

# Software Engineering and Architecture Continuous Delivery

---

Olivier Liechti  
HEIG-VD  
[olivier.liechti@heig-vd.ch](mailto:olivier.liechti@heig-vd.ch)



MASTER OF SCIENCE  
IN ENGINEERING



*"Here is a USB key with the .war file that you need to deploy on Tomcat. I have written down the configuration steps on this sheet of paper..."*



*“WTF... why do I get a database connection error when I  
deploy this damn application in the Glassfish cluster...???”*

Worried  
Ops guy



Worried  
Dev guy

*"We have deployed the new release... are we **sure** that we are ok?  
By the way, anyone has a clue about what is in the release?"*

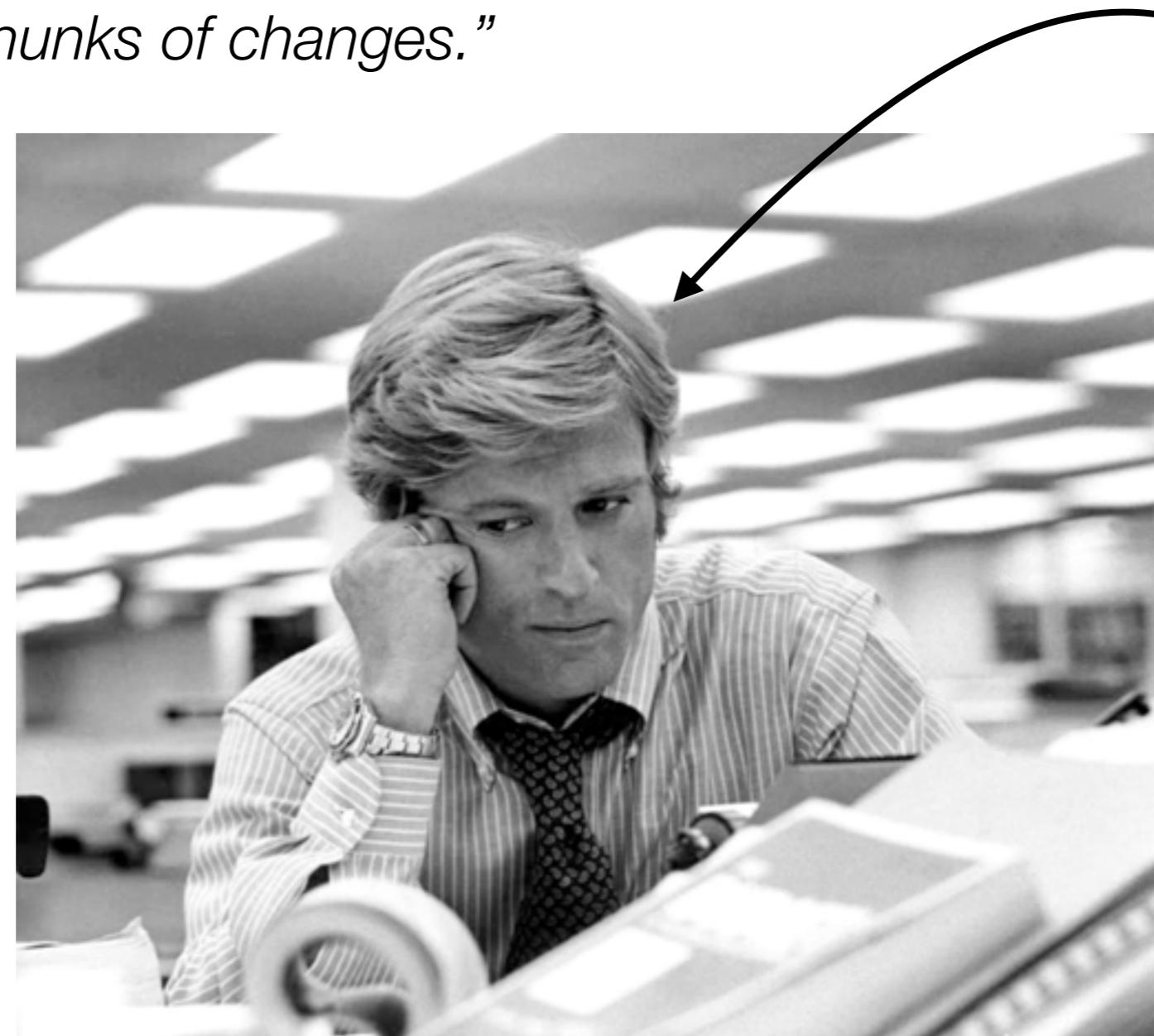


*Robert Redford*

*"I hate these production rollouts... it's always so **stressful**.  
What could we do to avoid bad surprises and not be so  
**afraid to touch the prod...**?"*

*“Releasing **small chunks** of changes is less scary than releasing big chunks of changes.”*

