

A Layered Approach to Progressive Delivery



Do you do this when
your team releases
to production?



Can you remember
a release night
that went like this?



How do you respond
when someone asks,
“How successful
was that release?”





It doesn't have to be that way!

linkedin.com/in/davekarow

What I'm up to: demystifying progressive delivery, especially the role of automated data attribution, early in partial releases.

How I got here: Three decades of experience in developer tools, developer communities, and evangelizing sustainable software delivery practices that deliver impact, without burning out humans.

Where I've been: DHL Worldwide Express, Sun Microsystems, Gupta Technologies, Remedy Software, Marimba (BMC), Keynote Systems (Dynatrace), SOASTA (Akamai), BlazeMeter (CA/Broadcom) and now Split Software.



A Layered Approach to Progressive Delivery

Build Your Way Up to Faster, Safer, Smarter Releases

01

Progressive Delivery: What Is It, Really?

02

Role Models: Progressive Delivery In The Wild

03

The Foundation: Decouple Deploy from Release

04

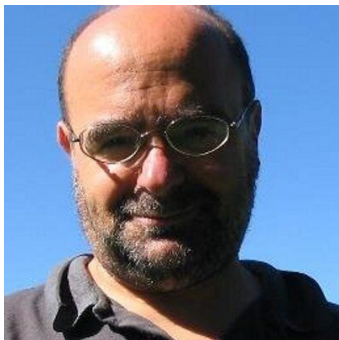
The Upper Layers: Data-Informed Practices, Automated



Progressive Delivery

What Is It, Really?

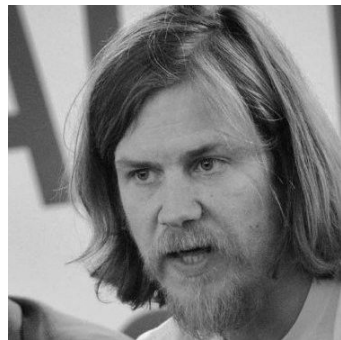
The Roots of “Progressive Delivery”



@SamGuckenheimer

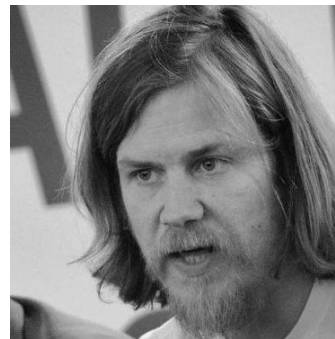
“Well, when we’re rolling out services. What we do is **progressive experimentation** because what really matters is the blast radius. **How many people will be affected when we roll that service out and what can we learn from them?**”

Sam Guckenheimer, quoted in
<https://www.infoq.com/presentations/progressive-delivery/>



@monkchips
(James Governor)

The Roots of “Progressive Delivery”



@monkchips

...a **new basket of skills and technologies** concerned with modern software development, testing and deployment.

Progressive Delivery is the next step after Continuous Delivery, where new versions are deployed to a **subset of users** and are **evaluated** in terms of correctness and performance **before** rolling them to the totality of the users and rolled back if not matching some **key metrics**.

Carlos Sanchez
(Sr. Cloud Software Engineer @ Adobe)

<https://blog.csanchez.org/2019/01/22/progressive-delivery-in-kubernetes-blue-green-and-canary-deployments/>



Role Models:

