Continuous Delivery in Reverse

Jonathan Hall
Agile Tour Vienna 2022
September 15, 2022

Raise your hand, and keep it up if…

- Your team does CI/CD
- Your team did a release *to production* last week
- Your team did a release *to production* yesterday
- Your team *does not* have a permanent branch called **`develop`**
- Your team *does not* have a special “hotfix” procedure
- Every developer on your team merged work into **`main`** yesterday
- Your team has no pull requests more than 24 hours old

Continuous Delivery sounds *great*, but we aren’t ready for it here because…

… too many bugs

… depend on manual testing

… team just isn’t “ready”

… regulated industry

Agenda

What is CI/CD?

Why CD is *Hard*

Making CD *Less* Hard

Q&A

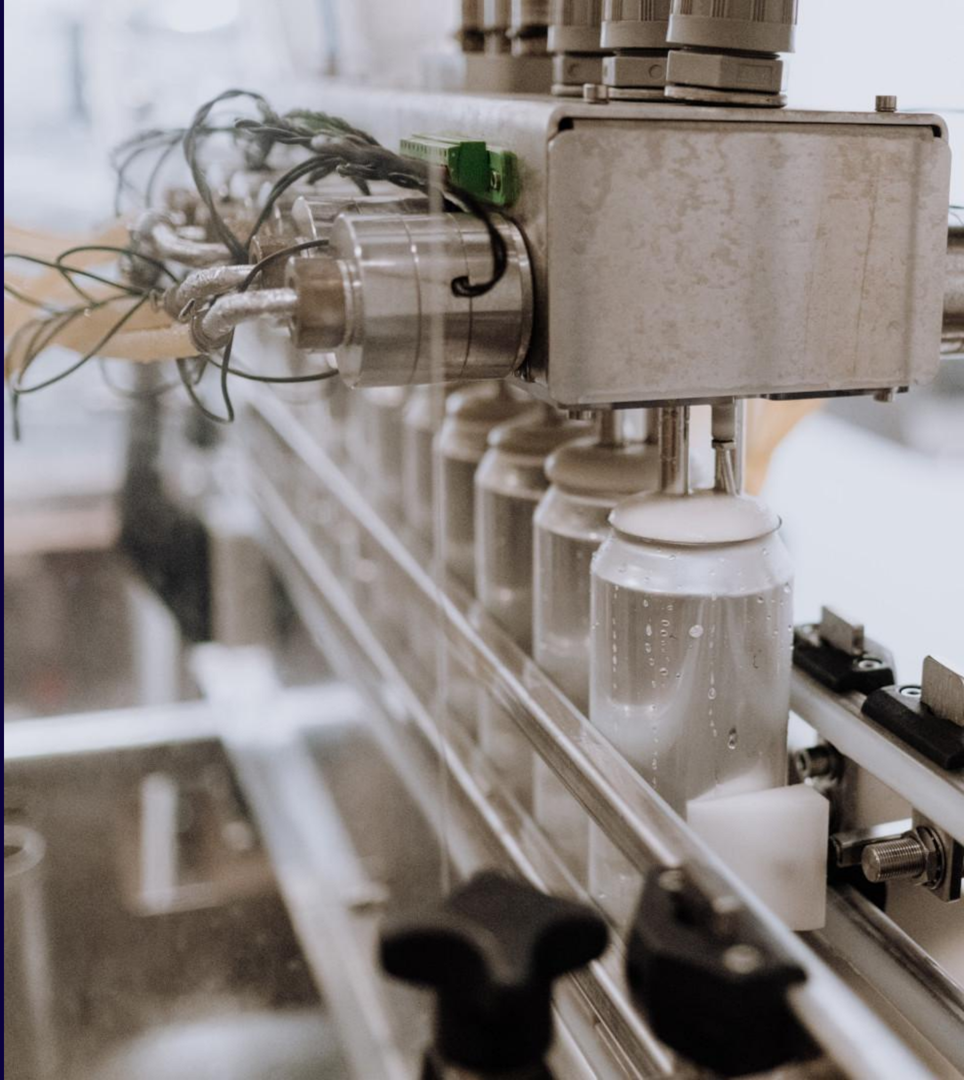
Hi, I’m Jonathan Hall

I help small companies deliver software with big tech confidence, on a small tech budget.