*“Something that you do **often** is less scary than something you do rarely.”*



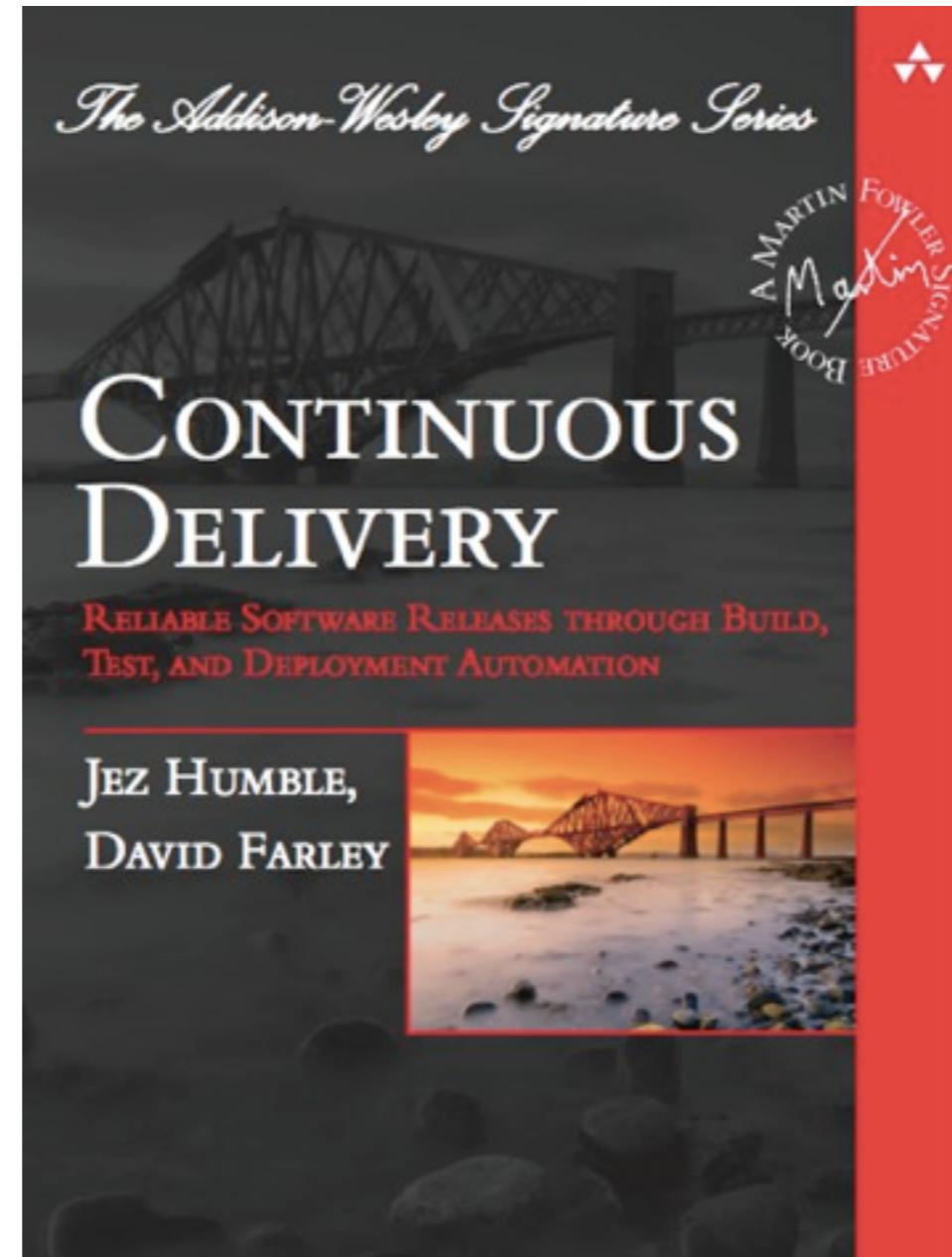
*Agile  
Robert Redford*

*“If something is **automated**, there is now worry that I can make a wrong manipulation.”*

*“If I have **evidence** that the system has been **fully tested**, at different levels, I feel secure.”*

# Continuous Delivery

---



*The slides are based on this book (ISBN:)*  
Free chapter: <http://www.informit.com/articles/article.aspx?p=1621865>

# Foreword, by Martin Fowler (1)

---

***“Continuous Integration*** is just the first step.

*Software that's been successfully integrated into a mainline code stream still isn't software that's out in production doing its job.*

*Dave and Jez's book pick up the story from CI to **deal with that “last mile”** describing **how to build the deployment pipeline that turns integrated code into production software.**”*

## Foreword, by Martin Fowler (2)

---

*“This kind of delivery thinking has long been a forgotten corner of software development, falling into **a hole between developers and operations teams.***

*So it’s no surprise that the techniques in this book rest upon bringing these teams together—a harbinger of the nascent but growing **DevOps** movement.*

*This process also involves **testers**, as testing is a key element of ensuring error-free releases.”*

# The Problem of Delivering Software

- How do we transform an **idea** into **something usable by end-users** as **quickly** as possible?
- We need an efficient **software development process**, but that is not enough.
- We need to cover the **complete delivery life-cycle**, until the release into the production environment.



# Release Anti-Patterns

---

- **Anti-Pattern:** Deploying Software Manually
- **Anti-Pattern:** Deploying to a Production-like Environment Only after Development Is Complete
- **Anti-Pattern:** Manual Configuration Management of Production Environments

# Principles of Software Delivery

---

1. Create a **Repeatable, Reliable Process** for Releasing Software
2. **Automate Almost Everything**
3. Keep Everything in **Version Control**
4. **If It Hurts, Do It More Frequently**, and Bring the Pain Forward
5. Build **Quality In**
6. “**Done**” Means Released
7. **Everybody Is Responsible for the Delivery Process**
8. **Continuous Improvement**