Progressive Delivery In the Wild

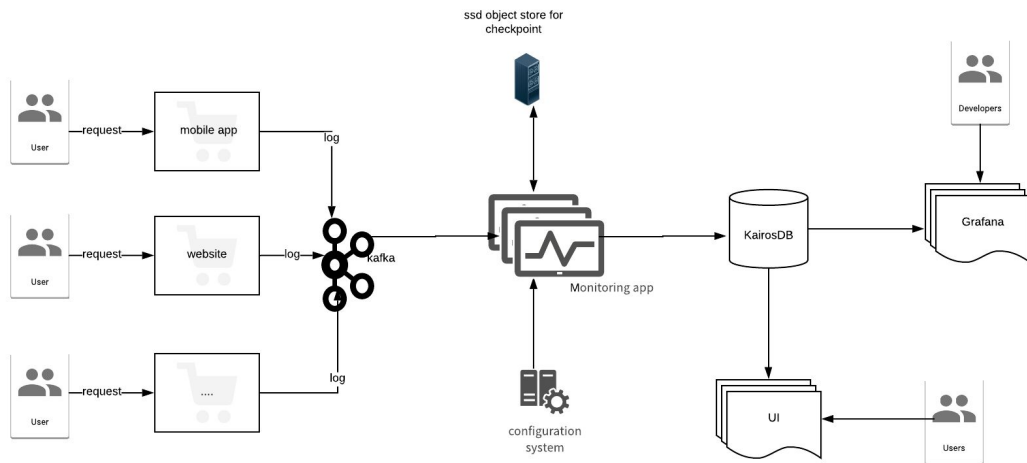
Example: Walmart EXPO

Test to learn

- **Purpose:** Understand customer behavior and validate or invalidate a hypothesis
- Feature likely only launches if it is better than the production experience
- Ex. Guest Cart

Test to launch

- **Purpose:** Mitigating risk by phasing out the rollout to customers, and ensuring no bugs are introduced
- Feature is necessary to launch for the business. Will launch if it is doing no harm
- Ex. ADA



EXPLORE

Demo Experiment

EXPERIMENT RUN	SCHEDULED RUN DATES	EXPERIMENT TOTAL	CONTROL	TREATMENT 1	ASSIGNED LATER
ACQUISITION SELECTED PLOT	NONE	0%	0%	0%	VLC - Layer 18

[EXPLORE](#)
[▶ RUN](#)
[↶ BACK TO CHART](#)
[⌕ FILTER](#)

Experiment Details

Qualifying Conditions

Any

[Factors & Variations](#)

Revision History

Experiment Type: TEST A

Basic Transparencies

Health Check & Traffic Layer

VLC - Layer 18

Multiple Evaluation Layers

LAYER_18

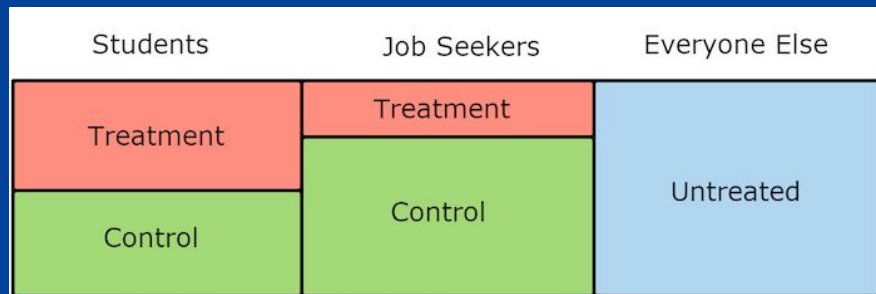
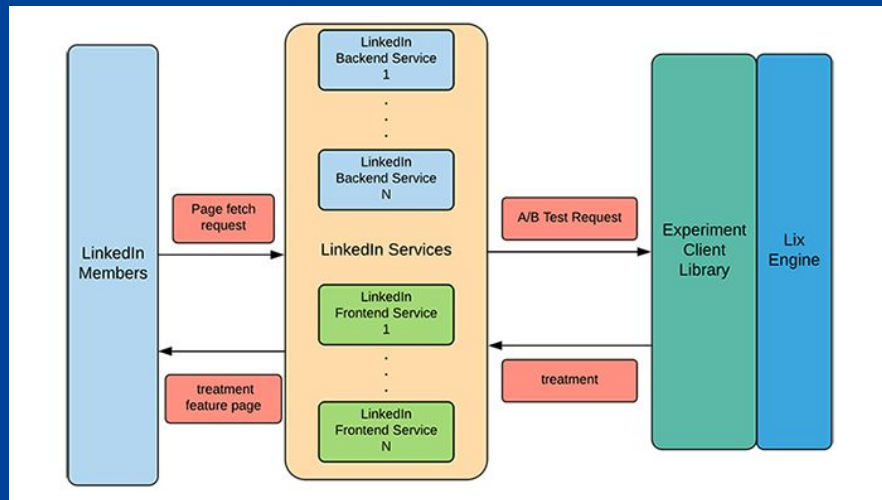
Variations

Control	Variation Name	Variation Spec	Treatment	% of Layer Traffic	Bucket Assignment	Actions
S	Control Demo Experiment Control	EYEZW	Mutate(s) Config(s) Origin(s) none Header/Cookie(s)	0%		➡ ⚙
C	Treatment(s) Demo Experiment Variation	pXZPP	Mutate(s) Config(s) Origin(s) none Header/Cookie(s)	0%		➡ ⚙

It's All About the Data

When someone runs an A/B test, they expect results, and whether they are positive or negative, they need to be correct. One challenge we faced in building our own platform was building trust in the results. With the support of a strong product analytics team, and a revamped data pipeline, we were able to achieve that.