“The Tiny Devops Guy”
Software Delivery Consultant
VP Eng @ HUBUC
Podcaster

https://jhall.io

What is CI/CD?



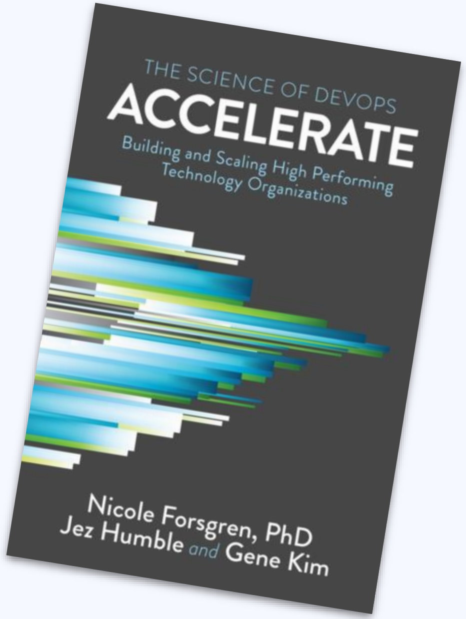
What is *real* CI/CD?

- Continuously …
- … Deliver / Deploy
- What about “CI” (Continuous Integration)?
- It’s a practice …
- … *not* a tool

Okay, but who cares?

“Continuous delivery improves both delivery performance and quality, and also helps improve culture and reduce burnout and deployment pain.”

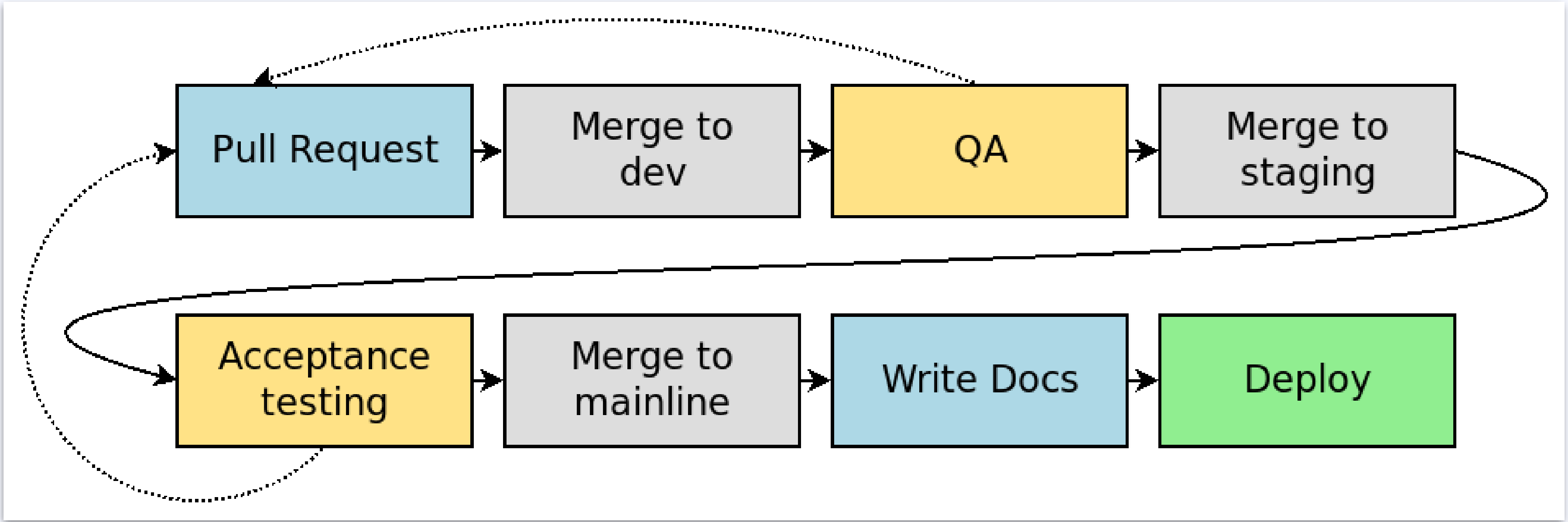
- *Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations*