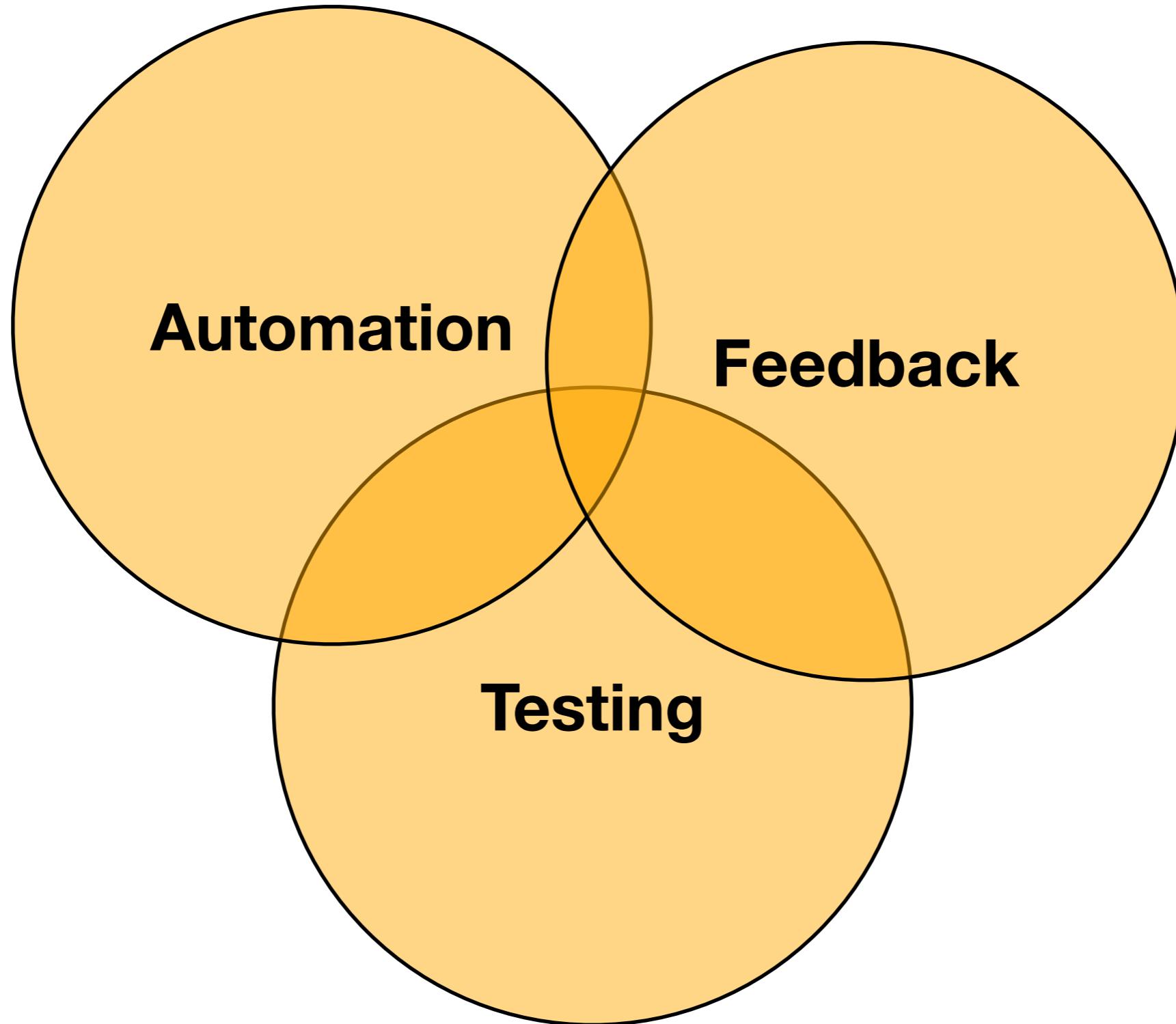
# Frequent and Automated Releases

---

- We, and our fellow practitioners, have discovered that in order to achieve these goals—low cycle time and high quality—**we need to make frequent, automated releases of our software**. Why is this?
- **Automated.** If the build, deploy, test, and release process is not automated, it is not repeatable. Every time it is done, it will be different, because of changes in the software, the configuration of the system, the environments, and the release process. Since the steps are manual, they are error-prone, and there is no way to review exactly what was done. This means there is no way to gain control over the release process, and hence to ensure high quality. **Releasing software is too often an art; it should be an engineering discipline.**
- **Frequent.** If releases are frequent, the delta between releases will be small. This significantly reduces the risk associated with releasing and makes it much easier to roll back. Frequent releases also lead to faster feedback—indeed, they require it. Much of this book concentrates on getting feedback on changes to your application and its associated configuration (including its environment, deployment process, and data) as quickly as possible.