Example: LinkedIn LiX



LIX Failed on Site Speed

Run Time: 2017-02-28 17:43:35 PT

xmc.cache.V2.disable

[1622829](#)
Experiment Id

LIX Key

2017-02-28 14:43:20

Start (PT)

2017-02-28 17:43:20

End (PT)

1

Rules Failed

Failed Rules

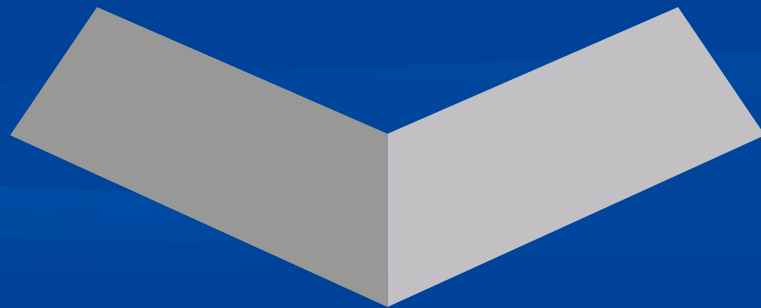
Page Key / Dimensions	Root Cause / Metric	Delta
oz-winner	Server Issue	197 ms (5.45%)
Geo: in	50 pct.	
Segment: 1	Harrier Debug	
Treatment: treatment	Page	

* Run Time is the email sending time. Start and End is the time range of collecting analysis data.

[See Analysis Result](#)



The Foundation: Decouple Deploy from Release



4 Ways to Decouple Deploy From Release (How You Roll Matters)

Approach Benefits	Blue/Green Deployment	Canary Release (container based)	Feature Flags	Feature Flags + Data, Integrated
Avoid Downtime				
Limit The Blast Radius				
Limit WIP / Achieve Flow				
Learn During The Process				





<https://www.split.io/blog/learn-the-four-shades-of-progressive-delivery/>

Harvey Balls by Sschulte at English Wikipedia [CC BY-SA (<https://creativecommons.org/licenses/by-sa/3.0/>)]



@SplitSoftware









4 Ways to Decouple Deploy From Release (How You Roll Matters)

Approach Benefits	Blue/Green Deployment	Canary Release (container based)	Feature Flags	Feature Flags + Data, Integrated
Avoid Downtime				
Limit The Blast Radius				
Limit WIP / Achieve Flow				
Learn During The Process				

<https://www.split.io/blog/learn-the-four-shades-of-progressive-delivery/>

Harvey Balls by Sschulte at English Wikipedia [CC BY-SA (<https://creativecommons.org/licenses/by-sa/3.0/>)]













4 Ways to Decouple Deploy From Release (How You Roll Matters)

Approach Benefits	Blue/Green Deployment	Canary Release (container based)	Feature Flags	Feature Flags + Data, Integrated
Avoid Downtime				
Limit The Blast Radius				
Limit WIP / Achieve Flow				
Learn During The Process				

<https://www.split.io/blog/learn-the-four-shades-of-progressive-delivery/>

Harvey Balls by Sschulte at English Wikipedia [CC BY-SA (<https://creativecommons.org/licenses/by-sa/3.0/>)]

4 Ways to Decouple Deploy From Release (How You Roll Matters)

Approach Benefits	Blue/Green Deployment	Canary Release (container based)	Feature Flags	Feature Flags + Data, Integrated
Avoid Downtime				
Limit The Blast Radius				
Limit WIP / Achieve Flow				
Learn During The Process				

















<https://www.split.io/blog/learn-the-four-shades-of-progressive-delivery/>

Harvey Balls by Sschulte at English Wikipedia [CC BY-SA (<https://creativecommons.org/licenses/by-sa/3.0/>)]



@SplitSoftware

4 Ways to Decouple Deploy From Release (How You Roll Matters)

Approach Benefits	Blue/Green Deployment	Canary Release (container based)	Feature Flags	Feature Flags + Data, Integrated
Avoid Downtime				
Limit The Blast Radius				
Limit WIP / Achieve Flow				
Learn During The Process				