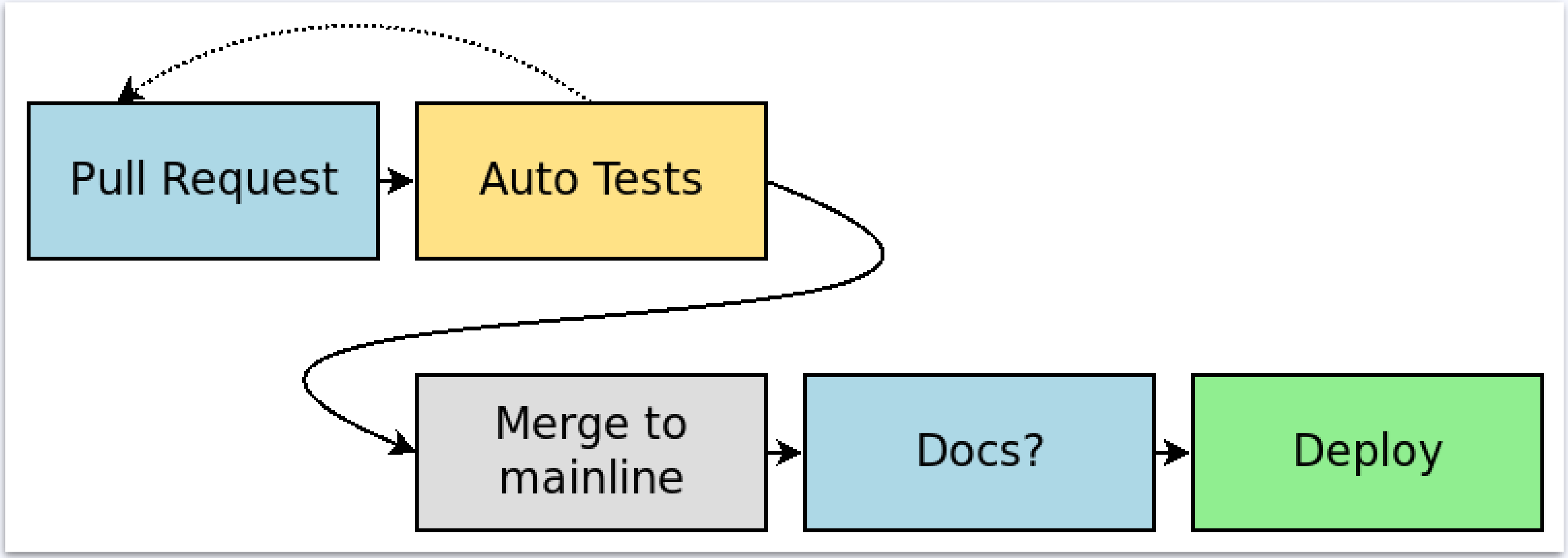
Why CD is *Hard*

Once upon a time at Bugaboo…

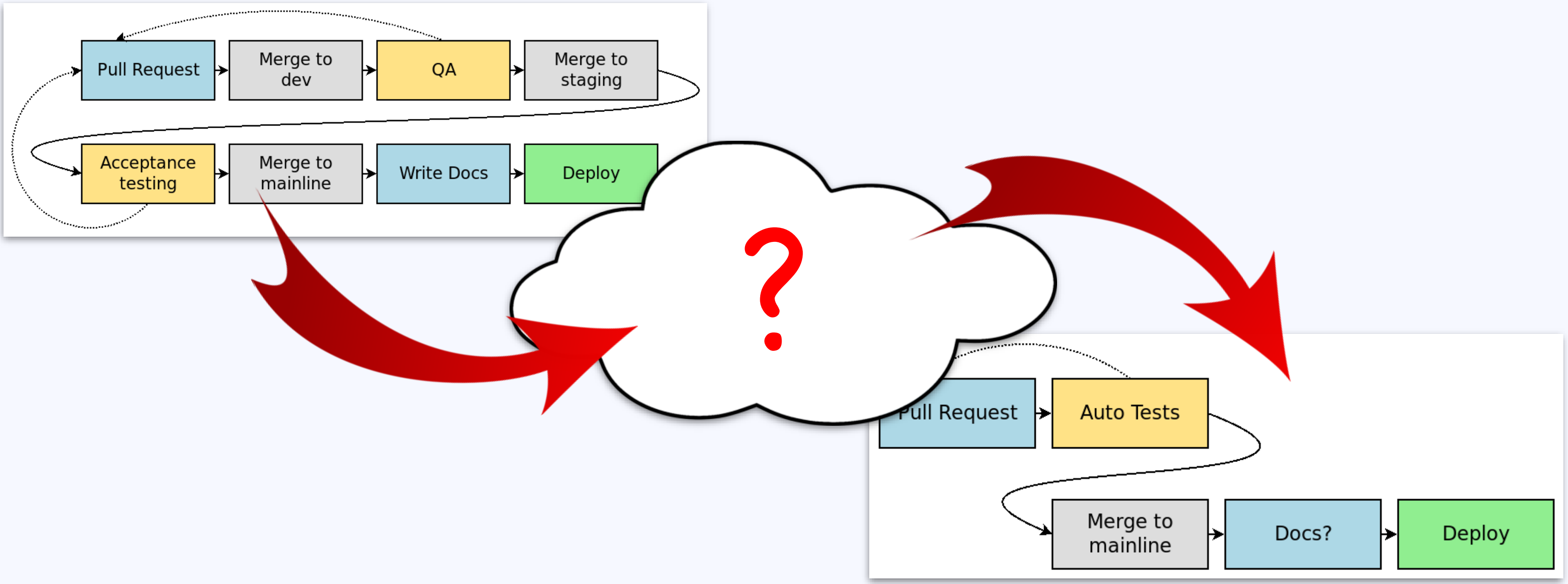
A typical deployment workflow, without CD



The goal?



But how do we get there?



The “Obvious” Approach

1. Write a bunch of automated tests
2. Automate the running of your tests for every code change (“CI” tool)
3. Wait until everyone (the team? management?) is confident that the automated tests are as good as, or better than manual testing
4. Finally, automate the deployment process! 🎉

The problem...

- Writing “enough” automated tests takes months… or years