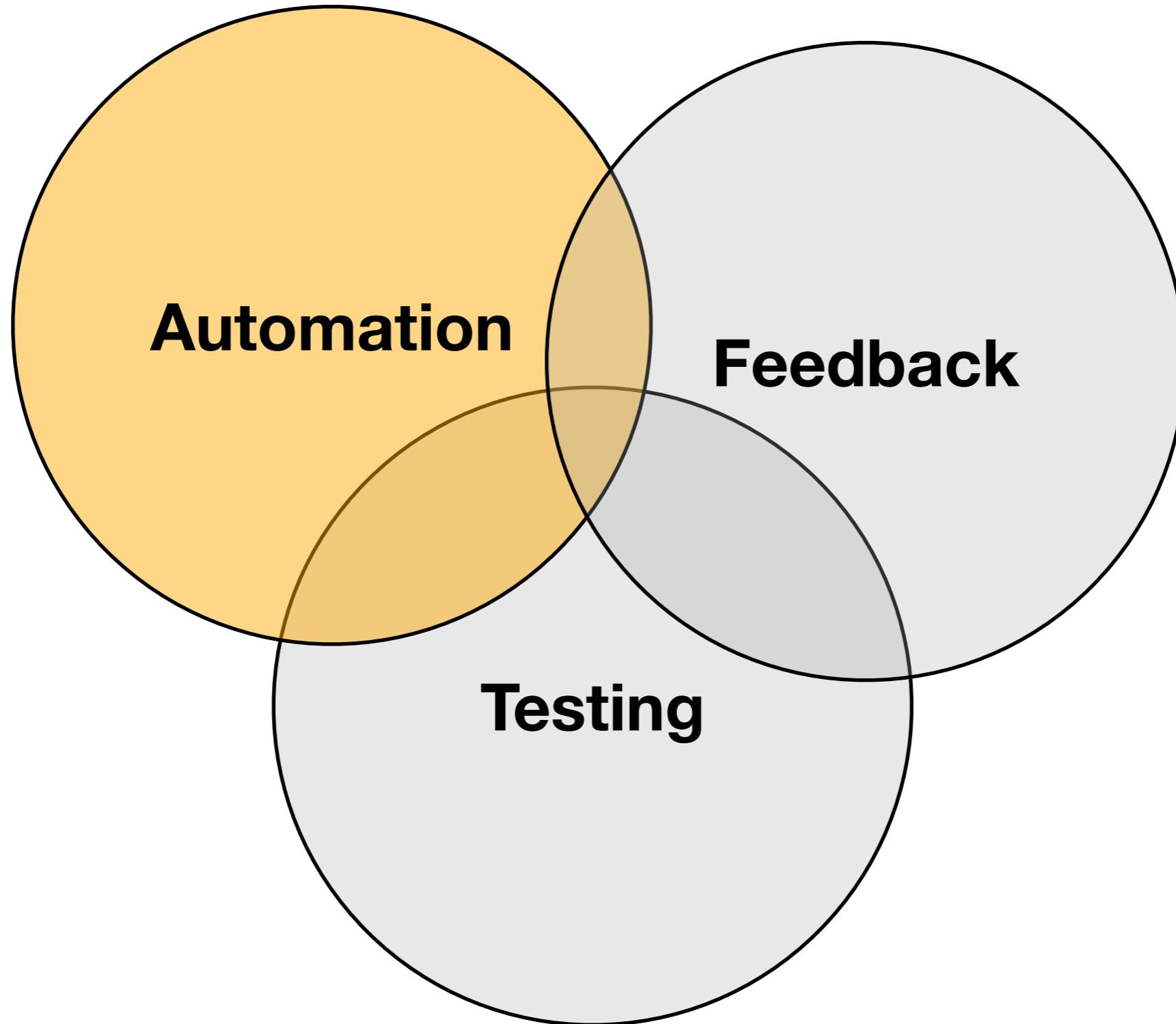
# The Pillars of Continuous Delivery

---



# The Pillars of Continuous Delivery

---



# The Deployment Pipeline

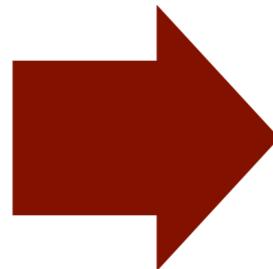
---

*“The deployment pipeline has its foundations in the process of continuous integration and is in essence **the principle of continuous integration taken to its logical conclusion.**”*





***Continuous Integration***



Kanpai-Japan.com

# *Continuous Delivery*

# The Deployment Pipeline



Figure 1.1 *The deployment pipeline*

# The Deployment Pipeline

---

*“Every change that is made to an application’s configuration, source code, environment, or data, **triggers the creation of a new instance of the pipeline.***

*One of the first steps in the pipeline is to **create binaries and installers.***

*The rest of the pipeline **runs a series of tests** on the binaries to prove that they can be released. Each test that the release candidate passes gives us more confidence that this particular combination of binary code, configuration information, environment, and data will work.*

