

Progressive Delivery: Patterns & Benefits Of Decoupling Deploy From Release

...

Dave Karow
Continuous Delivery Evangelist, Split.io



@davekarow

The future is already here
— it's just not very evenly
distributed.

William Gibson

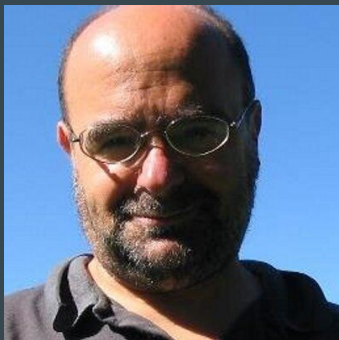
What is **Progressive Delivery** and what are the potential benefits?

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/>

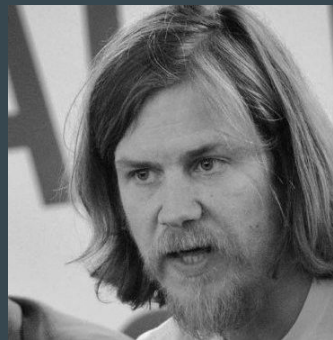
Origin story



@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/>

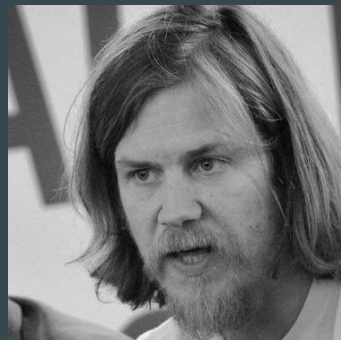


@monkchips
(James Governor)



@davekarow

Origin story



@monkchips
(James Governor)

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



@davekarow

Last Two Years:

Benefits of Progressive Delivery

Avoid Downtime

Limit the Blast Radius

















Limit WIP / Achieve Flow

Learn During the Process



@davekarow

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/>

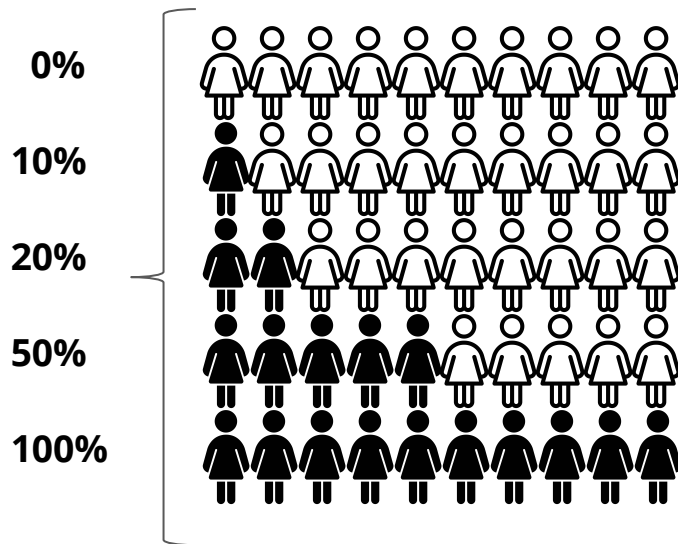


@davekarow

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

Feature Flag

Progressive Delivery Example



What a Feature Flag Looks Like In Code

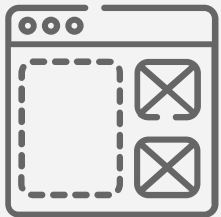
Simple “on/off” example:

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

Multivariate example:

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

DON'T BELIEVE EVERYTHING YOU SEE...



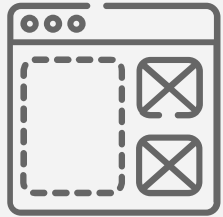
New Release



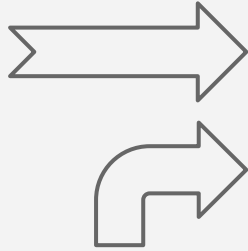
Metrics Change

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

DON'T BELIEVE EVERYTHING YOU SEE...



