



Bryan Finster



Defense
Unicorns



ENTERPRISE TRANSFORMATION

“We need an agile mindset!”

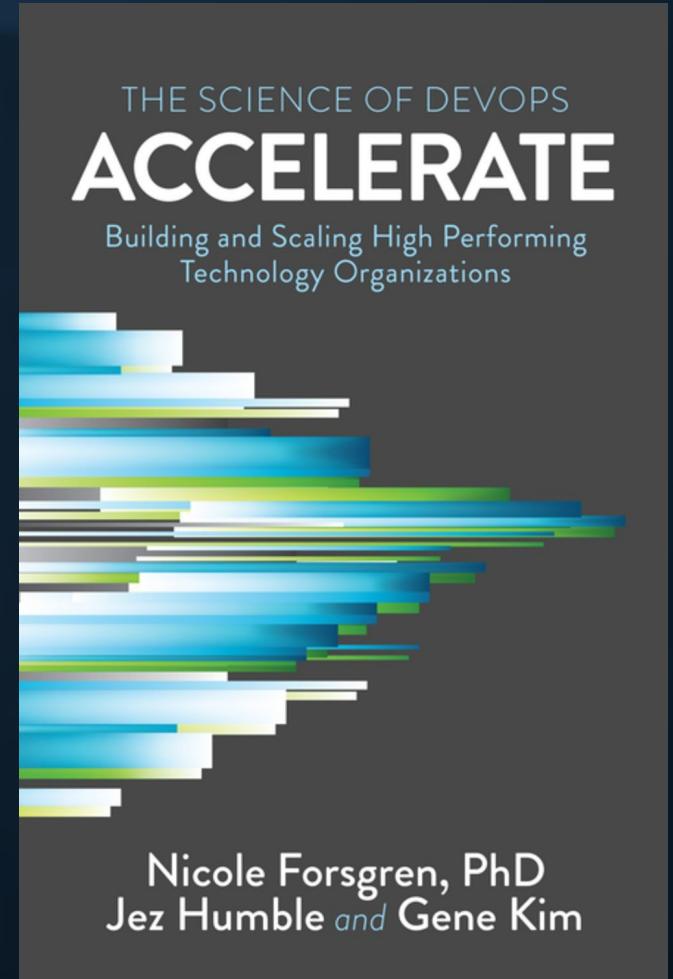
“We need to improve Scrum!”

“We need a scaling framework to standardize process!”

We need better outcomes for ourselves and our customers

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

”



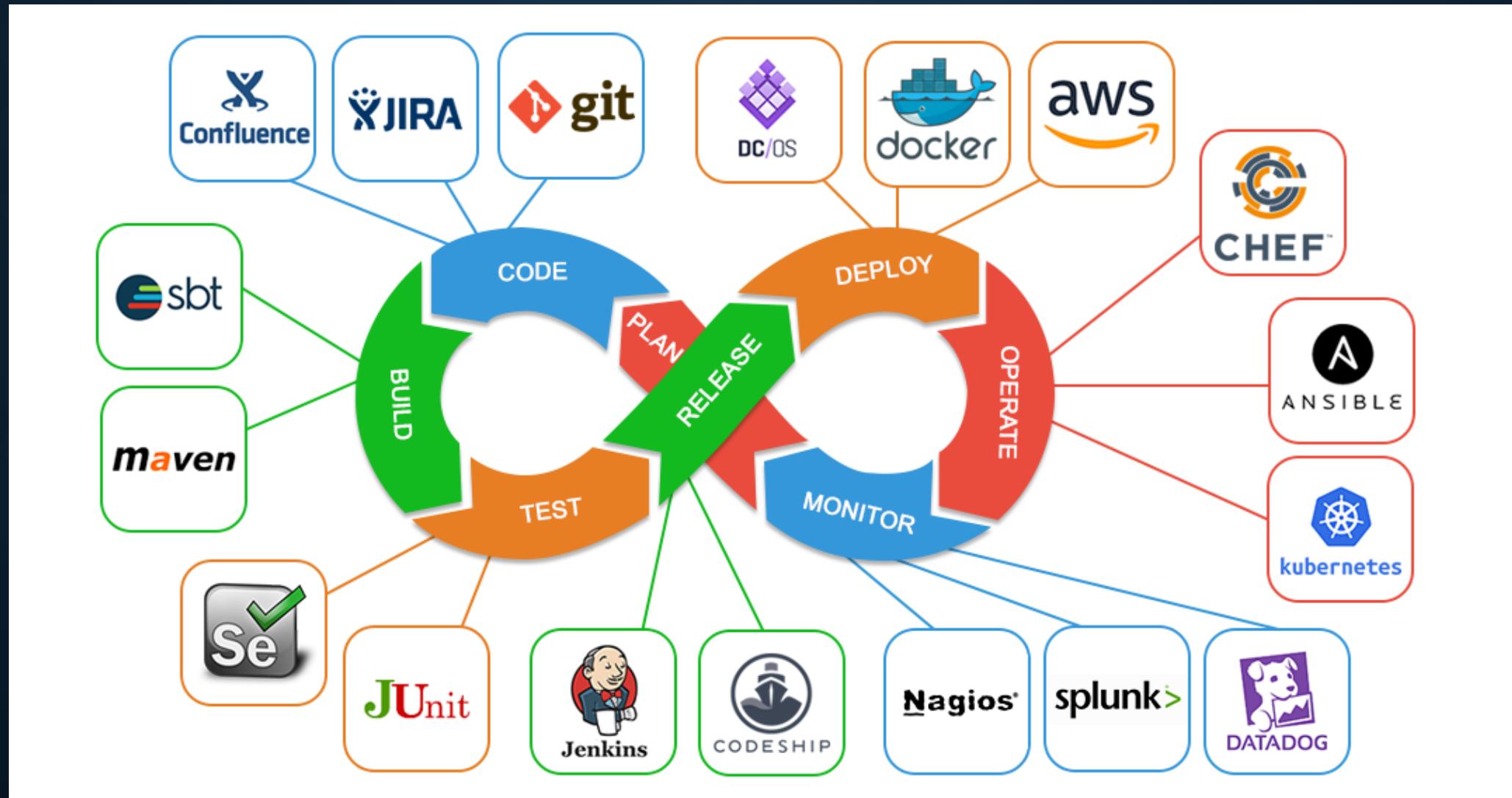
We need CD!