*If the **release candidate** passes all the tests, it can be **released.**”*

# The Deployment Pipeline

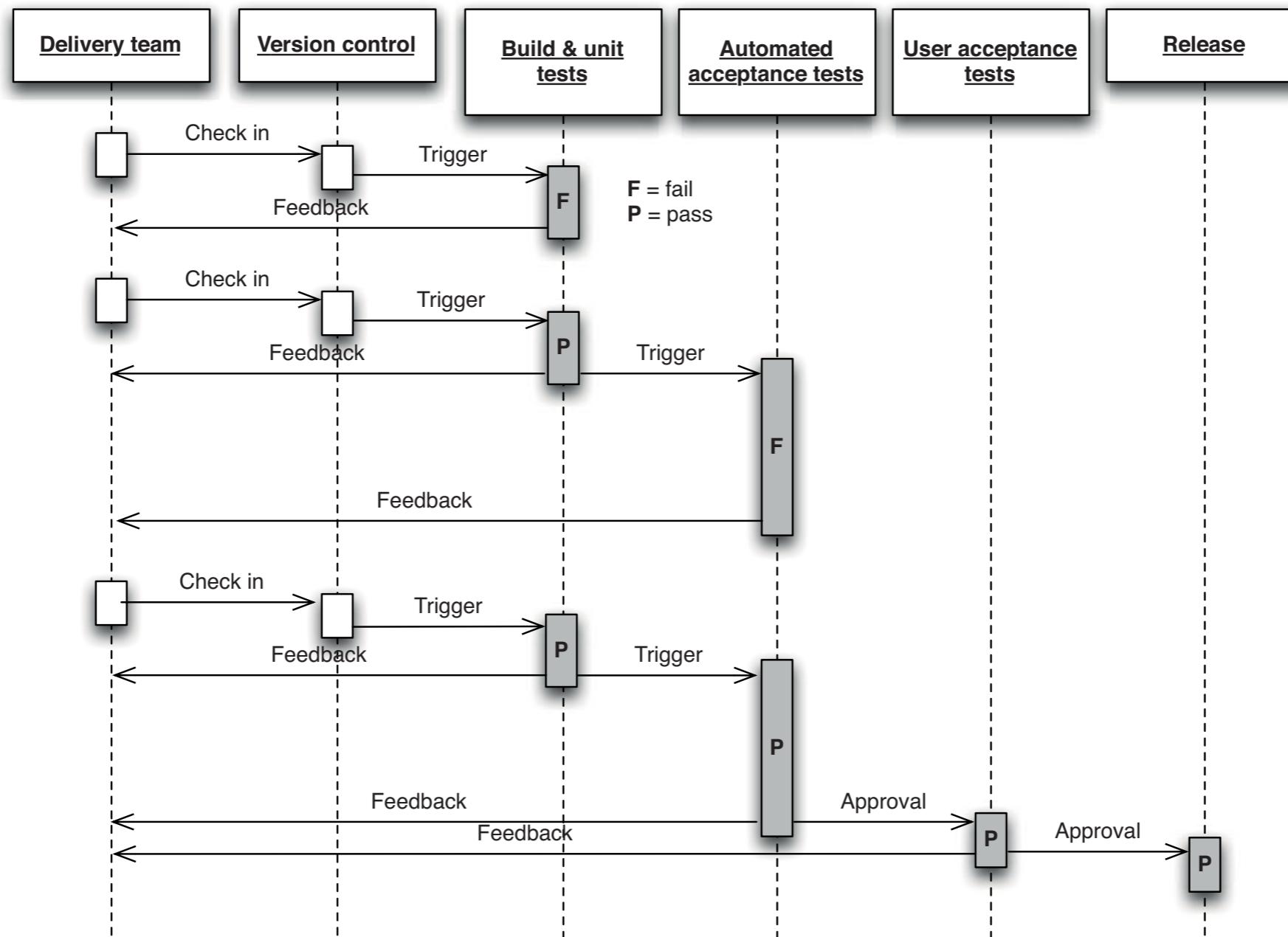


Figure 5.2 *Changes moving through the deployment pipeline*

# The Deployment Pipeline

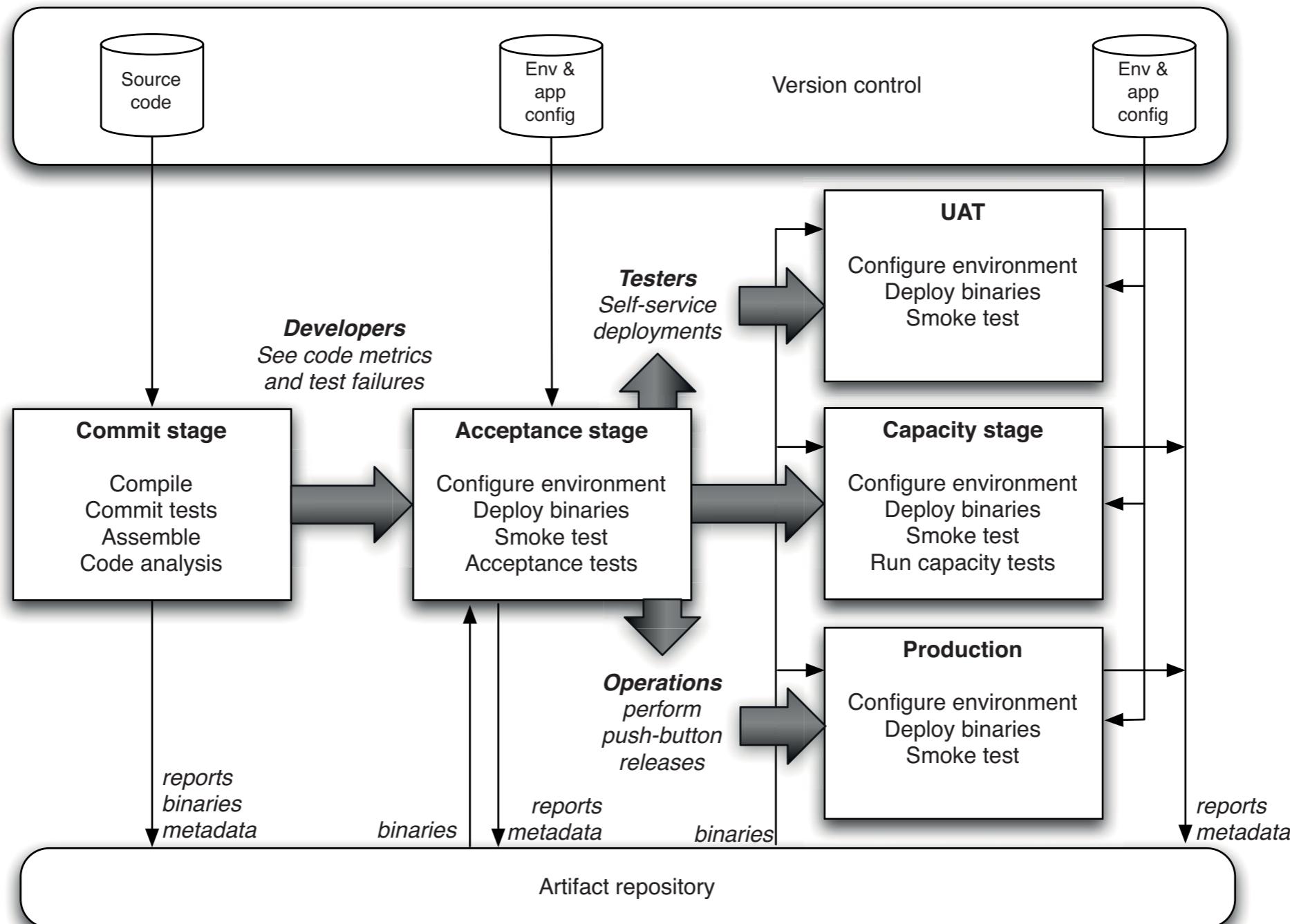


Figure 5.4 Basic deployment pipeline

# The Commit Stage

---

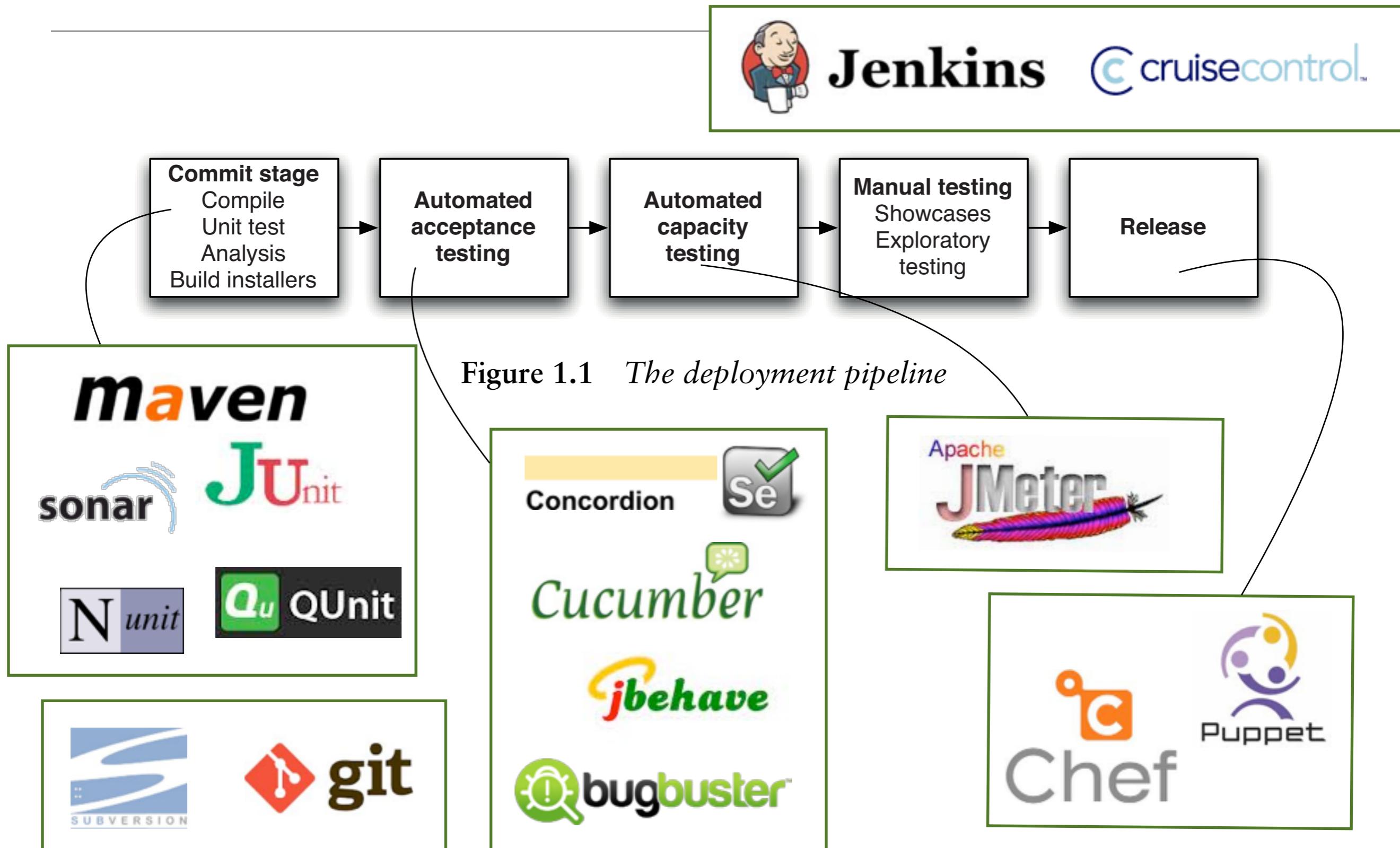
- The commit stage begins with a change to the state of the project—that is, a **commit to the version control system**.
- It **ends with either a report of failure or**, if successful, **a collection of binary artifacts and deployable assemblies** to be used in subsequent test and release stages, as well as reports on the state of the application.
- Ideally, a commit stage should **take less than five minutes to run**, and certainly no more than ten.