<https://www.split.io/blog/learn-the-four-shades-of-progressive-delivery/>

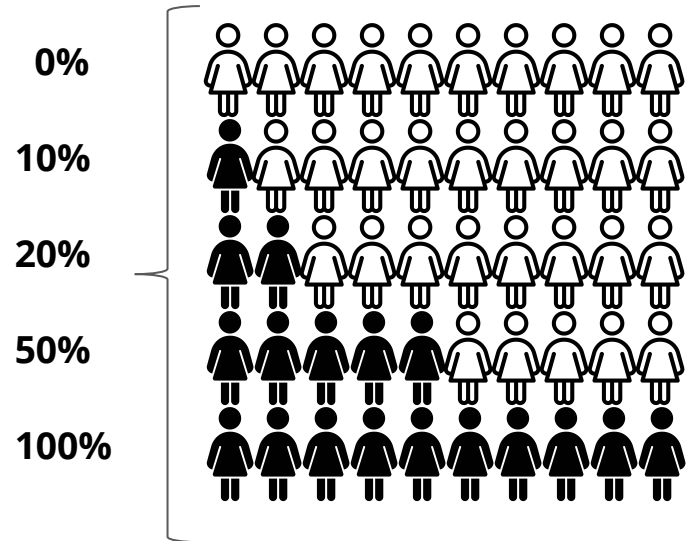
Harvey Balls by Sschulte at English Wikipedia [CC BY-SA (<https://creativecommons.org/licenses/by-sa/3.0>)]



@SplitSoftware

Feature Flag

Like a **dimmer switch** for changes



What a Feature Flag Looks Like In Code

Simple "on/off" example

```
treatment = flags.getTreatment("related-posts");  
if (treatment == "on") {  
    // show related posts  
} else {  
    // skip it  
}
```

What a Feature Flag Looks Like In Code

Multivariate example

```
treatment = flags.getTreatment("search-algorithm");  
if (treatment == "v1") {  
    // use v1 of new search algorithm  
} else if (treatment == "v2") {  
    // use v2 of new search algorithm  
} else {  
    // use existing search algorithm  
}
```

Foundational Capabilities Unlocked By Feature Flags



**Decouple Deploy From Release
With Feature Flags**

- Incremental Feature Development for Flow
- Testing In Production
- Kill Switch (big red button)

04

The Upper Layers: Data-Informed Practices, Automated

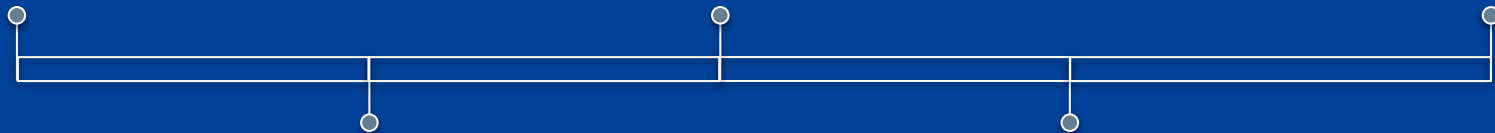




Why Automate Data-Informed Practices? A Different Way to “Ship” Becomes Possible

DEPLOY

Code deployed
No exposure

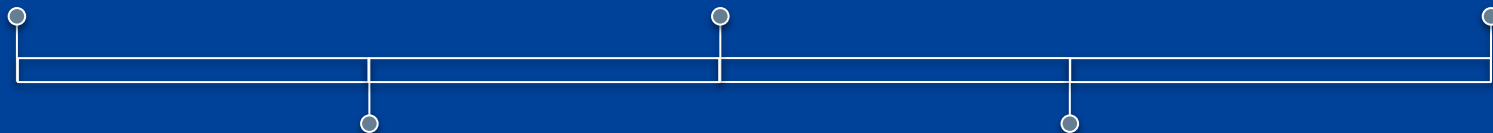




Why Automate Data-Informed Practices? A Different Way to “Ship” Becomes Possible

DEPLOY

Code deployed
No exposure



ERROR MITIGATION

0-50% Ramp
Identify bugs/crashes



Why Automate Data-Informed Practices?

