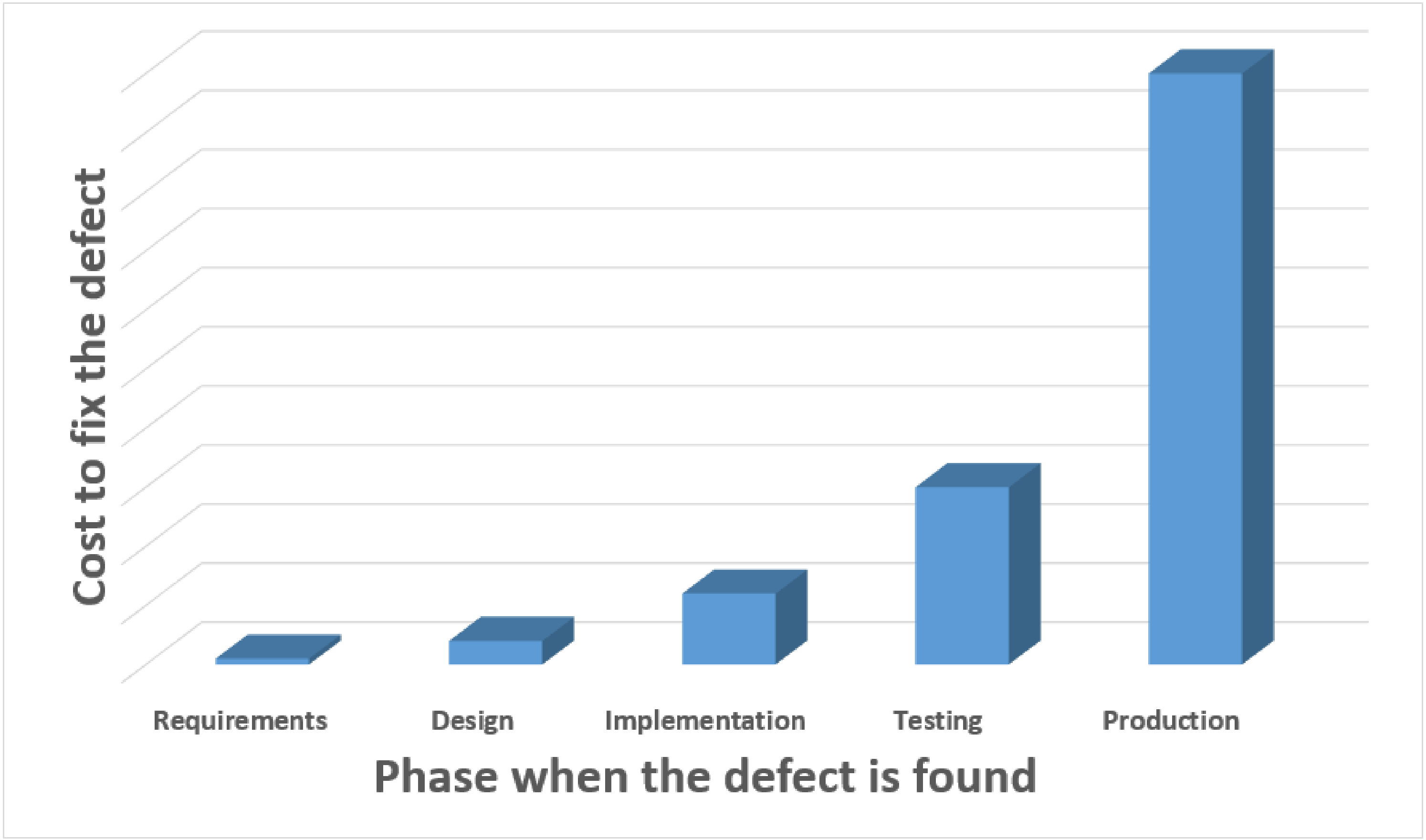
Continuous Integration and Continuous Delivery

