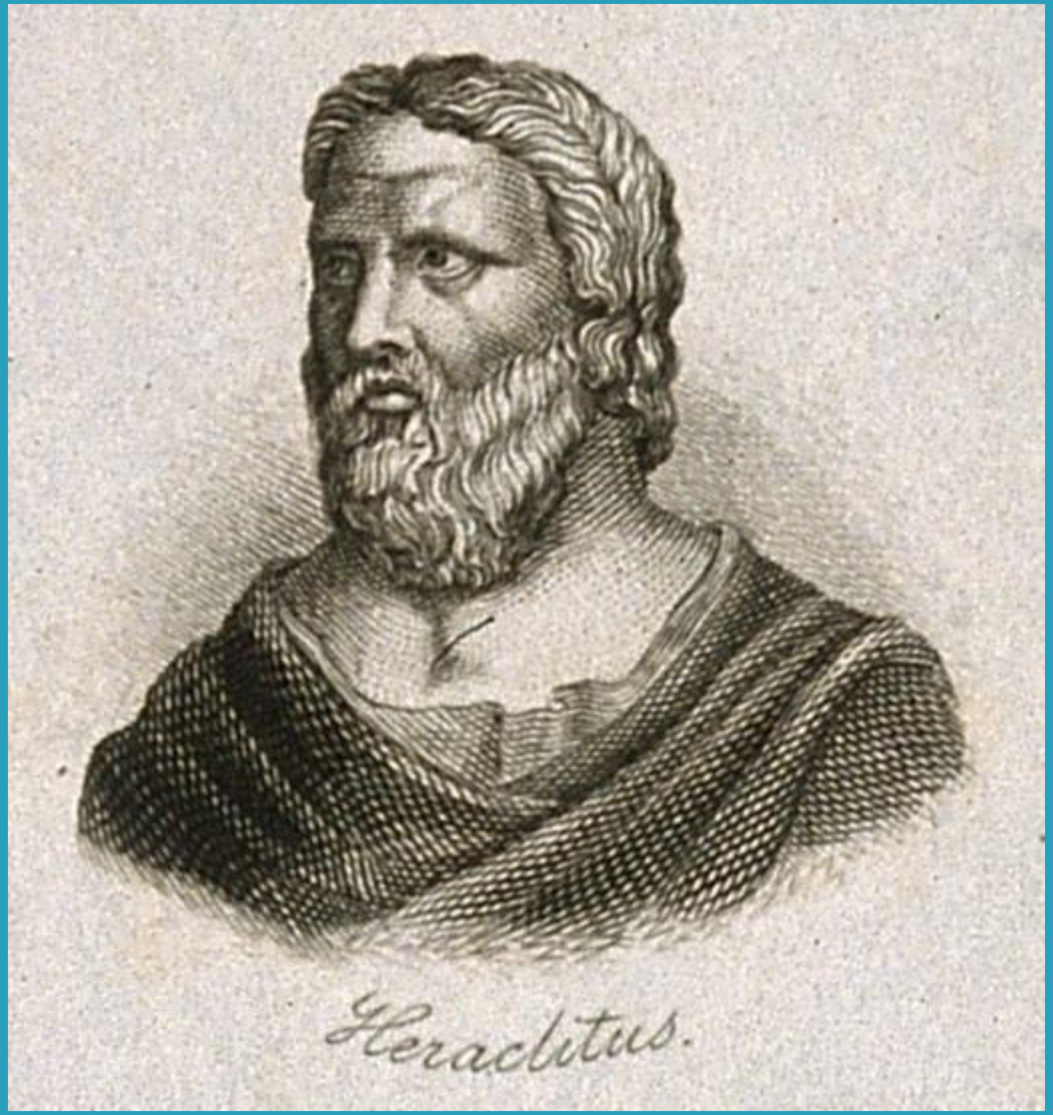
Verifying Knowledge in DevOps

Chris B. Behrens
Senior Software Developer

@chrisbbehrens

Change is the only constant



Relating This to Lean

Handoffs are consequences of change

Because change happens, we need to defer commitment

Above all, create knowledge

Automated Testing

A precise definition of “test”

An expectation, an observation and a reconciliation

Any part can be wrong

The expectation can be wrong

The reconciliation can be wrong

When the observation is wrong, then we have created knowledge

Change has broken our premises – “regression”