- New features usually win
- “Confidence” is never achieved
- Fear, Uncertainty & Doubt (F.U.D.) takes over

Making CD *Less* Hard

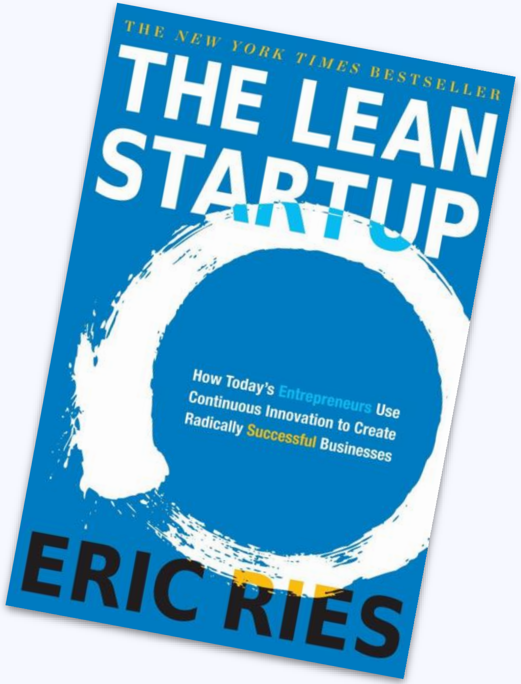
What would an *easy* solution look like?

- No big up-front effort
- No “leap of faith” moment
- Can be adopted piecemeal
- Possible to experiment with
- Can revert process changes that don’t work
- No pieces you don’t need
- Know where to focus your efforts

Wouldn’t an MVP be great?

“That version of a new product which allows a team to collect the maximum amount of validated learning … with the least effort.”