A Different Way to “Ship” Becomes Possible

DEPLOY

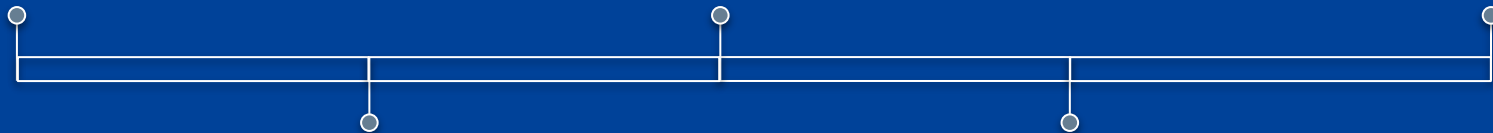
Code deployed
No exposure

MEASURE

Maximum Power Ramp
Understand impact

ERROR MITIGATION

0-50% Ramp
Identify bugs/crashes





Why Automate Data-Informed Practices?

A Different Way to “Ship” Becomes Possible

DEPLOY

Code deployed
No exposure

MEASURE

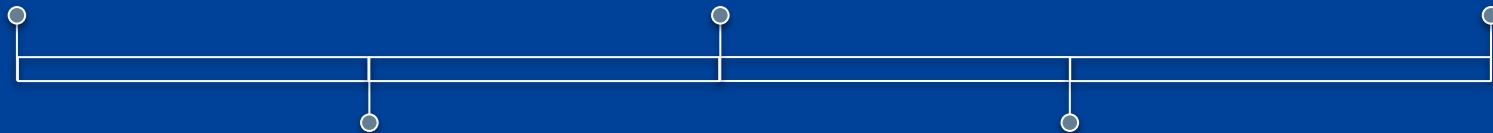
Maximum Power Ramp
Understand impact

ERROR MITIGATION

0-50% Ramp
Identify bugs/crashes

SCALE MITIGATION

50-100% Ramp
Identify scaling issues





Why Automate Data-Informed Practices?

A Different Way to “Ship” Becomes Possible

DEPLOY

Code deployed
No exposure

MEASURE

Maximum Power Ramp
Understand impact

RELEASE

Complete rollout

ERROR MITIGATION

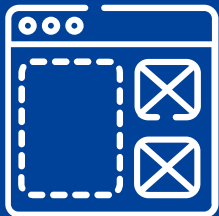
0-50% Ramp
Identify bugs/crashes

SCALE MITIGATION

50-100% Ramp
Identify scaling issues



Heads Up! Not Everything Is As It Seems



New Release

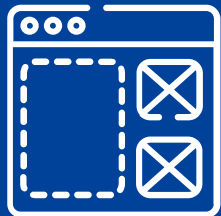


Metrics Change

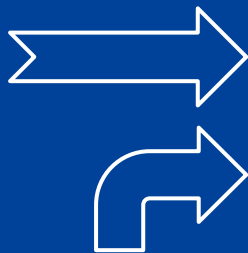
**“Can’t we just
change things
and monitor
what happens?”**