- The commit stage represents, in more ways than one, **the entrance into the deployment pipeline**. Not only is it the point at which a new release candidate is created; it is also where many teams start when they begin to implement a deployment pipeline. **When a team implements the practice of continuous integration, it creates a commit stage in the process of doing so.**

# Implementing a Deployment Pipeline

---

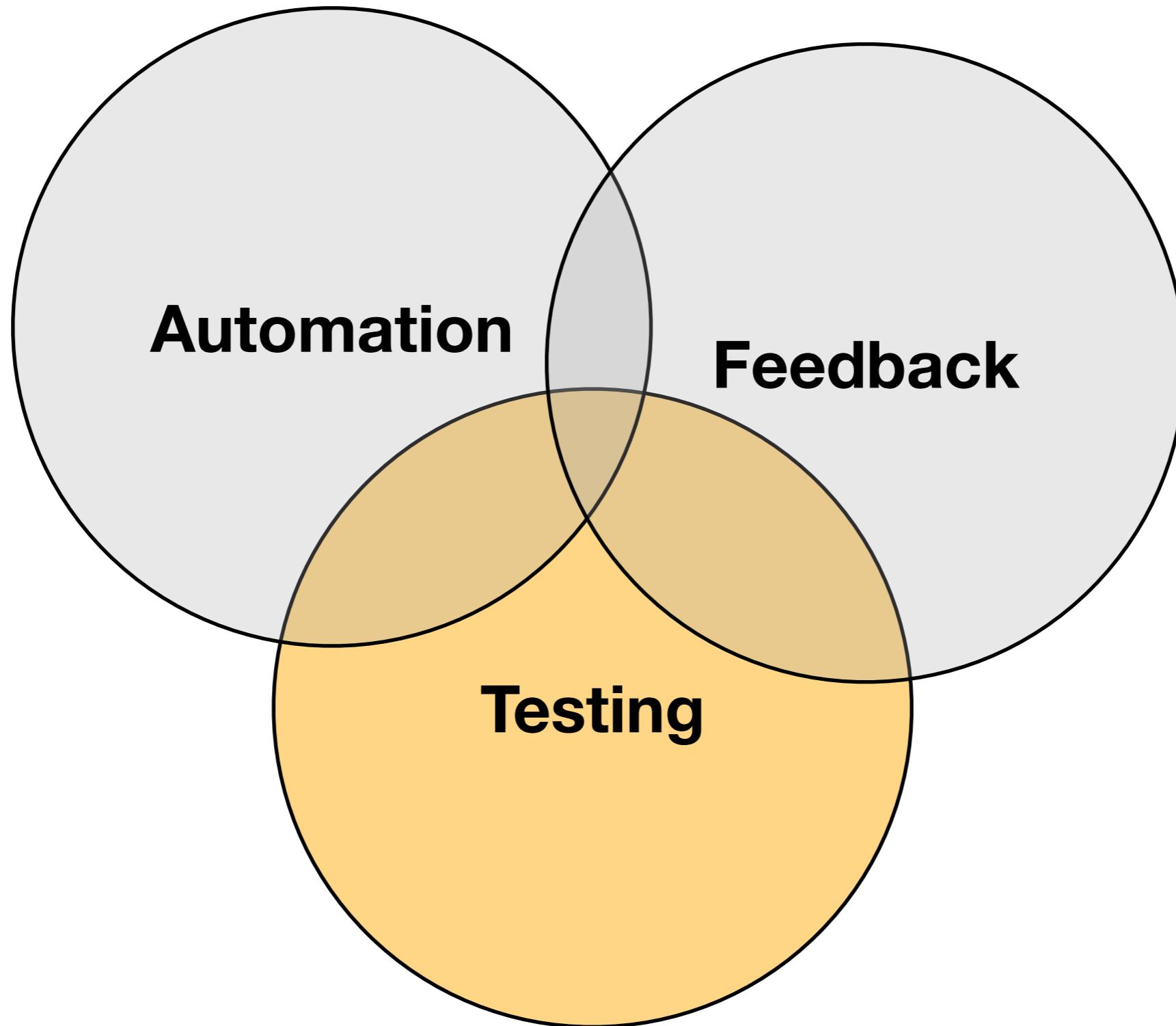
- **Take an agile approach.** Start simple, iterate and improve on a continuous basis.
- Consider the following elements:
  - Automate the **build and deployment** process.
  - Automate **unit tests** and **code analysis**.
  - Automate **acceptance tests**.
  - Automate **releases**.