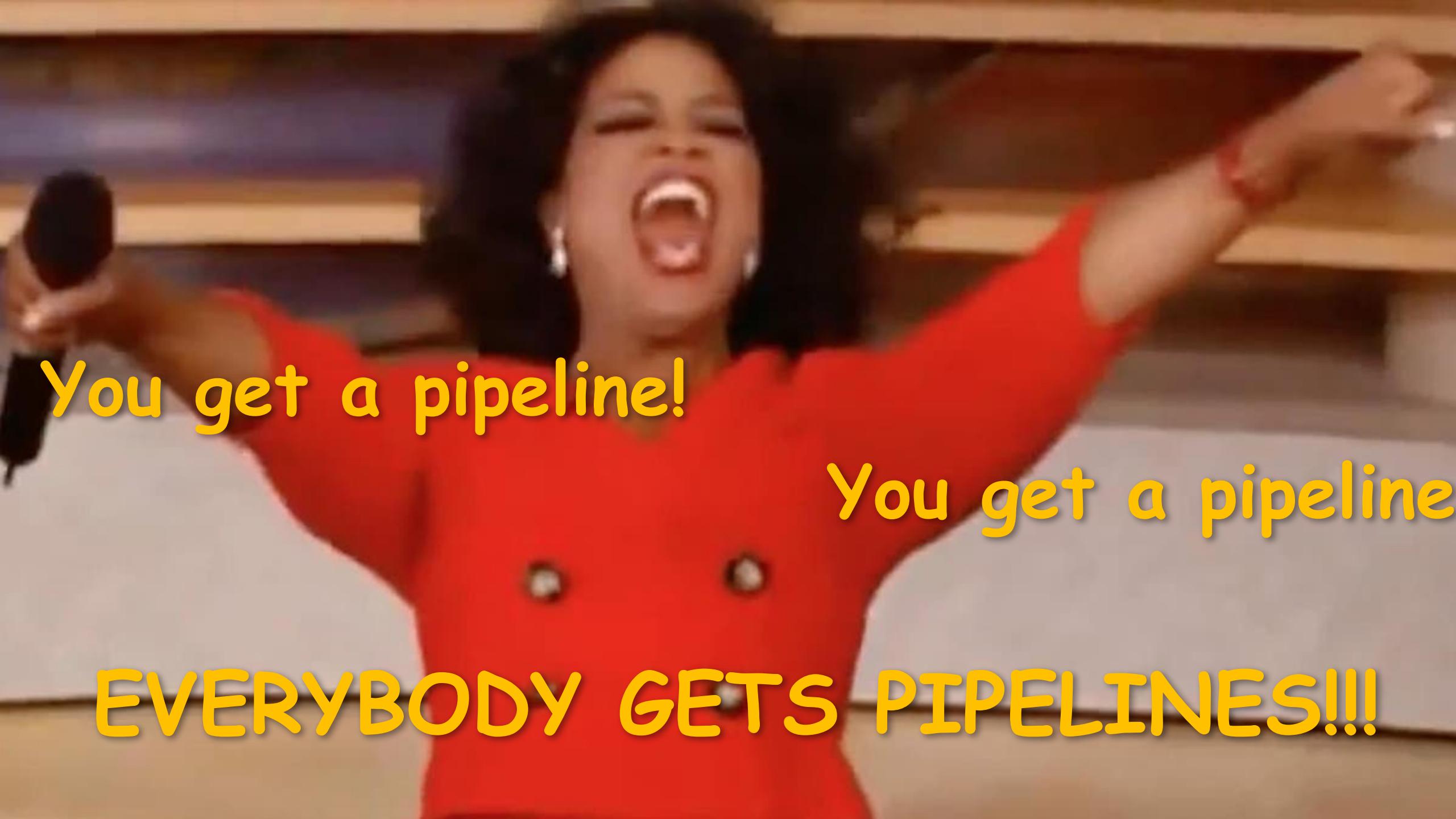
Do CD!