Problem To Solve: Separating Signal From Noise



New Release



Metrics Change



Everything else in
the world

- Product changes
- Marketing campaigns
- Global Pandemics
- Nice Weather



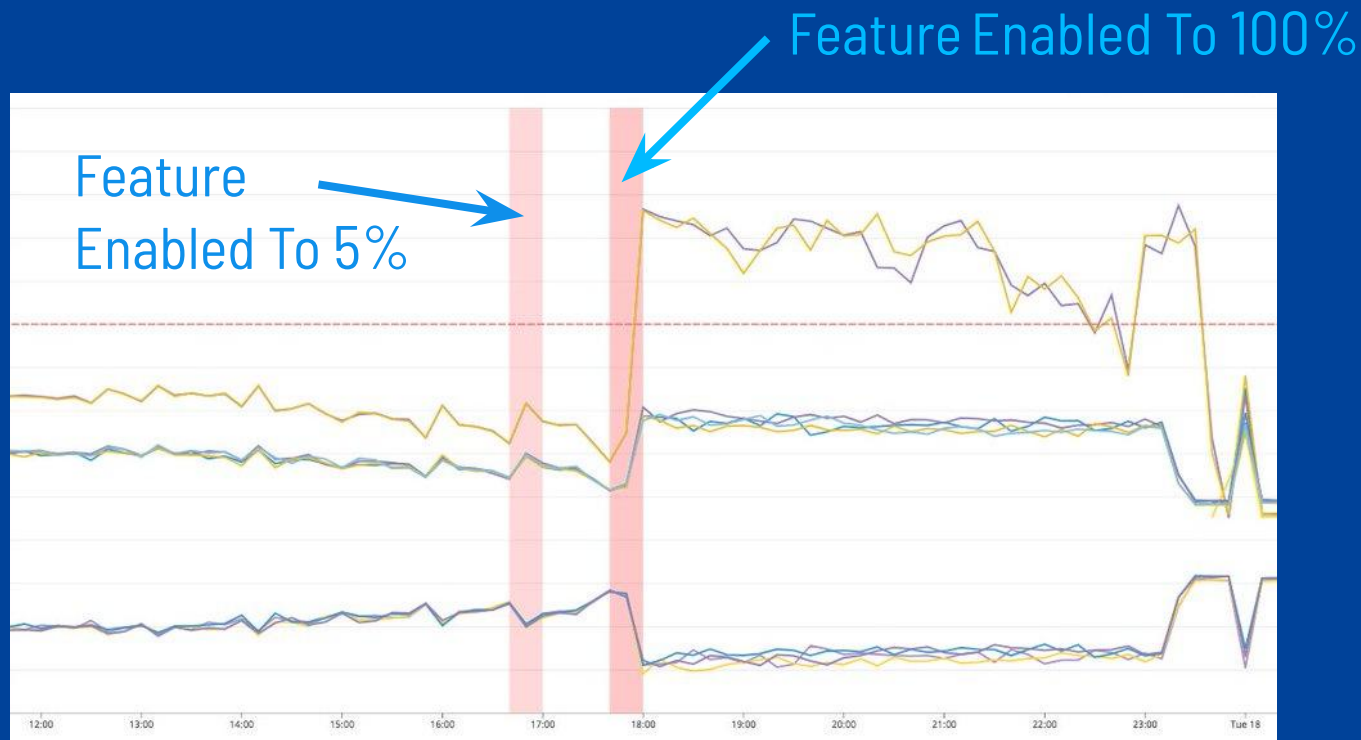
Problem to Solve: Seeing Early Signs of Trouble

