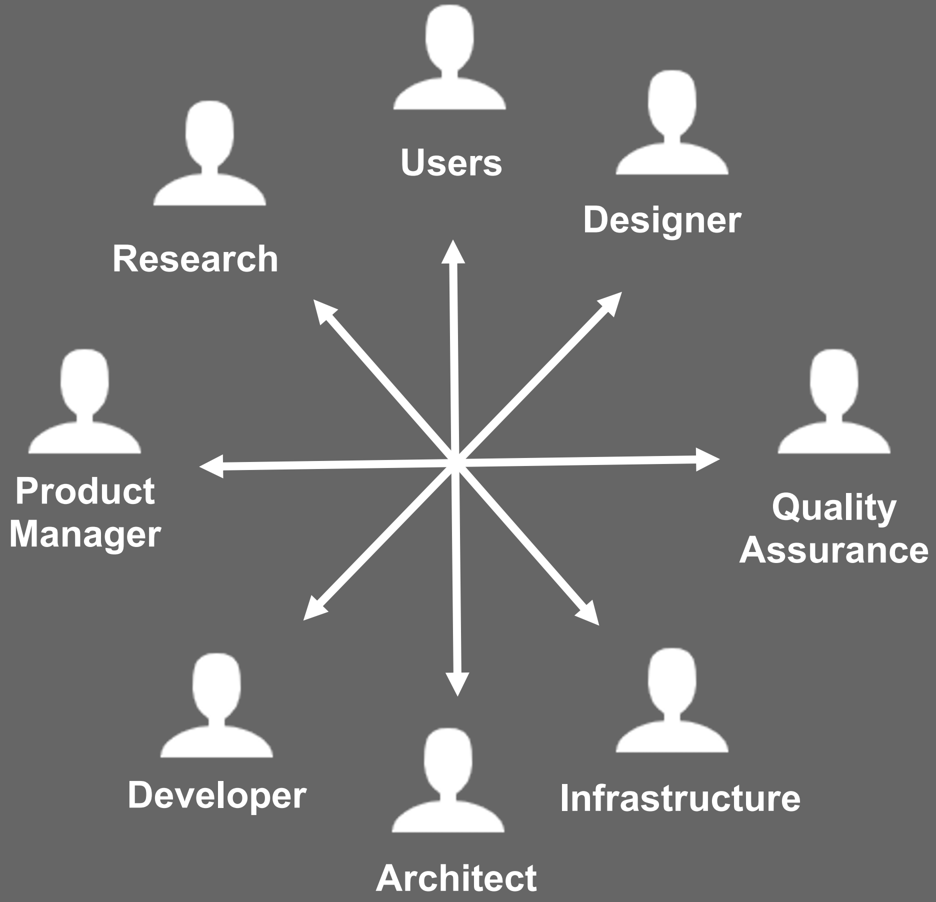
- Work with your gatekeepers to document what they do
- Turn those into user journey tests you can automate
- Show them the automation in action and the outputs
- Get them to trust the automation and let go over time

Clicking about a bit is not a test strategy.

Let’s talk about test

Left-shift testing.

Track your ~~bug~~ quality trends


Users
Designer
Research
Product Manager
Quality Assurance
Developer
Architect
Infrastructure

Rule #4 : It’s not done unless there are automated tests

Our Rules

#1 You should be able to release at any time

#2 Collaborate early, collaborate often

#3 Make releases easy and boring

#4 It’s not done unless there are automated tests

Where did that get us?

- We fixed our infrastructure
- We reduced hosting bills in the range of 70-80%
- We went from 1 release every 2 weeks to 6-8 a day
- Cycle time down from 8 to 3 days on average
- Our users love getting changes faster

...how we made the work work

Icons by icomoon.io and game-icons.net