Increase the coverage of your unit tests

Better Seen Than Heard

Demo

Whip up a quick unit test project and a unit test

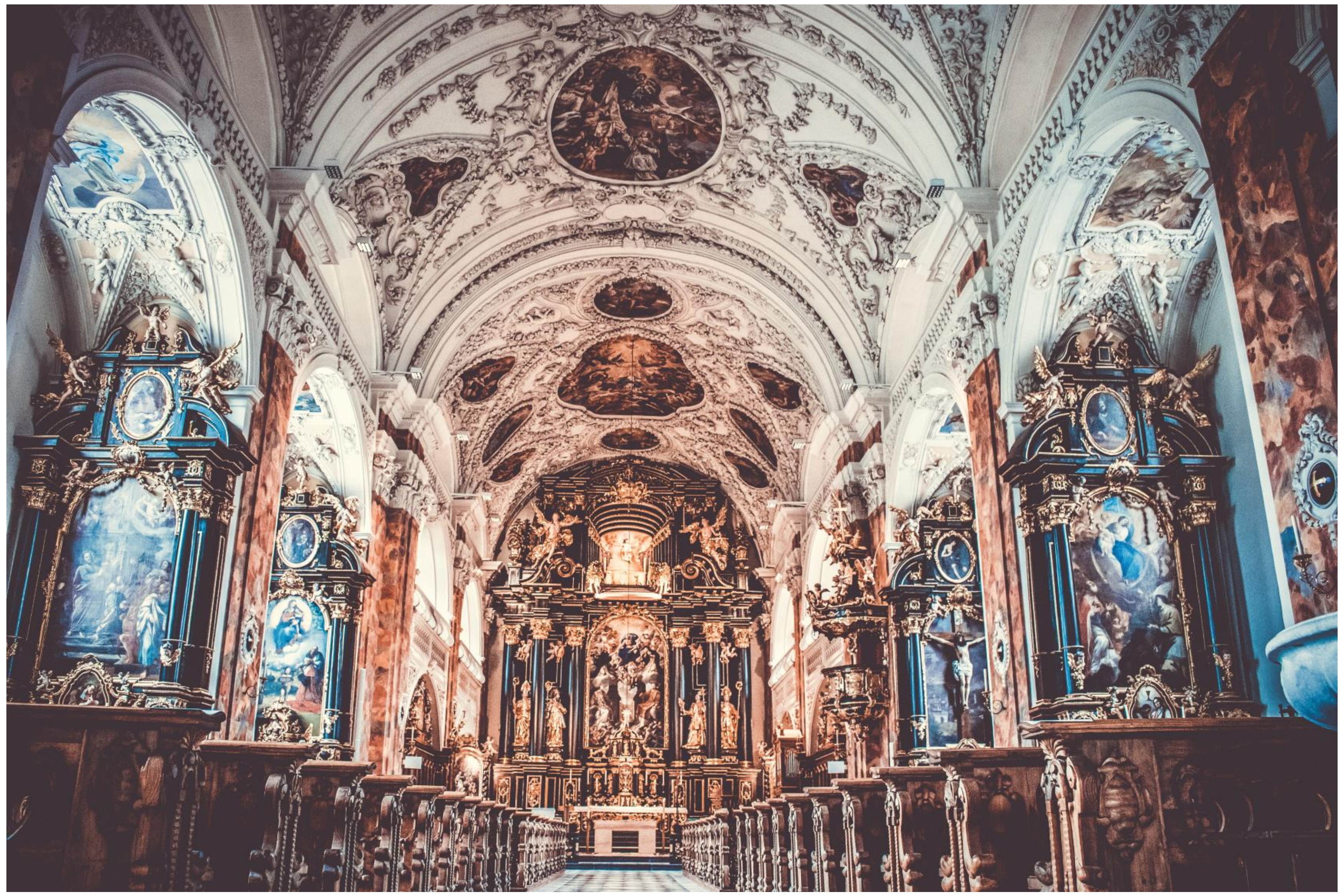
Execute it manually

Leverage our simple build

To execute it automatically

Getting Eyes on It

The Cathedral and the Bazaar



A closed system with a priesthood

An open system where anyone can contribute

Given enough eyeballs, all bugs are shallow.