And the problems begin...

CREDIT CARD CONTINUOUS DELIVERY

BUY ALL THE TOOLS





You get a pipeline!

You get a pipeline

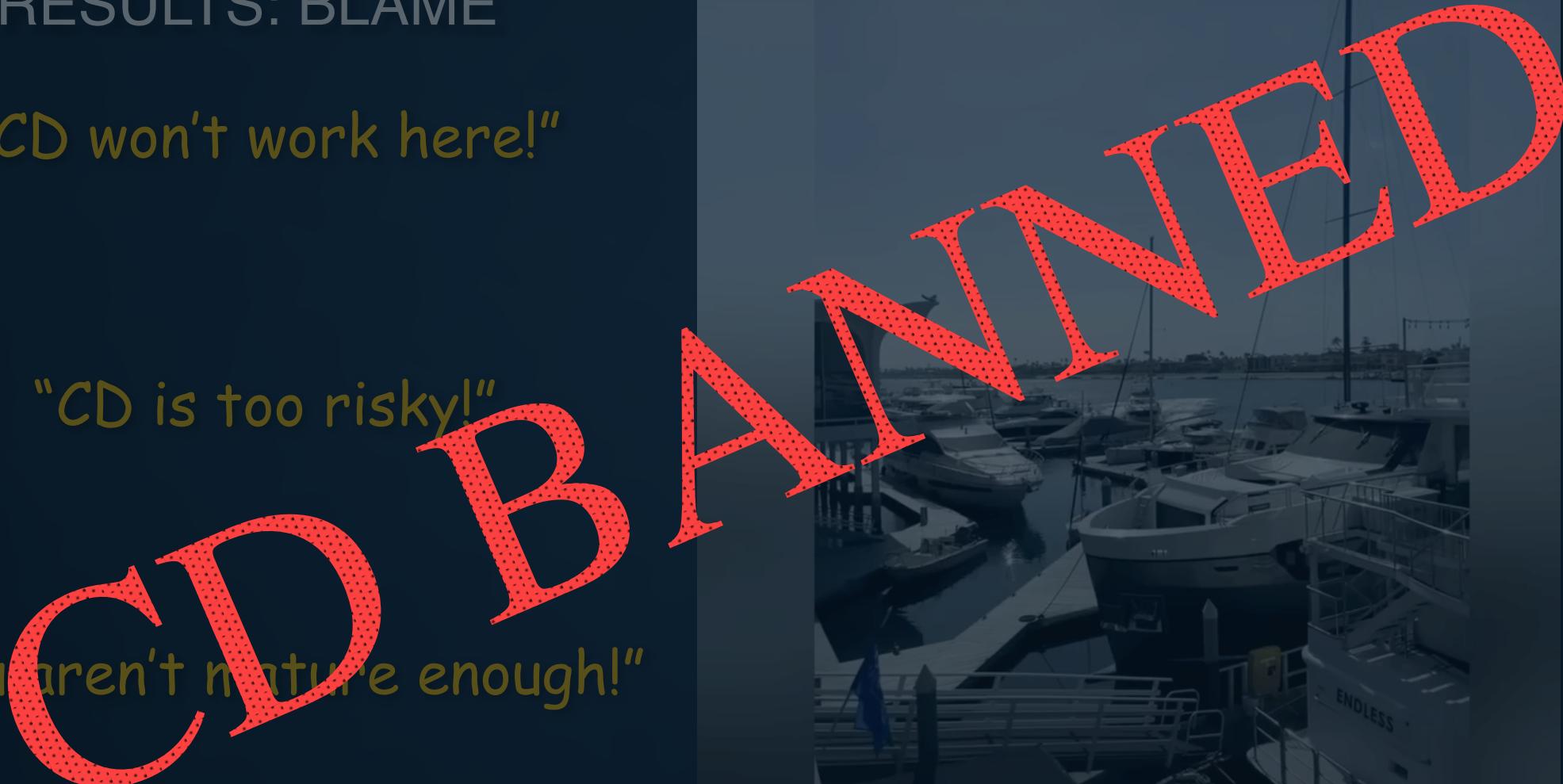
EVERYBODY GETS PIPELINES!!!