New Release



Metrics Change



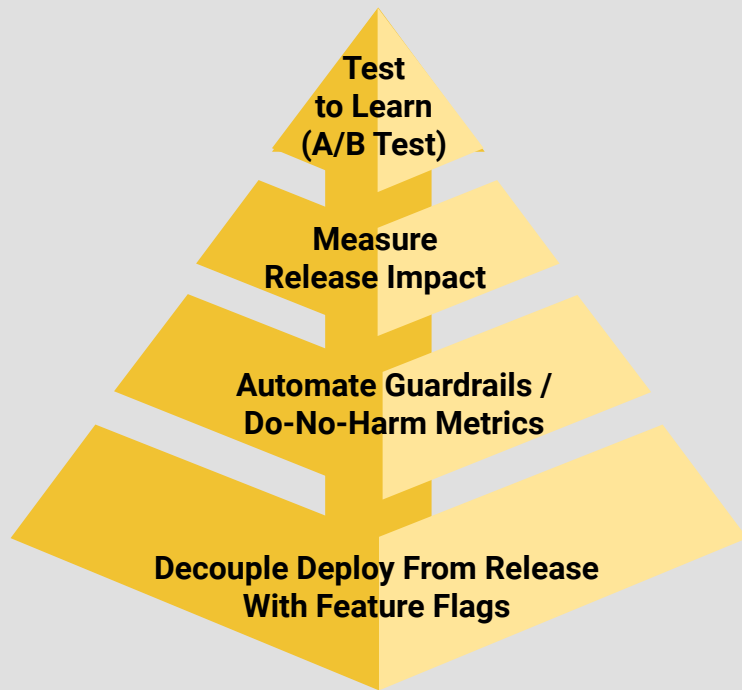
Everything else in
the world

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

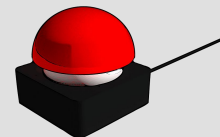
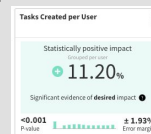
MEASURING CAUSALITY



New! Uplevel Your Game



- Take Bigger Risks, Safely
- Learn Faster With Less Investment
 - Dynamic Config
 - Painted Door
- Iteration w/o Measurement = Feature Factory 🚫
- Direct Evidence of Our Efforts → Pride 😎
- Alert on Exception / Performance Early In Rollout
- “Limit The Blast Radius” w/o Manual Heroics
- Incremental Feature Development for Flow
- Testing In Production
- Kill Switch (big red button)



Lessons Learned From Those Who Have Gone Before Us

Lessons learned at LinkedIn

- **Build for scale:** no more coordinating over email
- **Make it trustworthy:** targeting and analysis must be rock solid
- **Design for every team,** not just data scientists

Ya Xu

Head of Data Science, LinkedIn
Decisions Conference 10/2/2018



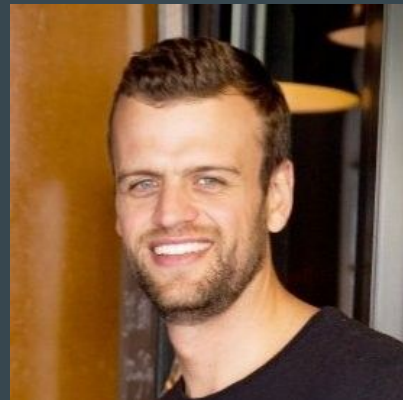
Lessons learned at TVNZ OnDemand (paraphrased by Dave)

- **Don't Assign By Query:** waiting for a static list takes too long!
- **Do Assign On-The-Fly:** randomize those who show up instead
- **Don't Analyze By Hand:** waiting for results kills your cadence

Nathan Wichmann

Product Manager, TVNZ OnDemand

More from Nathan: split.io/customers/tvnz/



@davekarow

Checklists to Level Up:

- Three Foundational Capabilities
- Pattern: Release Faster With Less Risk
- Pattern: Engineer for Impact (Not Output)

Foundational Capability #1

Decouple deploy from release

- ❑ Allow changes of exposure w/o new deploy or rollback
- ❑ Support targeting by UserID, attribute (population), random hash



Foundational Capability #2

Automate a reliable and consistent way to answer,
“Who have we exposed this to so far?”

- ❑ Record who hit a flag, which way they were sent, and why
- ❑ Confirm that targeting is working as intended
- ❑ Confirm that expected traffic levels are reached



Foundational Capability #3

Automate a reliable and consistent way to answer,
“**How is it going for them** (and us)?”

- ❑ Automate comparison of system health (errors, latency, etc...)
- ❑ Automate comparison of user behavior (business outcomes)
- ❑ Make it easy to include “Guardrail Metrics” in comparisons to avoid the local optimization trap



Pattern: Release Faster With Less Risk

Limit the blast radius of unexpected consequences so you can replace the “big bang” release night with more frequent, less stressful rollouts.

Build on the foundational capabilities to:

- ❑ Ramp in stages, starting with dev team, then dogfooding, then % of public
- ❑ Monitor at feature rollout level, not just globally (vivid facts vs faint signals)
- ❑ Alert at the team level (build it/own it)
- ❑ Kill if severe degradation detected (stop the pain now, triage later)
- ❑ Continue to ramp up healthy features while “sick” are ramped down or killed

Pattern: Engineer for Impact (Not Output)

Focus precious engineering cycles on “what works” with experimentation, making statistically rigorous observations about what moves KPIs (and what doesn't).

Build on the foundational capabilities to:

- ❑ Target an experiment to a specific segment of users
- ❑ Ensure random, deterministic, persistent allocation to A/B/n variants
- ❑ Ingest metrics chosen before the experiment starts (not cherry-picked after)
- ❑ Compute statistical significance before proclaiming winners
- ❑ Design for diverse audiences, not just data scientists (buy-in needed to stick)



Whatever you are, try to
be a good one.

William Makepeace Thackeray

Progressive Delivery Resources

<https://www.split.io/pd-in-wild-resources/>



Progressive Delivery Resources

<https://www.split.io/pd-in-wild-resources/>



@davekarow