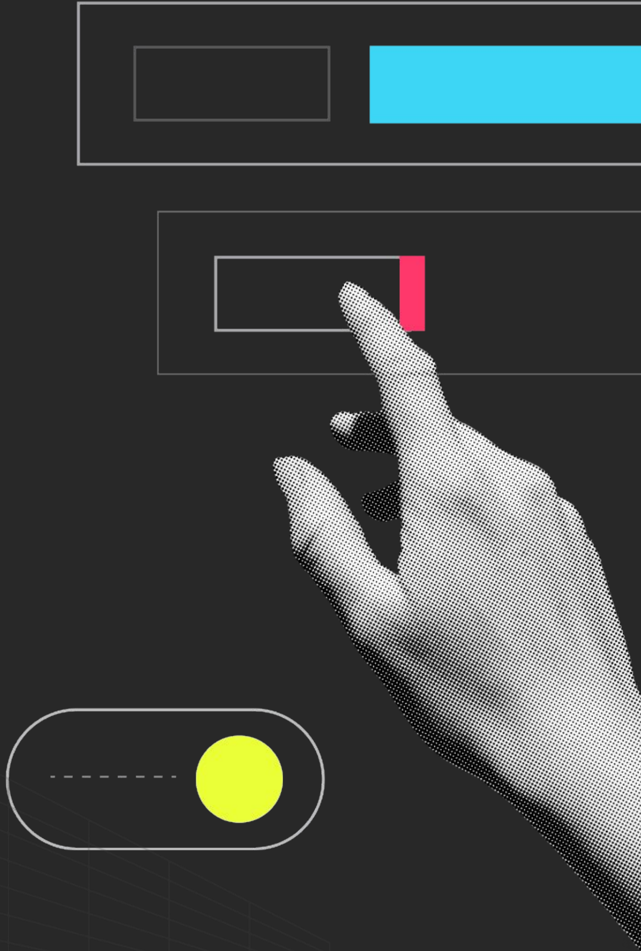
LaunchDarkly

Go Faster, Be Safer

Release Velocity and Psychological Safety

Munnawar Hashim
Developer Advocate
LaunchDarkly
Twitter: @munnawar_h



Starting off with a big question…

Why do we test changes?

Fear of Failure

Outages are costly

A failed deployment can have a cascading effect on platforms and customers.

Failures Damage Perceptions

Unstable environments get a reputation. People stop trusting your products and tools - and adoption slows.

Fear of Becoming Stagnant

Lack of Innovation

If we're not testing constantly - are we not innovating our platforms? Test environments give us freedom to explore.

Slowed Pace

If our pace slows - do our competitors win? Do our users look for more interesting or useful products?

It’s how we’ve always done it

Cultural Processes

“We always test in dev, then QA, and then release to prod” is a common phrase. It’s how we “learn” to develop.

Perception of Safety

Methodical test processes create the perception of safer deployments, but at what cost?

Testing IS a good thing. It’s just about striking a balance.

What if…?

Moving Slowly != Deploying Safely

Instead

Safety is shipping frequently, with greater control

Fast

- ✓ Shipping faster = faster iteration on issues, faster resolution.
- ✓ Moving away from waterfall releases - trunk based development, nightly builds/commit and ship
- ✓ Adjust your needs quickly - accelerate deployments to more groups/faster *smush this in with point 1.
- ✓ Collaborate across projects and release activities

Safe

- ✓ Problematic release? Rollback immediately via killswitch.
- ✓ Gate features behind targets, control pace of rollout, and catch issues in production before they go wide
- ✓ Smaller changes with more predictable impacts and fewer 2am outage calls
- ✓ Integrate with common tooling, and automate releases remove human error