RESULTS: BLAME

"CD won't work here!"

"CD is too risky!"

"You aren't mature enough!"



**"CD MEANS WE
DELIVER ON-DEMAND..."**

...SO WE DELIVER ON DEMAND ONCE PER MONTH”

“Our customers don’t want changes more frequently.”



RESULTS: AUTOMATED STAGNATION

"We automated delivery. Why aren't we seeing the promised improvements?"

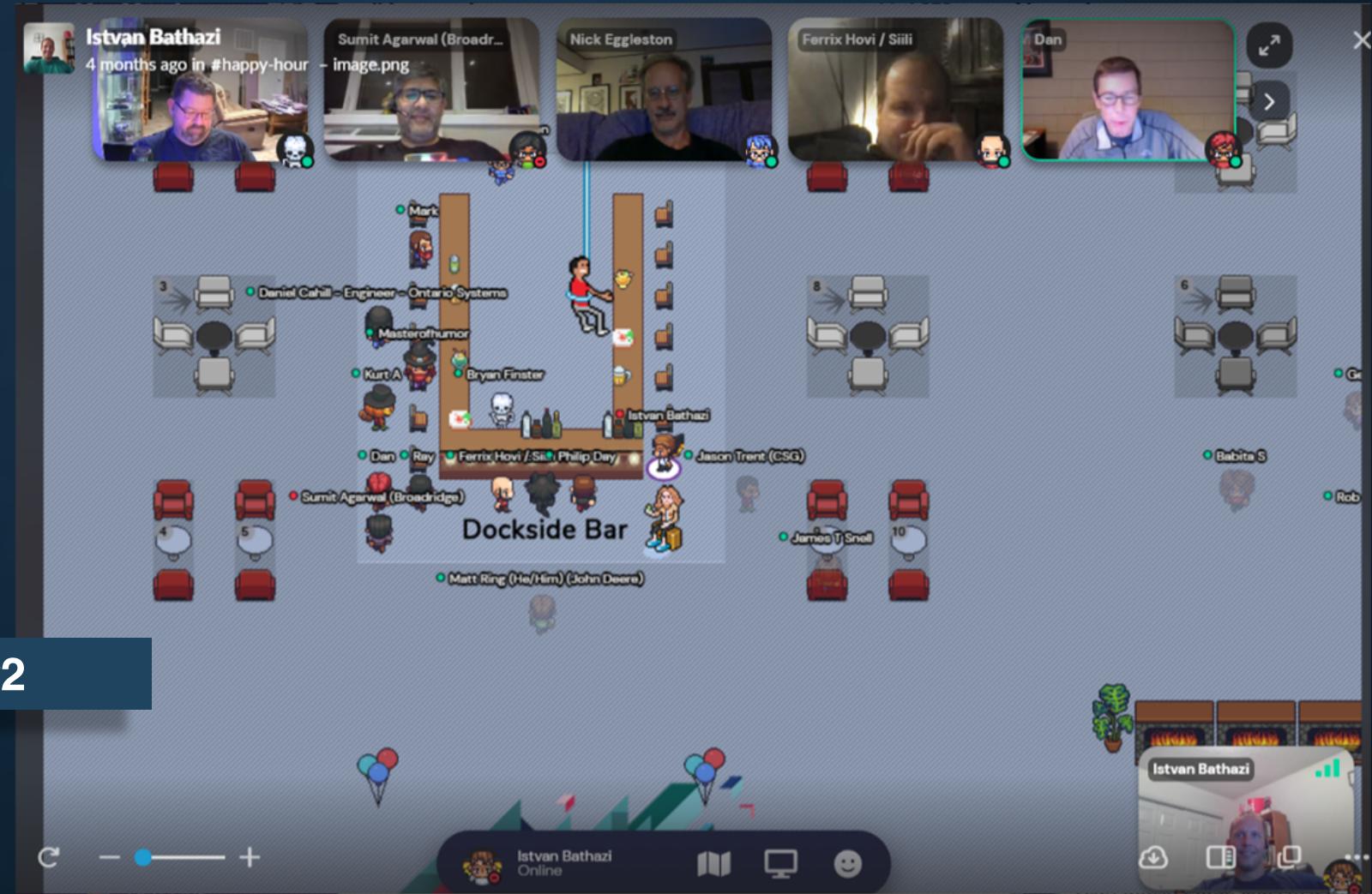
- Behaviors stayed the same
- Batch sizes stayed large
- Feedback remained slow
- Quality process did not improve
- Communication did not improve