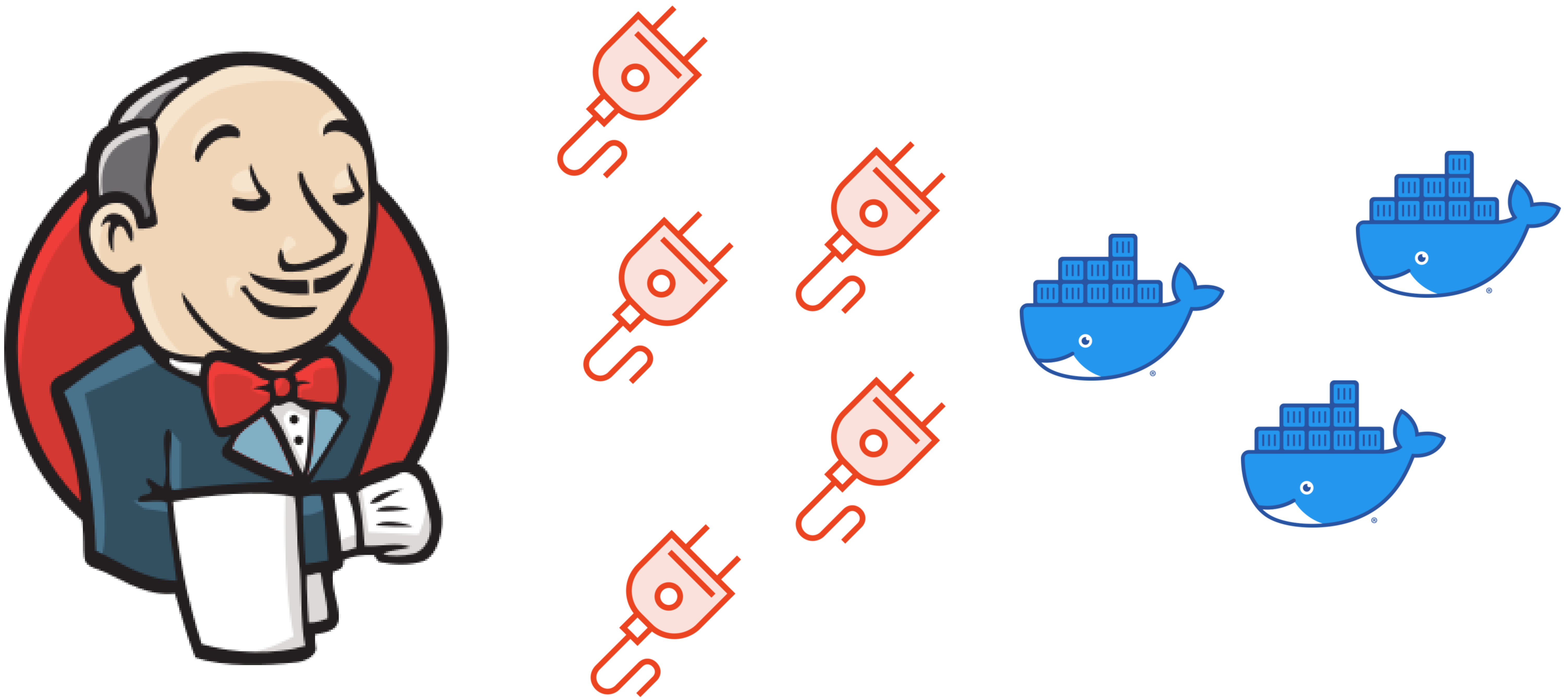
How the Bazaar Works

The bazaar makes YOU more careful

Inspection shifts the defect left

It puts the power in the hands of the users

My Open Source Story



A Bad Plug-In

A plug-in for managing Docker containers

Automatic upgrades for minor versions

Minor versions, by definition, are backward-compatible

A dependency of my dependency was broken

I pulled up the code on Github and found the problem

The developer fixed the problem in a few hours

We want as many eyes on our code as possible