- AirBNB ⇢ Craigslist ad
- Zappos ⇢ Online photos from shoe store
- eCourse ⇢ Landing page?
- Continuous Delivery ⇢ Lean CD


THE NEW YORK TIMES BESTSELLER
THE LEAN STARTUP
How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses
ERIC RIES

Lean CD *reverses* the conventional wisdom

1. Automate the deployment process! 🎉
2. Write automated tests…

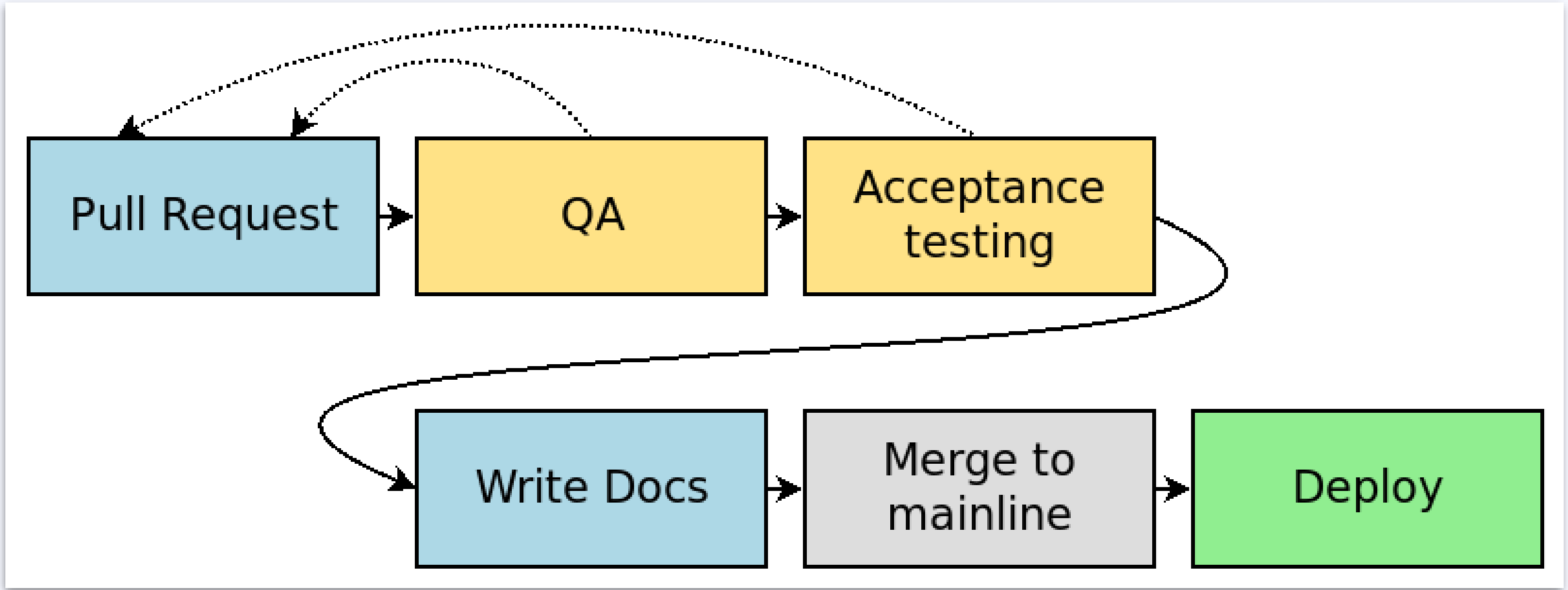
   if you want to*

*I really, really hope you want to!