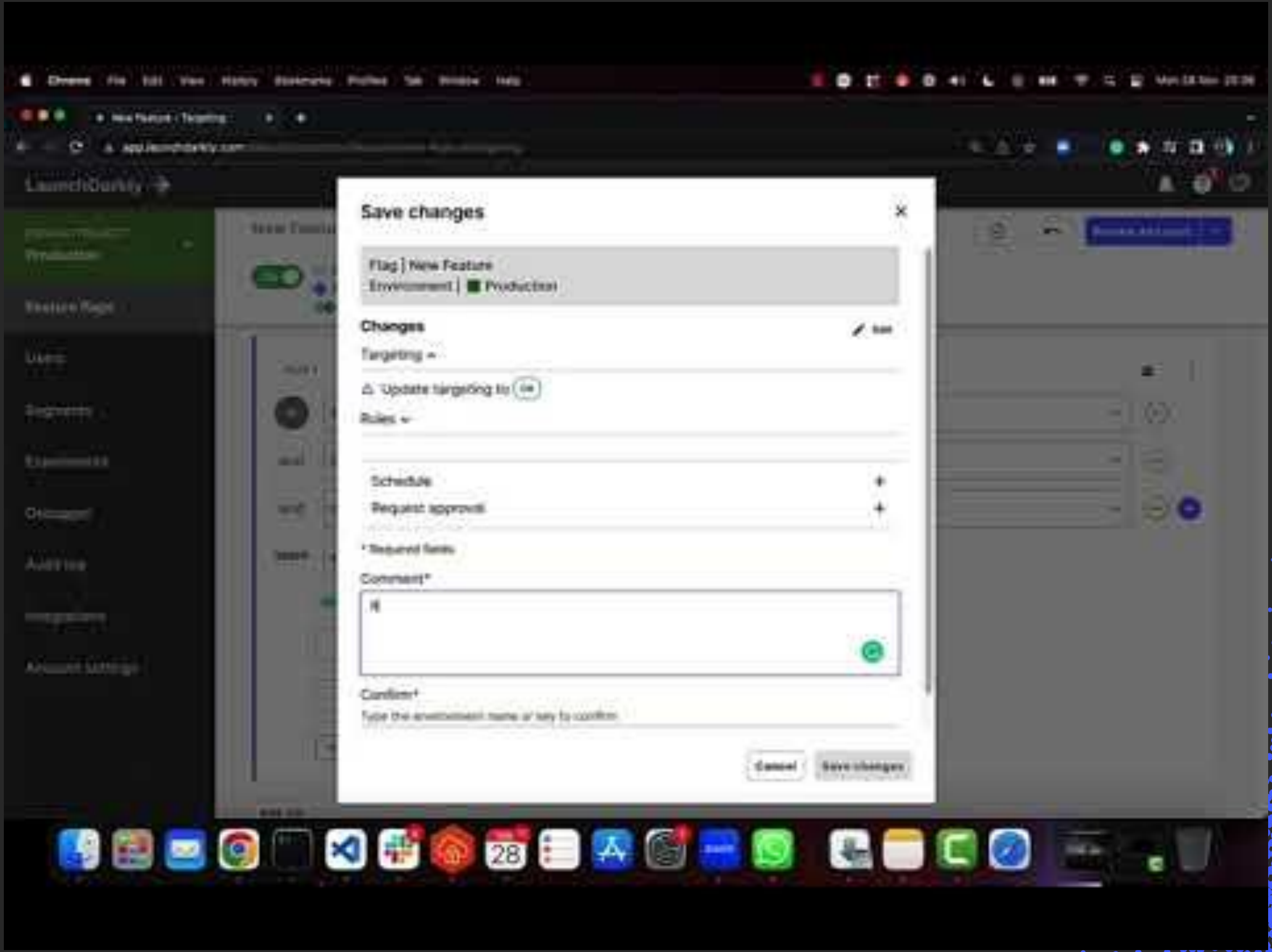
[adult swim]

Fast

- ✓ Shipping faster = faster iteration on issues, faster resolution.
- ✓ Moving away from waterfall releases - trunk based development, nightly builds/commit and ship
- ✓ Collaborate across projects and release activities

Safe

- ✓ Problematic release? Rollback immediately via killswitch.
- ✓ Gate features behind targets, control pace of rollout, and catch issues in production before they go wide. Ensuring smaller changes with more predictable impacts and fewer 2am outage calls
- ✓ Integrate with common tooling, and automate releases remove human error