# Tools



# The Pillars of Continuous Delivery

---



# Testing

---

*“In many projects, testing is treated as a distinct phase carried out by specialists.*

*However, high-quality software is only possible if **testing becomes the responsibility of everybody involved in delivering software** and is practiced right from the beginning of the project and throughout its life.”*

# Testing

---

*“Testing is primarily concerned with **establishing feedback loops** that drive development, design, and release.*

*Any plan that defers testing to the end of the project is broken because it removes the feedback loop that generates higher quality, higher productivity, and, most importantly of all, any **measure of how complete the project is.**”*

# Types of Tests

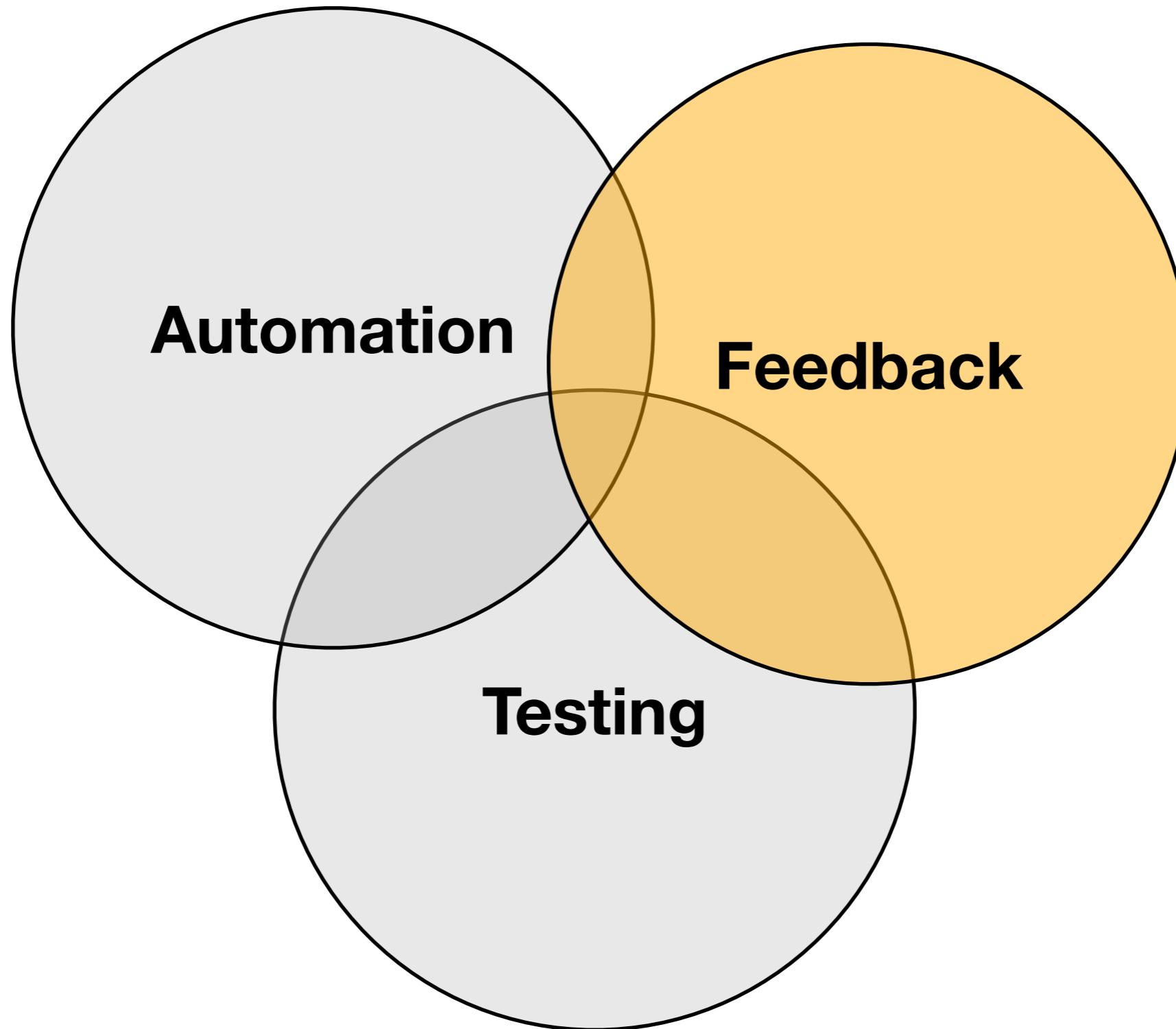
- **Unit Tests**
- **Integration Tests**
- **Automated Acceptance Tests**
- **Systemic Quality Tests**  
(performance, capacity, security)
- **User Acceptance Tests**



Figure 1.1 *The deployment pipeline*

# The Pillars of Continuous Delivery

---



# Feedback

---

- Feedback is essential to frequent, automated releases. There are three criteria for feedback to be useful:
  - Any change, of whatever kind, needs to **trigger** the feedback process.
  - The feedback must be **delivered as soon as possible**.
  - The delivery team must **receive feedback** and then **act on it**.