DOCKSIDE BAR, OCT 2021

Two options:

1. Gripe about it
2. Help fix it

We chose #2



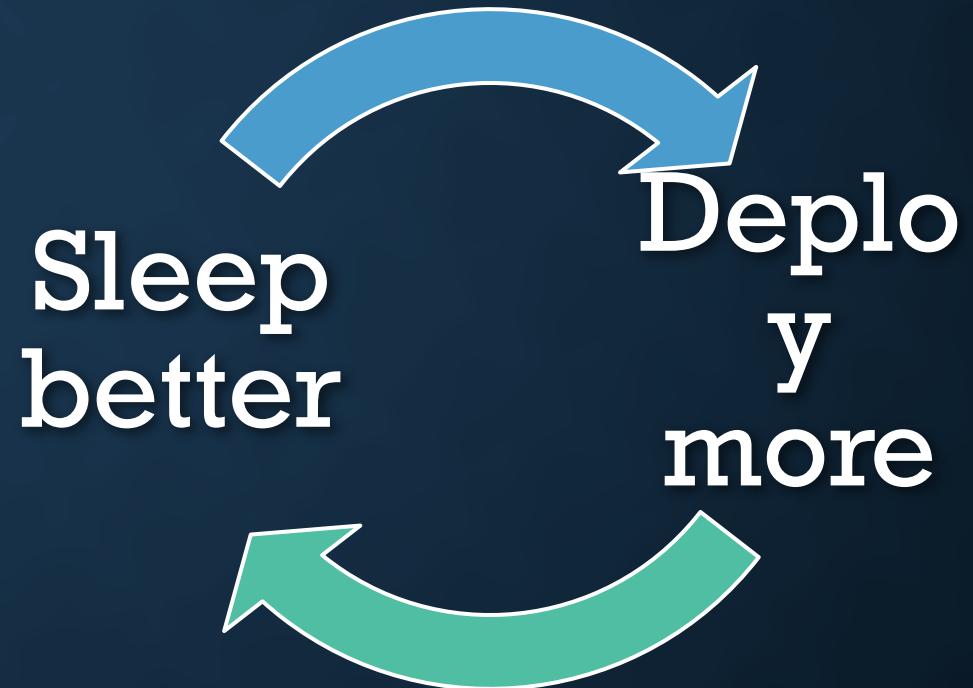
OUR GOAL



- ✓ Minimum set of measurable behaviors
- ✓ True in every context
- ✓ Enabling constraints that reduce pain and improve outcomes

WHY DO WE CARE?

Everywhere
should be
an awesome
place to work!



3 DAYS LATER...

The screenshot shows the homepage of MinimumCD.org. The header includes the site name "MinimumCD.org" and a "Github" link. A search bar says "Type to search...". The main navigation menu on the left lists "Minimum Viable CD", "Beyond the Minimums", "Recommended Practices", "Experience Reports", "Frequent Questions", and "Translations". The "Recommended Practices" item has a blue arrow icon. The "Minimum Viable CD" page title is "Minimum Viable CD". Below it are links "Signatories" and "Jump to Section >". A quote box contains the text: "Continuous delivery improves both delivery performance and quality, and also helps improve culture and reduce burnout and deployment pain." followed by a reference to "Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations". The main content area discusses the undersigned's belief in the benefits of continuous delivery and lists three actions: introducing new practitioners, discussing engineering practices, and helping each other improve capabilities. It concludes with the statement that only by implementing core practices can we begin to see the benefits of continuous delivery.

MinimumCD.org

Github

Type to search...

Minimum Viable CD

Beyond the Minimums

Recommended Practices

Experience Reports

Frequent Questions

Translations

Minimum Viable CD

↳ Signatories

≡ Jump to Section >

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

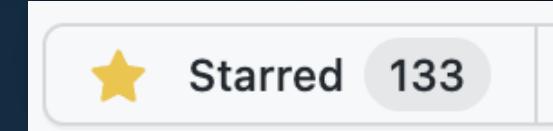
– Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations

We, the undersigned, believe that a minimal definition of continuous delivery (CD) is required to improve the flow of delivery and achieve the outcomes above. While our contexts may be different, there are universal practices common in all. By defining them we can:

- Introduce new practitioners in a consistent way
- Discuss engineering practices that comprise CD
- Help each other improve current capabilities

Only by implementing core practices do we begin to see the benefits of continuous delivery.