Save changes

Flag | New Feature
Environment | 🟩 Production

Changes

Targeting

⚠ Update targeting to On

Rules

Schedule +

Request approval +

* Required fields

Comment*

Confirm*

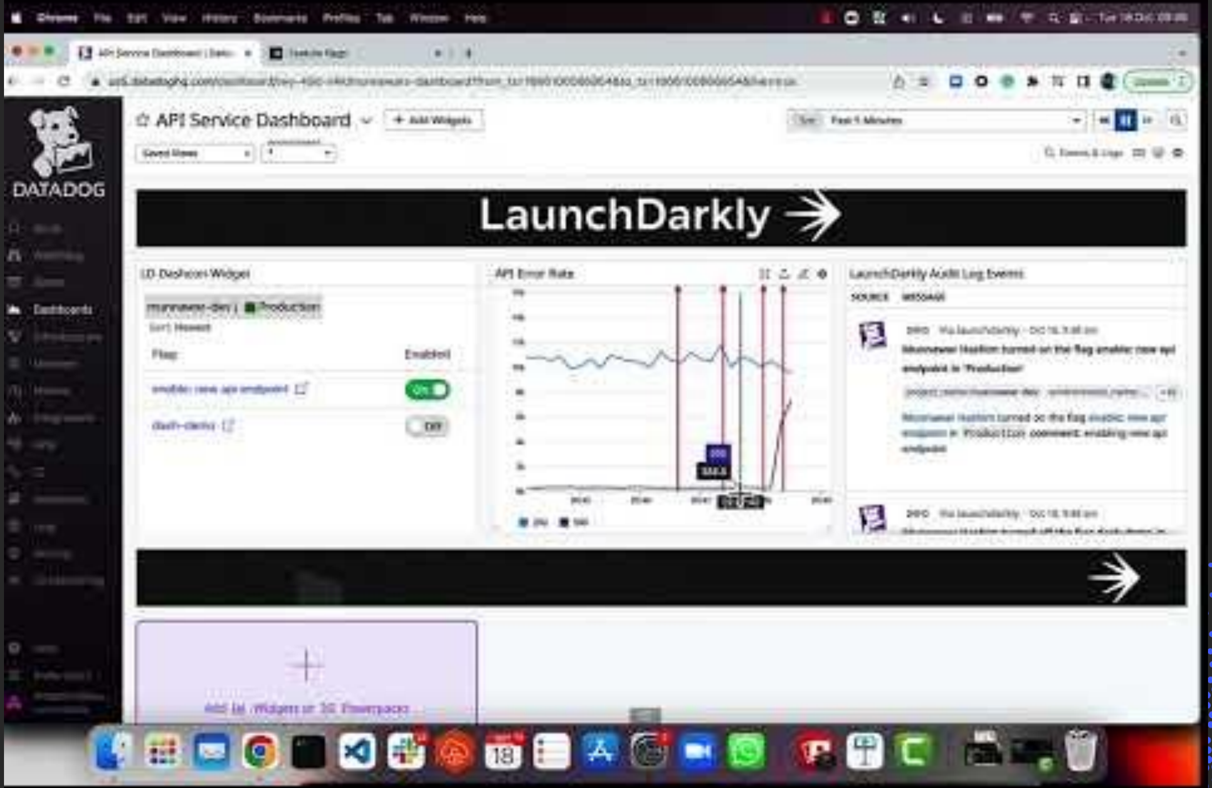
Type the environment name or key to confirm

Fast

- ✓ Shipping faster = faster iteration on issues, faster resolution.
- ✓ Moving away from waterfall releases - trunk based development, nightly builds/commit and ship
- ✓ Collaborate across projects and release activities

Safe

- ✓ Problematic release? Rollback immediately via killswitch.
- ✓ Gate features behind targets, control pace of rollout, and catch issues in production before they go wide. Ensuring smaller changes with more predictable impacts and fewer 2am outage calls
- ✓ Integrate with common tooling, and automate releases remove human error


API Service Dashboard
+ Add Widgets
DATADOG
Dashboards
LaunchDarkly
LD Dashboard Widget
Production
Flag
Enabled
enable: new api endpoint
On
dash-demo
Off
API Error Rate
LaunchDarkly Audit Log Events
SOURCE
MESSAGE
Add Widgets or Powerpacks

Amazon Kinesis	Amplitude	AppDynamics	Azure DevOps	Bitbucket
CircleCi	Cloudflare	Compass	Datadog	Dynatrace
Elastic	Github	GitLab	Google Cloud	Heap
Honeycomb	Jira Software	LogDNA	Microsoft Teams	mParticle

New Relic

Pendo

Segment

servicenow.