Feature Enabled To 100%



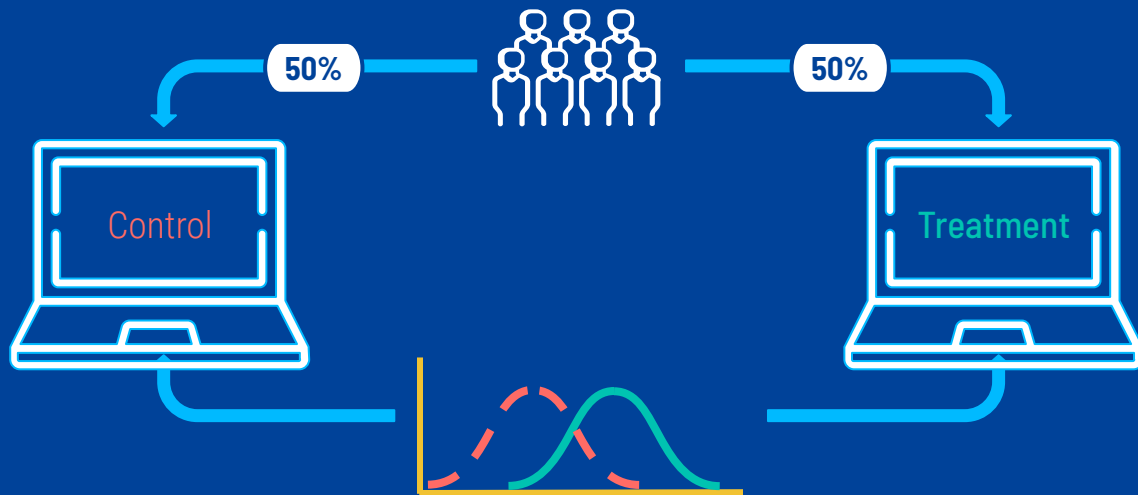


Problem to Solve: Seeing Early Signs of Trouble

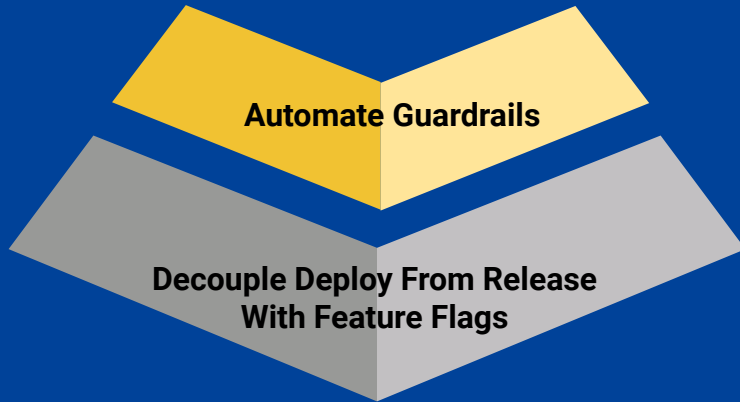




Solution: Cancel Out External Influence With a **Stats Engine**
(Think noise cancelling headphones, but for your metrics)

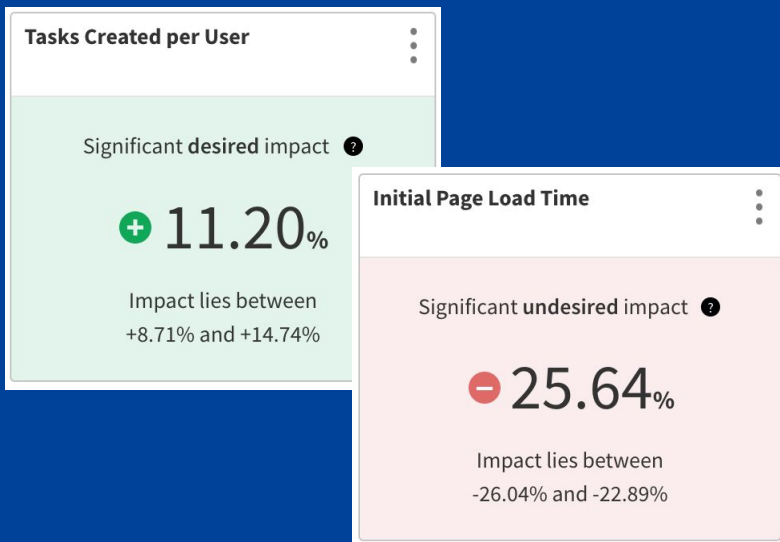


WALK: Automate Guardrails/Do-No-Harm Metrics

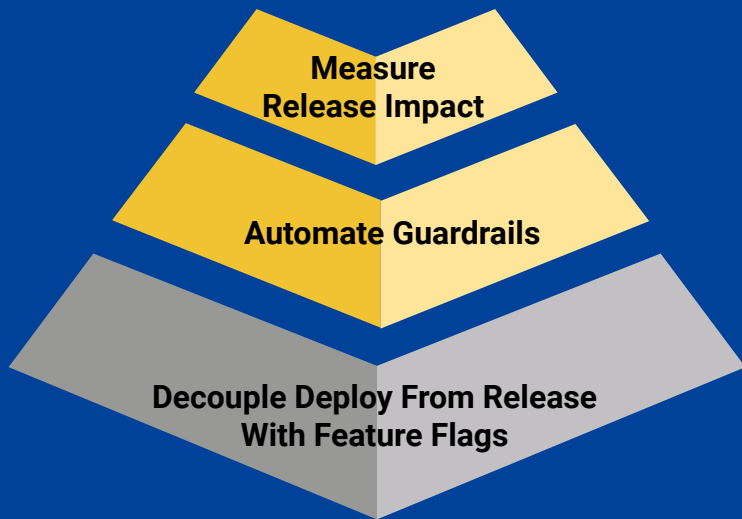


- Alert on Exception / Performance Early In Rollout
- “Limit The Blast Radius” w/o Manual Heroics