This doesn't happen unless you make it happen

Eyes as a First-class Artifact

An artifact that WON'T be dropped under pressure

Everything else gets dropped when the schedule pressure hits

Version control is an example of a first-class artifact

Let your build save you from a bad deployment

A Version Control Process for Eyes on Code

Pull Request (PR) review

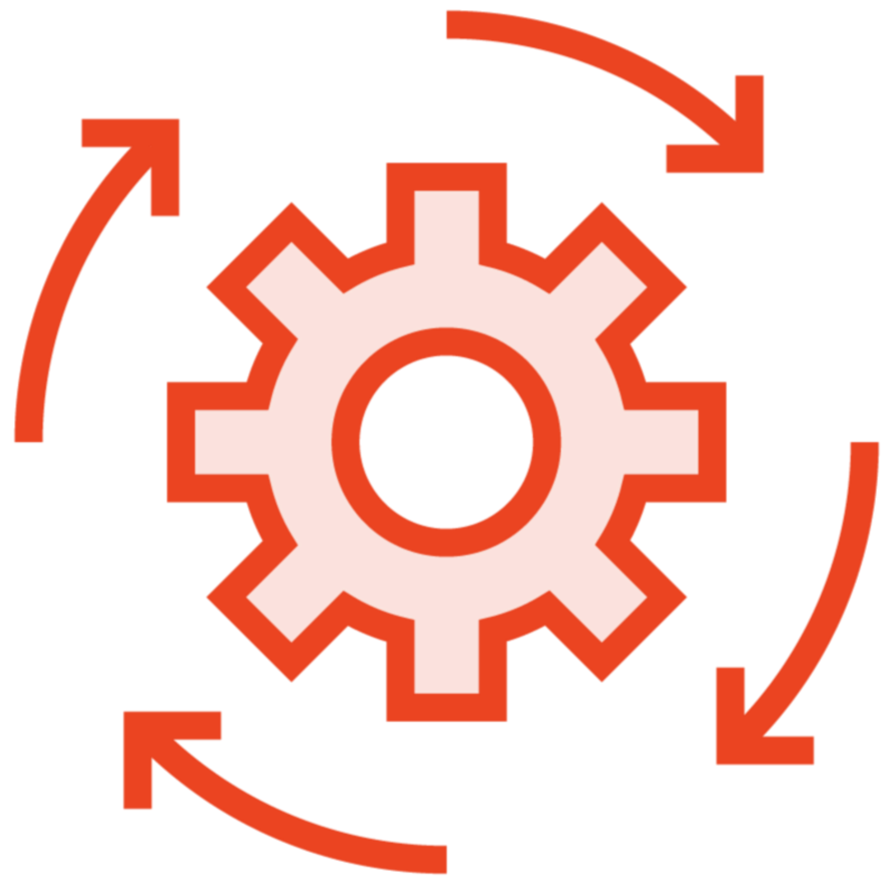
What's a pull?

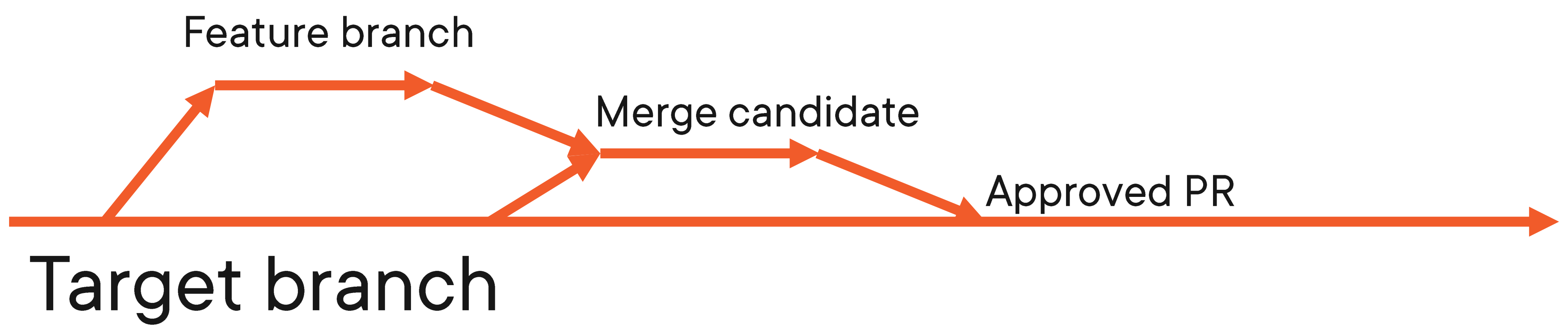
The process of merging code back to the main branch