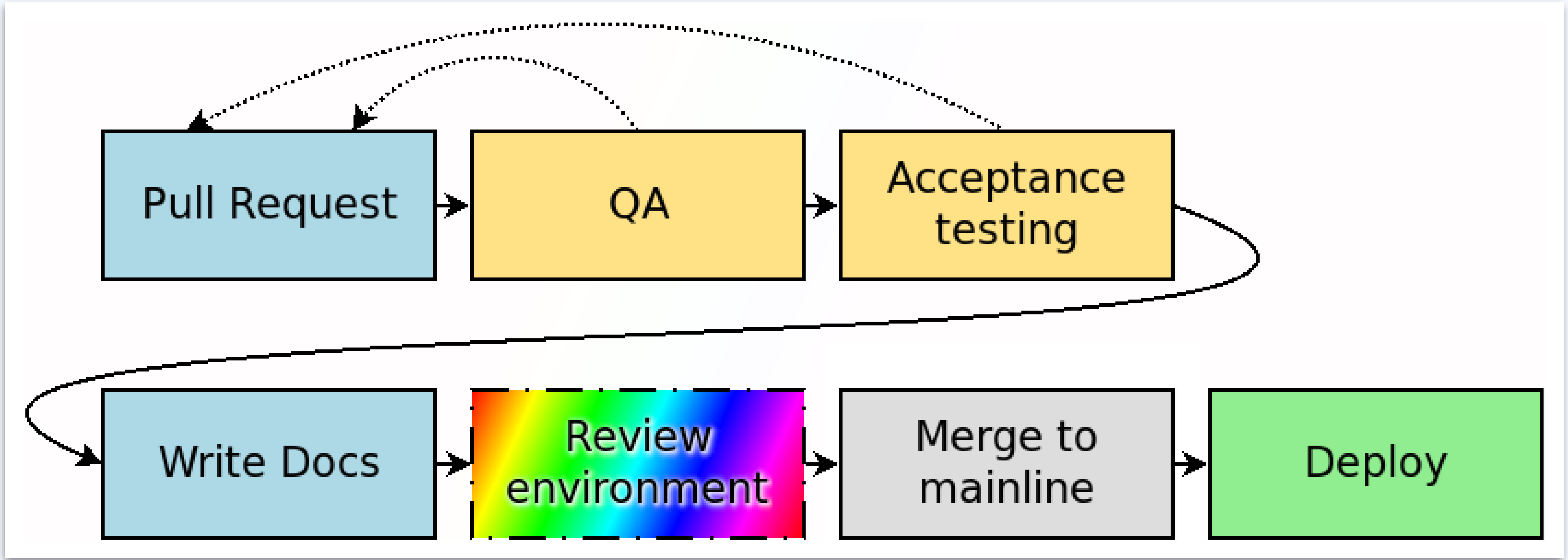
The deceptively simple rules of Lean CD

1. Automate the deployment process
2. Reorder the workflow: *all* manual steps happen before **Merge**
3. Iterate to improve the workflow for your specific needs

Lean CD deployment workflow



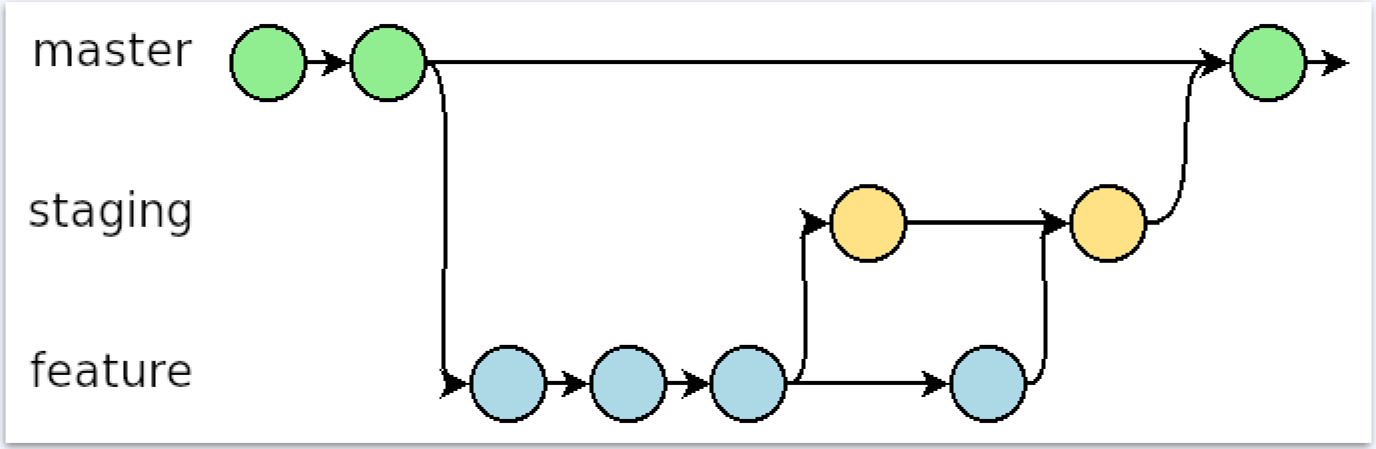
What about testing?