# Broadcasting Feedback

---

“You should **measure continually and broadcast the results** of the measurements in some hard-to-avoid manner, such as on a very visible poster on the wall, or on a computer display dedicated to showing bold, big results.

Such devices are known as **information radiators**.”



# What to Measure?

---

*“What you choose to measure will have an enormous influence on the behavior of your team. According to the lean philosophy, it is essential to optimize globally, not locally.*

*So it is important to have a global metric that can be used to determine if the delivery process as a whole has a problem.”*



*Is it a good idea to measure lines of code?  
Or the number of bugs fixed?*

# What to Measure?

---

*“For the software delivery process, the most important global metric is **cycle time**. This is the time between deciding that a feature needs to be implemented and having that feature released to users.”*

If the pipeline implementation makes it possible how the cycle time is spent on the successive phases, then it is possible to find bottlenecks and to fix them.

*Happy  
DevOps Family*



# Summary

---

*“There is **no one-size-fits-all solution** to the complex problem of implementing a deployment pipeline.*

*The crucial point is to create a system of record that **manages each change from check-in to release**, providing the **information** you need to discover problems as early as possible in the process.”*

# Summary

---

*“Having an implementation of the deployment pipeline can then be used to **drive out inefficiencies** in your process so you can make your feedback cycle faster and more powerful, perhaps by adding more automated acceptance tests and parallelizing them more aggressively, or by making your testing environments more production-like, or by implementing better configuration management processes.”*

# Summary

---

*“A deployment pipeline, in turn, depends on having some **foundations** in place: **good configuration management, automated scripts for building and deploying your application, and automated tests** to prove that your application will deliver value to its users.*

*It also requires **discipline**, such as ensuring that only changes that have passed through the automated build, test, and deployment system get released.”*

# Resources

---

- <http://continuousdelivery.com>



**CONTINUOUS DELIVERY**

BLOG PUBLICATIONS TALKS ABOUT

**GET THE BOOK**

 Amazon (hardback, kindle)  
InformIT (pdf, epub, mobi)  
中文 (in Chinese)  
日本語 (in Japanese)

You can also see a list of all my publications and talks, including slides, on the Publications page.

**GET THE SOFTWARE**



**Publications**

Here is a list of all my publications and conference talks, by category, with newest first.

**Book**

Jez Humble and David Farley, *Continuous Delivery*, Addison-Wesley, 2010

**Papers**

"Why Enterprises Must Adopt Devops to Enable Continuous Delivery", Cutter IT Journal Vol. 24 No. 8 (August 2011, registration required)  
"Agile Release Management" white paper, ThoughtWorks, 2010 (registration required)  
Jez Humble, Chris Read, Dan North, "The Deployment Production Line", Proceedings of Agile 2006, IEEE Computer Society ([Download](#))

**Articles and Interviews**

[Is the Enterprise Ready for DevOps?](#), InfoQ, 1 August 2012  
[Interview with Chris Little for BMC DevOps Leadership Series](#), 28 March 2012  
[Four Principles of Low-Risk Software Releases](#), InformIT, 16 February 2012  
[Analysis for Continuous Delivery: Five Core Practices](#), InformIT, 25 January 2012  
[Simple Talk's Geek of the Week](#), <http://simple-talk.com/>, 28 September 2011  
[Interviewed by Forrester's Jeffrey Hammond on Devops and Continuous Delivery](#), <http://forrester.com/>, 9 September 2011  
[Interview with me and Dave Farley on Continuous Delivery](#), <http://infoq.com/>, 6 September 2011  
"Tired of Playing Ping-Pong with Dev, QA and Ops?", <http://ciupdate.com/>, 10 May 2011  
"Five predictions for 2011", <http://cmcrossroads.com/>, 4 February 2011  
"What DevOps Means for Enterprises", <http://agileweboperations.com/>, 18 January 2011  
"Continuous Delivery: The Value Proposition", <http://informit.com/>, 26 October 2010  
"Continuous Delivery: Anatomy of the Deployment Pipeline", <http://informit.com/>, 7 September 2010 (Chapter 5 of the book)

**Recordings**

[Hanselminutes: Me and Martin Fowler interviewed by Scott Hanselman](#), 2 October 2012  
[Devops Cafe: Interviewed by Damon Edwards and John Willis on continuous delivery](#), 26 September 2012  
[Devnology interview Dave Farley and me about continuous delivery](#), 27 November 2011  
30m interview on Continuous Delivery, the Lean Startup, and Devops during Agile 2011  
[Interviewed with Martin Fowler during QCon San Francisco](#), 2010

**FOLLOW ME ON [twitter](#)**

**SIGN UP FOR THE NEWSLETTER**

email address

**SIGN UP >>**

**RECENT POSTS**

There's No Such Thing as a "Devops Team"  
Elisabeth Hendrickson Discusses Agile Testing  
Continuous Delivery: The Case of Apple  
John Allspaw Discusses Devops and Continuous Delivery  
Why Software Development Methodologies Suck

**ARCHIVES**

October 2012 (3)  
September 2012 (1)  
August 2012 (1)  
July 2012 (2)  
February 2012 (1)  
January 2012 (1)  
December 2011 (1)  
July 2011 (2)  
May 2011 (1)  
January 2011 (1)  
November 2010 (1)  
October 2010 (1)  
September 2010 (1)  
August 2010 (2)  
July 2010 (1)  
June 2010 (1)  
February 2010 (1)