Guardrail Metrics

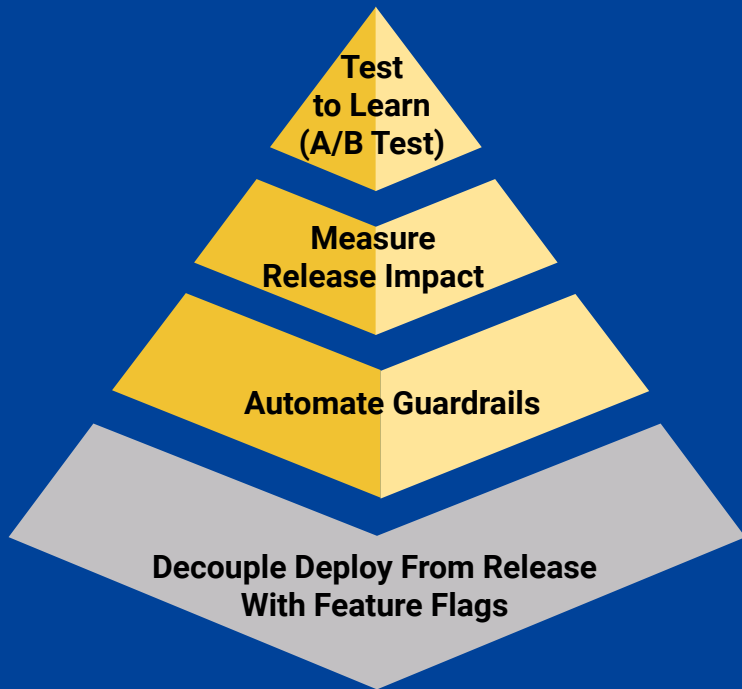


RUN: Measure Release Impact



- Iteration w/o Measurement = Feature Factory 🤡
- Direct Evidence of Our Efforts → Pride 😎

FLY: Test to Learn (A/B Test)



- Take Bigger Risks, Safely
- Learn Faster With Less Investment
 - Dynamic Config
 - Painted Door

This is What Sustainable Software Delivery Looks Like:

DEPLOY

Code deployed
No exposure

MEASURE

Maximum Power Ramp
Understand impact

RELEASE

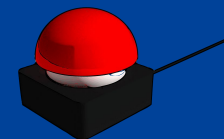
Complete rollout

ERROR MITIGATION

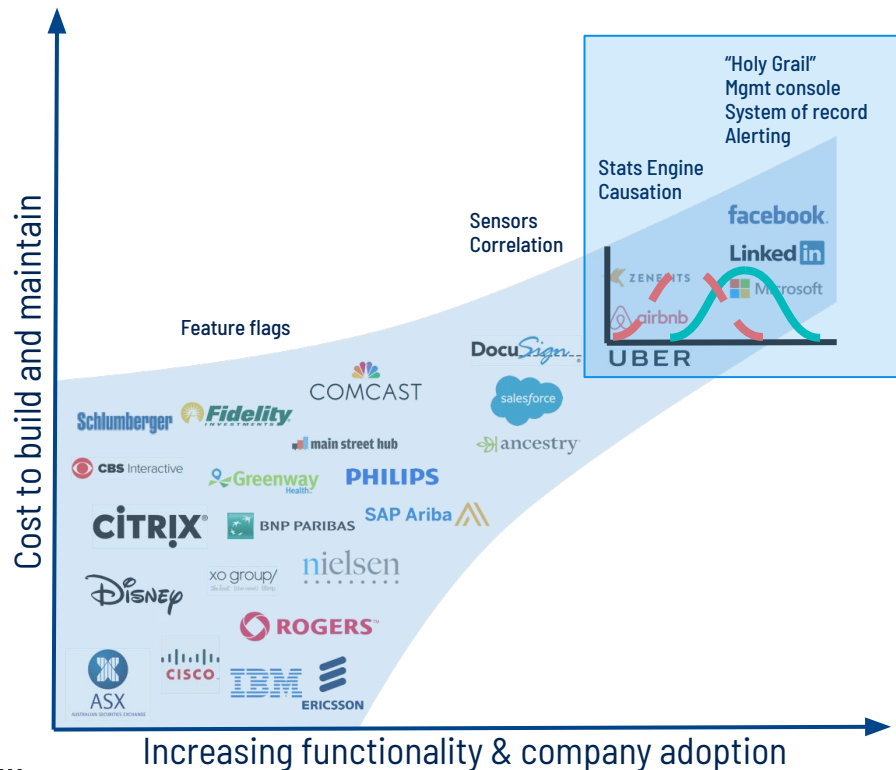
0-50% Ramp
Identify bugs/crashes

SCALE MITIGATION

50-100% Ramp
Identify scaling issues



How In-House Progressive Delivery Platforms Paved The Way For Split (They Proved the value of Layering Up, But It Required Big Investments)



 **Microsoft** | > \$50M annual cost

 **LinkedIn** | > \$30M annual cost

UBER | > \$25M annual cost

Q&A + Booth Invitation



After the talk, come join me in the Split booth in the Expo Hall.

We're also holding a raffle to win an Oculus Quest 2!