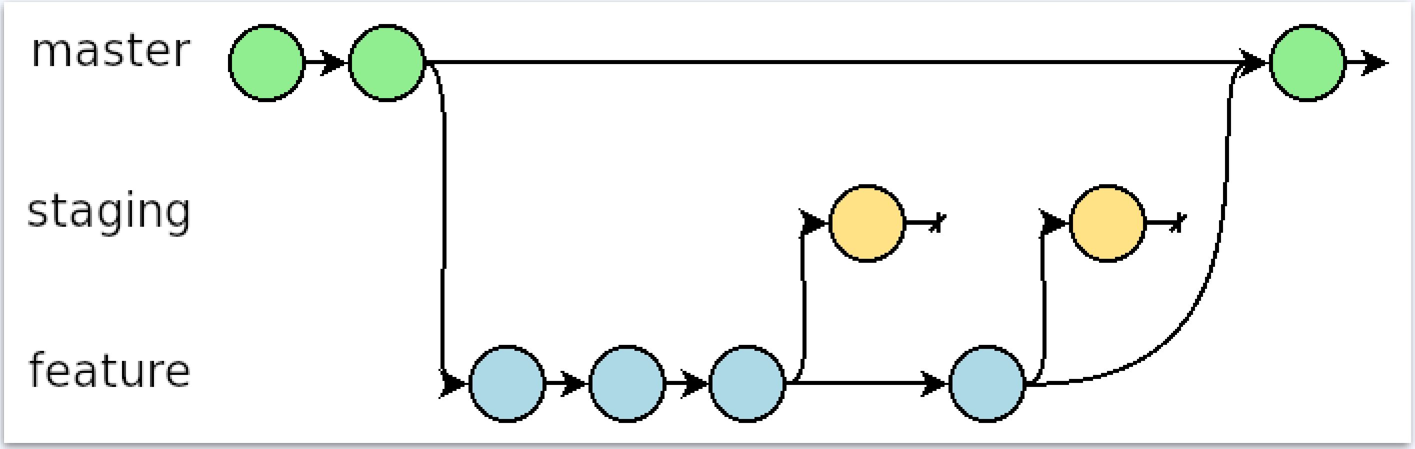
Types of “Review Environments”

- Local development environment (i.e. the dev’s laptop)
- A permanent test (or “staging”) environment
- Multiple, permanent test environments
- Ephemeral preview environments
- Test in production (TIP)

Repurpose your “Staging” environment…



… as a “Review” environment



Pros & cons of a dedicated review environment

Cons

- Slower than local testing
- Can become a bottleneck

Pros

- Slower than local testing
- Can become a bottleneck
- Actually, these problems already existed!
    - The real “Pro” is that now the bottleneck is obvious

Bottlenecks are your cue to improve something…

Improving the test environment situation

- Add new (permanent) review environments
- Add ephemeral preview environments
- Lean more heavily on local testing
- Lean more heavily on TIP

Smaller batches FTW

- Work in smaller batches (Smaller PRs, smaller commits, etc)
- Smaller batches are less risky
- Smaller batches are easier to review
- Smaller batches are easier to test
- Smaller batches are easier to revert
- Smaller batches reveal the *true nature* of your bottlenecks

Where to next?

1. Identify a bottleneck
2. Find a creative solution
3. Rinse, and repeat

Need some help?

LeanCDSeminar.com

4 weeks, starting October 3, 2022

Free for conference attendees

Coupon code: **VIENNA**

Questions?

https://jhall.io/vienna-2022

https://leancdseminar.com (**VIENNA**)

https://jhall.io