The main branch is locked against direct merges

So, merges only happen as a part of the PR process

Pull Request Builds





Computer eyes are not enough

If for no other reason, because they cannot truly verify correctness

Human Eyes on a Pull Request

Senior developer eyes on all PRs

Ideally, this is their only job

The build checks it first to make sure that it’s a structurally valid PR

Then a human mind reviews the code for intent, correctness and conforming to the requirement

And iterates with the developer to get it approved

Without a dedicated PR Reviewer, the end of sprint crunches the review

Yet Another Kind of Eyes

This doesn’t validate correctness

But other stuff than correctness matters

Static analysis

A failed analysis can break the build (a good thing)

Developer can execute the scanner locally

They understand the problem so they don’t create it next time

https://www.pluralsight.com/courses/microsoft-devops-solutions-designing-build-automation

The Last Kind of Eyes

Open source

```
attachEvent("onreadystatechange",H),e.attachE
oolean Number String Function Array Date RegE
_={};function F(e){var t=_[e]={};return b.ea
t[1])===!1&&e.stopOnFalse){r=!1;break}n=!1,u&
?o=u.length:r&&(s=t,c(r))}return this},remove
nction(){return u=[],this},disable:function()
re:function(){return p.fireWith(this,argument
ending",r={state:function(){return n},always:
romise)?e.promise().done(n.resolve).fail(n.re
dd(function(){n=s},t[1^e][2].disable,t[2][2].
=0,n=h.call(arguments),r=n.length,i=1!==r||e&
(r),l=Array(r);r>t;t++)n[t]&&b.isFunction(n[t
/><table></table><a href='/a'>a</a><input typ
/TagName("input")[0],r.style.cssText="top:1px
est(r.getAttribute("style")),hrefNormalized:
```

This may not be possible for IP reasons

But be SURE that the code is the business

Because it may be something else

The Big Win: Automated Deployment

If the idea of automating your deployment seems impossible, that is the project that needs it the most.

Ramping Things Up

More developers, deployment more often

Don’t let the perfect be the enemy of the good

Use manual steps for the time being

Can I automate my deployment?

Can I automate any PART of my deployment?

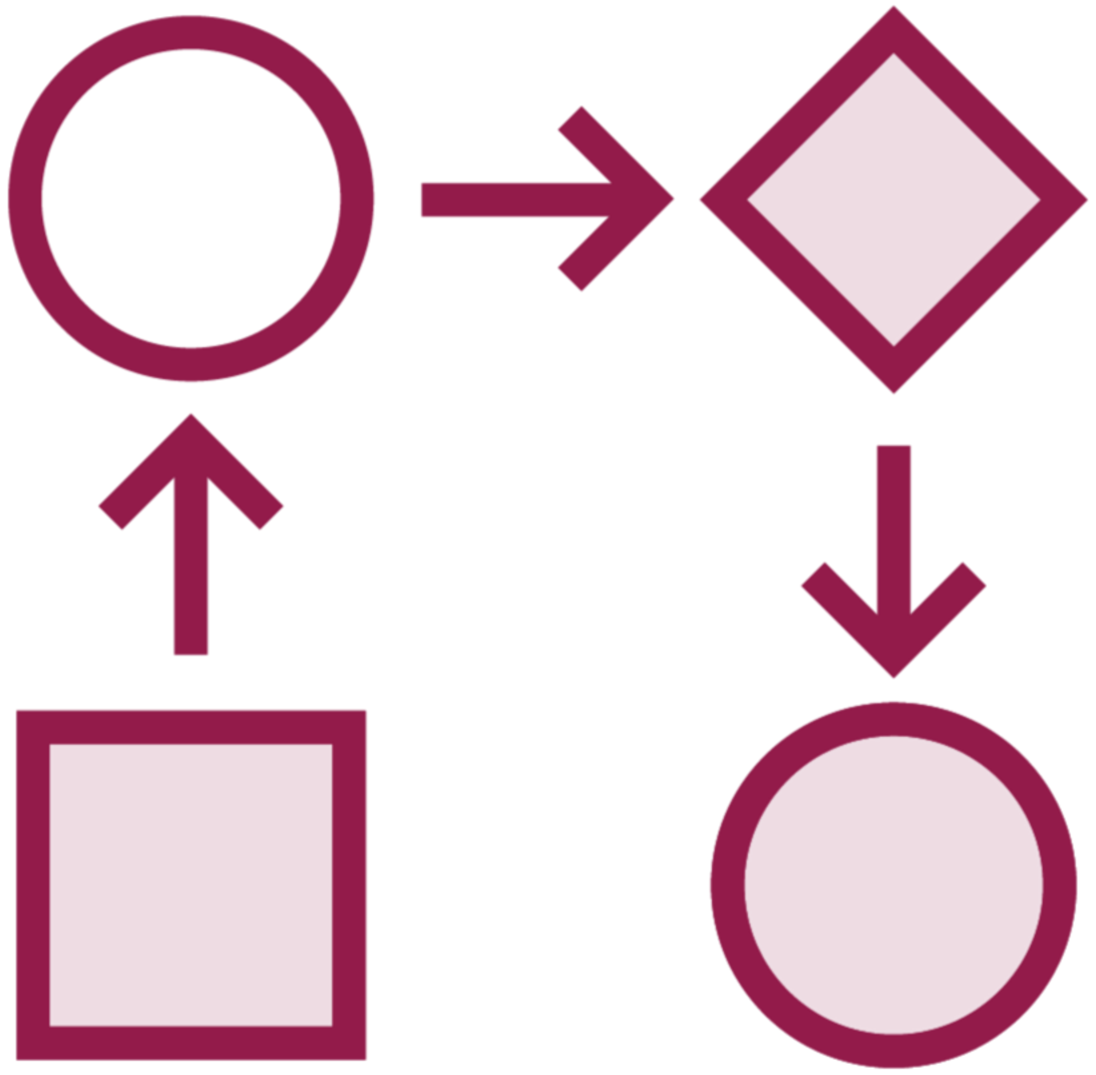
The Virtuous Path for Pre-production Deployment