ServiceNow

SFx

SignalFx

Slack

Sleuth

Splunk

Terraform

Tray.io

Trello

Visual Studio Code

Webhooks

zapier

Zapier

Zendesk

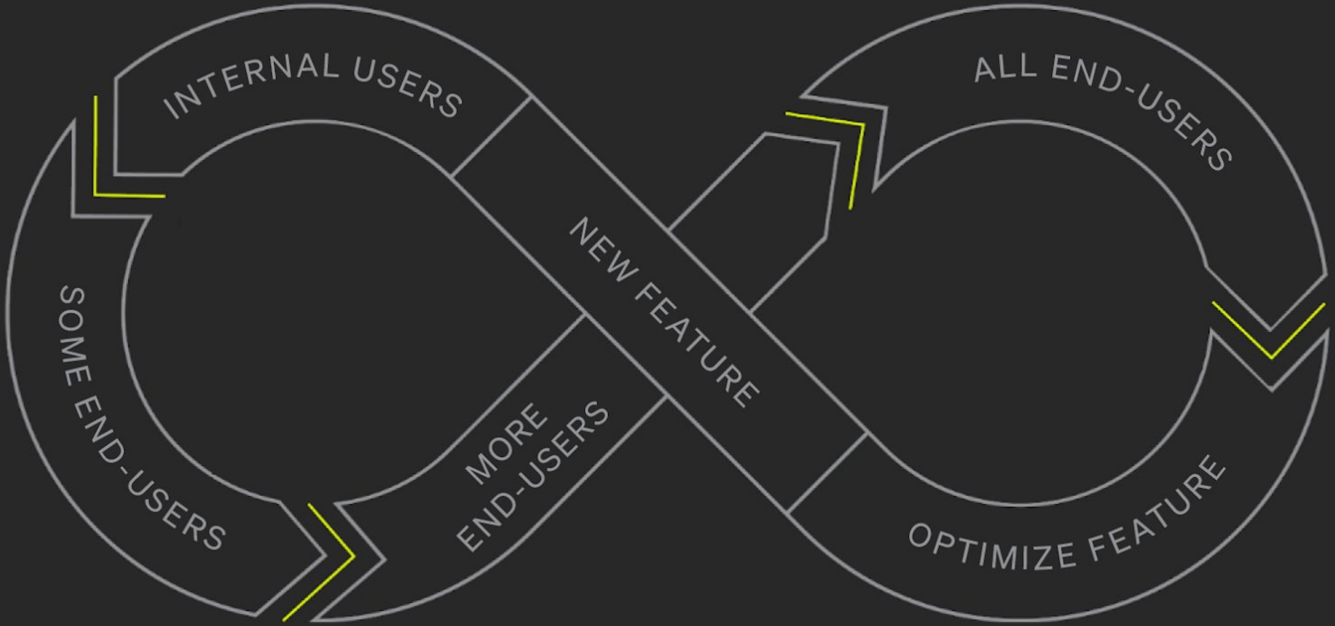
Android	Apex	C/C++ (Client)	C/C++ (Server)	C#	Electron
Erlang	Flutter	Gatsby	Go	Haskell	iOS
Java	JavaScript	Lua	.NET (Server)	.NET (Client)	Node.JS (Client)
Node.JS (Server)	PHP	Python	React	React Native	Roku
Ruby	Rust	Swift	Xamarin		

Instead of making fewer bigger changes, move faster by making many smaller changes

Deployment != Release

- ✓ Deploy code with no release
- ✓ Minimize blast radius of deployments
- ✓ Rollbacks are turning off a feature flag
- ✓ Deployments no longer force branching strategies

Visibility and Control



Validate Performance

Blameless

Mistakes happen. Don’t shoot the messenger - learn together instead.

Non-catastrophic

Smaller changes = smaller bets. Continuously validate decisions and adjust course.

Stress Free

With safety, we focus on delivering value instead of averting disasters.

Ship smaller changes, more often

Use release to course correct without impacting all users

Understand and evaluate successful releases

TLDR;

Ship small, ship often

Keeping continuous

Speed and safety can coexist

Defining your blast radius

Measure your impact

Code matters when it’s measured

@JessicaCregg

If you want to release software more often, make releasing software less scary.

Munnawar Hashim
Developer Advocate
LaunchDarkly
Twitter: @munnawar_h