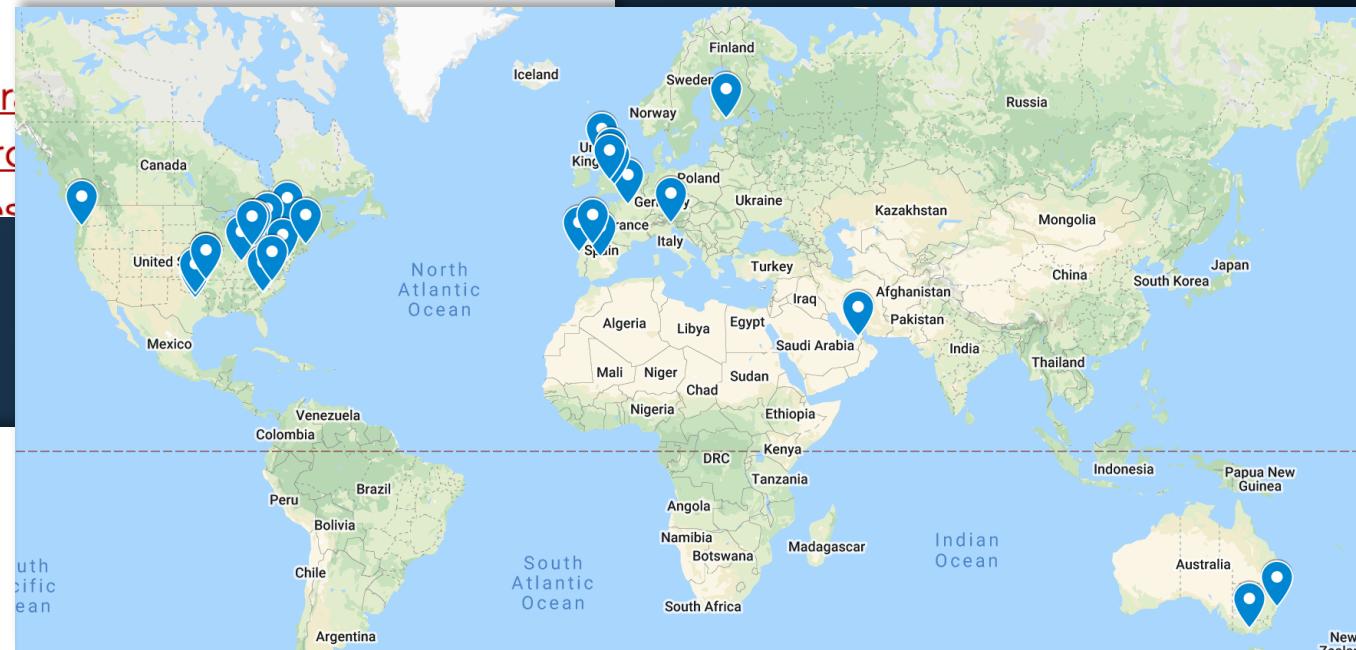
TODAY...



Contributors

(39)

[Azlam Abdulsalam](#) · [Justin Abrahms](#) · [Austin Abro](#) · [Gr](#)
[Istvan Bathazi](#) · [Kaine Bent](#) · [Marc Boudreau](#) · [Kelly Br](#)
[Daniel Calle](#) · [Patrice Corbard](#) · [Jeff Dunn](#) · [Nick Eagles](#)



Signatories

(124)

[Dave Farley](#) · [Bryan Finster](#) · [Ferrix Hovi](#) · [Justin Abrahms](#) · [Joe Arrowood](#) ·
[Jerreck McWilliams](#) · [Istvan Bathazi](#) · [Sara Gramling](#) · [Tracy Bannon](#) · [Dana Finster](#)
· [Patrick S Kelso](#) · [Ben Link](#) · [Chris Kernaghan](#) · [Chris Gossett](#) · [Joshua Barton](#) ·

CONSTRAINTS THAT DRIVE IMPROVEMENT

- Use Continuous integration
- The application pipeline is the only way to deploy to any environment.
- The pipeline decides the releasability of changes, its verdict is definitive
- Artifacts created by the pipeline always meet the organization's definition of deployable
- Immutable artifact. No human changes after commit.
- All feature work stops when the pipeline is red
- Production-like test environment
- Rollback on-demand
- Application configuration deploys with artifact

CONTINUOUS INTEGRATION

- ✓ Tested changes integrate into the trunk *at least* daily
- ✓ Changes are, to the best of our knowledge, deliverable
- ✓ Fixing a broken build is the highest priority



“Why can’t we do this?”