Automated provision of test environments

Allowing for human feedback

Deploy to Staging first

A Workflow



Somebody opens a work ticket

Developer branches from deployment-bound branch

Developer writes unit tests

Developer creates a Pull Request

PR Build succeeds

PR reviewer iterates with developer until approval

System merges the feature branch with the main

This triggers the provisioning and deployment of a verification environment

The ticket is marked as in review, and the stakeholder is notified

Sounds Complicated

It takes work, but it's possible

Some manual intervention may be needed

The Certainty Chain

The developer is certain because of his unit tests

The PR reviewer is certain because of the build and his review

The stakeholder is certain because they reviewed it

The product development cycle is the process of constructing certainty.

Azure Hosting and Automated Deployment

Let’s shift to a cloud-hosted scenario

Our Production resources are now in the cloud instead of our own data center

We largely get Infrastructure as Code for free

And we can scale OUT instead of UP with parallel instances of the resources

And we can take advantage of the pathway that the designers have anticipated

If I were starting from scratch, I’d use more-difficult-to-use tools that gave me more flexibility

But this path is VERY easy to learn

Demo

Add a deployment cycle to our process

Make a simple change to our code

We can verify that a deployment happened

When we see it on our Azure website

Automated Deployment Wrap-up

All DevOps is a combination of science and lore

You want to maximize the science and minimize the lore

The lore was the publish and artifact drop

Don’t be discouraged if you run into fiddly bits

What if I'm Not Using Azure?

ADO can push to other deployment targets

Other deployment systems can push to Azure

https://app.pluralsight.com/library/courses/automating-jenkins-groovy

https://app.pluralsight.com/library/courses/octopus-deploy-getting-started

The Paradox of DataOps

Consistent and changing

Consistent with the applications they serve…

But changing along with those applications

Resolving the Paradox

Infrastructure as Code? Rebuild the database every time?

Nope

To horizontally scale the database…

You need something that regresses to the transaction logs of the db

SECRETS DO NOT BELONG IN VERSION CONTROL

We need two things:

- A known state in the target db
- A script to migrate us to the new state

Database Deployment in a Nutshell

We add the new script to the sum of all previous scripts

Then, an engine executes the scripts that haven't already been executed on the target db

And then executes the new script to get to the new state

https://app.pluralsight.com/library/courses/microsoft-azure-web-applications-services-deploying

https://app.pluralsight.com/library/courses/sql-server-databases-docker-developing

Summary

Creating Knowledge

Accumulating evidence

Building certainty

Automated unit testing

Static analysis

Automated deployment