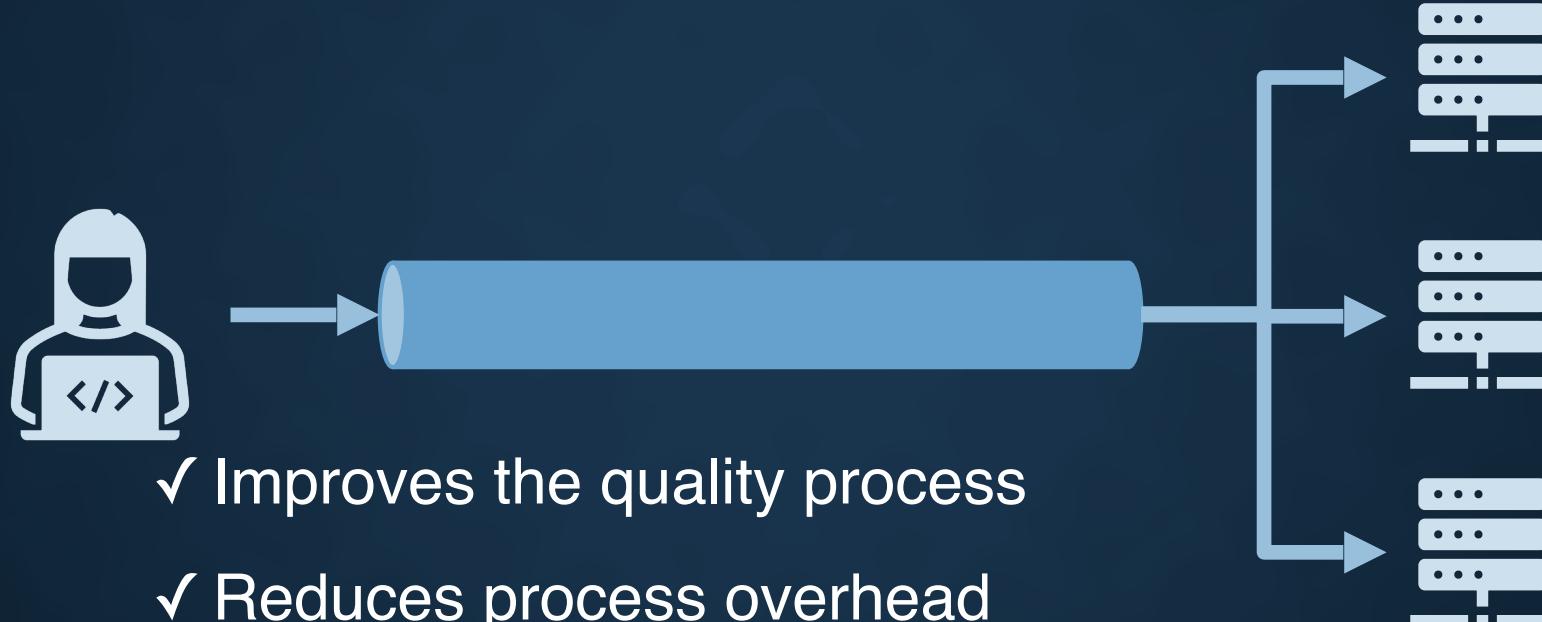
CI UNCOVERS COMMON PROBLEMS WITH...

- Communication
- Work decomposition
- Evolutionary development
- Testing
- Teamwork



**Overcoming these
makes things suck less**

ONE PATH TO ANY ENVIRONMENT



- ✓ Improves the quality process
- ✓ Reduces process overhead
- ✓ Reduces handoffs
- ✓ Ensures we can safely resolve incidents

Test the process, not just the code

REMOVE DRAMA FROM DELIVERY



**The secret of CD:
It's always an emergency**

THE PIPELINE DECIDES “RELEASEABLE”

✓ Requires that we objectively define
“releasable” and automate it.

- Functional
- Performant
- Secure
- Compliant
- Etc.



	Base	Beginner	Intermediate	Advanced	Expert
Culture & Organization	<ul style="list-style-type: none"> Prioritized work Defined and documented process Frequent commits 	<ul style="list-style-type: none"> One backlog per team Share the pain Stable teams Adopt basic Agile methods Remove boundary dev & test 	<ul style="list-style-type: none"> Extended team collaboration Component ownership Act on metrics Remove boundary dev & ops Common process for all changes Decentralize decisions 	<ul style="list-style-type: none"> Dedicated tools team Team responsible all the way to prod Deploy disconnected from Release Continuous improvement (Kaizen) 	<ul style="list-style-type: none"> Cross functional teams No rollbacks (always roll forward)
Design & Architecture	<ul style="list-style-type: none"> Consolidated platform & technology 	<ul style="list-style-type: none"> Organize system into modules API management Library management Version control DB changes 	<ul style="list-style-type: none"> No (or minimal) branching Branch by abstraction Configuration as code Feature hiding Making components out of modules 	<ul style="list-style-type: none"> Full component based architecture Push business metrics 	<ul style="list-style-type: none"> Infrastructure as code
Build & Deploy	<ul style="list-style-type: none"> Versioned code base Scripted builds Basic scheduled builds (CI) Dedicated build server Documented manual deploys Some deployment scripts etc. 				<ul style="list-style-type: none"> Build bakery Zero touch continuous deployments
Test & Verification	<ul style="list-style-type: none"> Automatic unit tests Separate test environment 		<ul style="list-style-type: none"> Some automatic acceptance tests 	<ul style="list-style-type: none"> Automatic performance tests Automatic security tests Risk based manual testing 	<ul style="list-style-type: none"> Verify expected business value
Information & Reporting	<ul style="list-style-type: none"> Baseline process metrics Manual reporting 	<ul style="list-style-type: none"> Measure the process Static code analysis Scheduled quality reports 	<ul style="list-style-type: none"> Common information model Traceability built into pipeline Report history is available 	<ul style="list-style-type: none"> Graphing as a service Dynamic test coverage analysis Report trend analysis 	<ul style="list-style-type: none"> Dynamic graphing and dashboards Cross silo analysis

Another maturity model?

Gatekeeping?

No

An improvement journey



Measurable objectives that uncover systemic problems



WHERE TO START?

“Why can’t we deliver today’s changes today?”

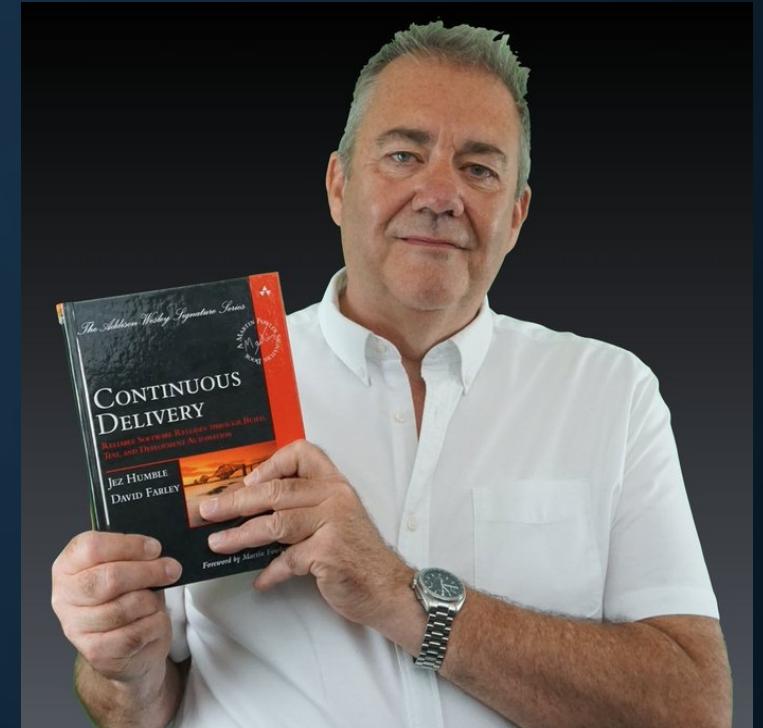
ROADMAP

- ✓ Define “releasable” in your context
- ✓ Fall passionately in love with testing
- ✓ Start with continuous integration
- ✓ Start automating manual validation of “releasable”
- ✓ Improve team structure to remove handoffs
- ✓ Relentlessly improve

“
It's a clear, focused, no-holds-barred statement of what it takes to achieve CD.

If you can do what Minimum CD says, you will be doing a better job. It gives us a clear, simple focus on the essentials of CD that can help teams to understand what really matters to build better software faster.

”



Dave Farley, Author: Continuous Delivery

WHAT HELP DO WE NEED?

Contributing

Do you want to submit translations, good practices, suggestions, or an experience report?

Read our [contribution guidelines](#) -.

- Translations
- Good practices “Beyond the Minimums”
- Experience reports
- Spreading the word

Translations

Español

Français

Italiano

Português

Suomi

MINIMUMCD.ORG



/bryan-finster



@BryanFinster



blog.bryanfinster.com