Kamil Król

Software challenges

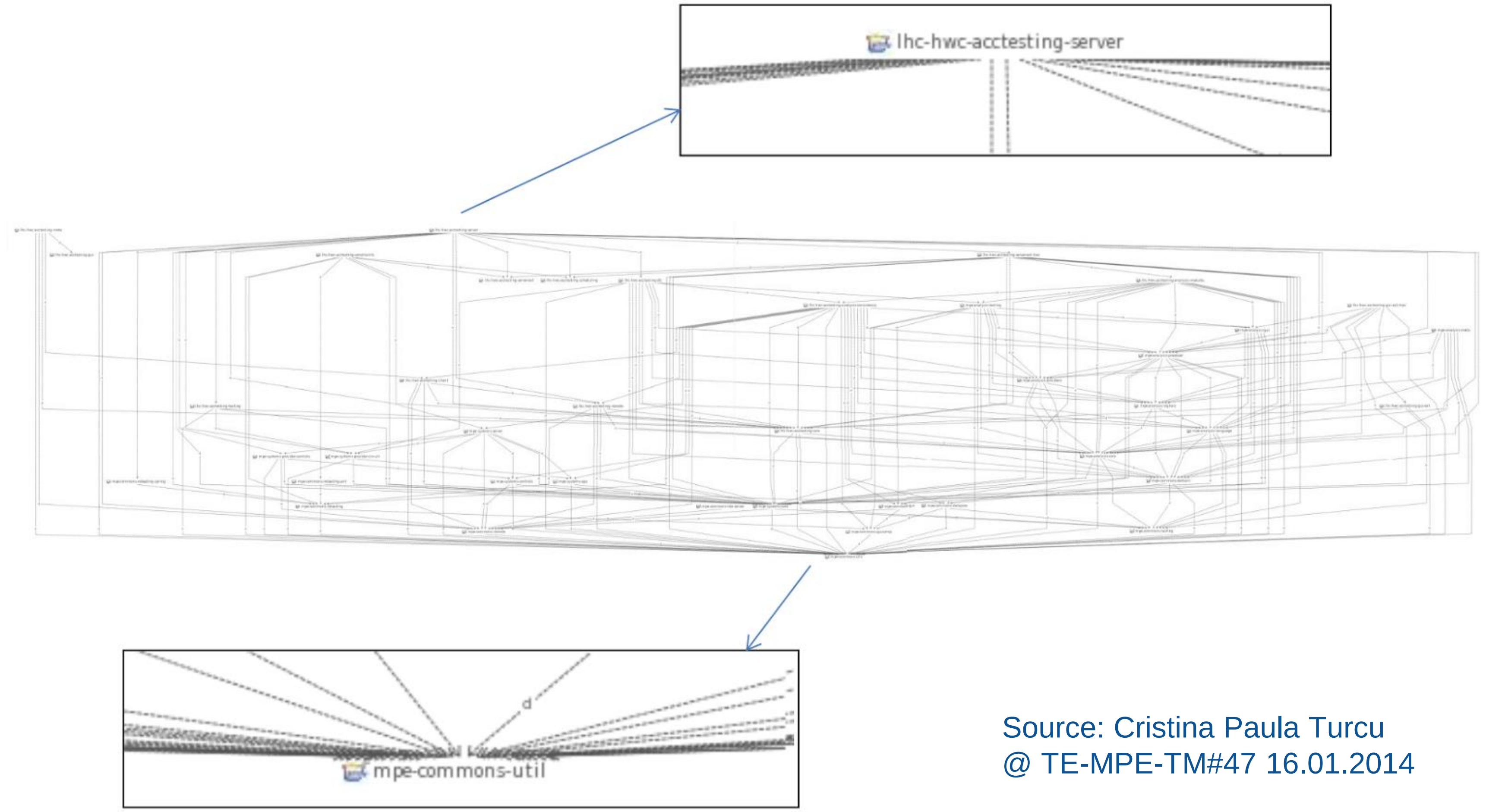
- Team work
  - Multiple people
  - Different backgrounds
  - High turnover
- Growing complexity of project
- Quality degradations

Software challenges



The cost of fixing the defect grows exponentially in time!

Software challenges



Source: Cristina Paula Turcu
@ TE-MPE-TM#47 16.01.2014

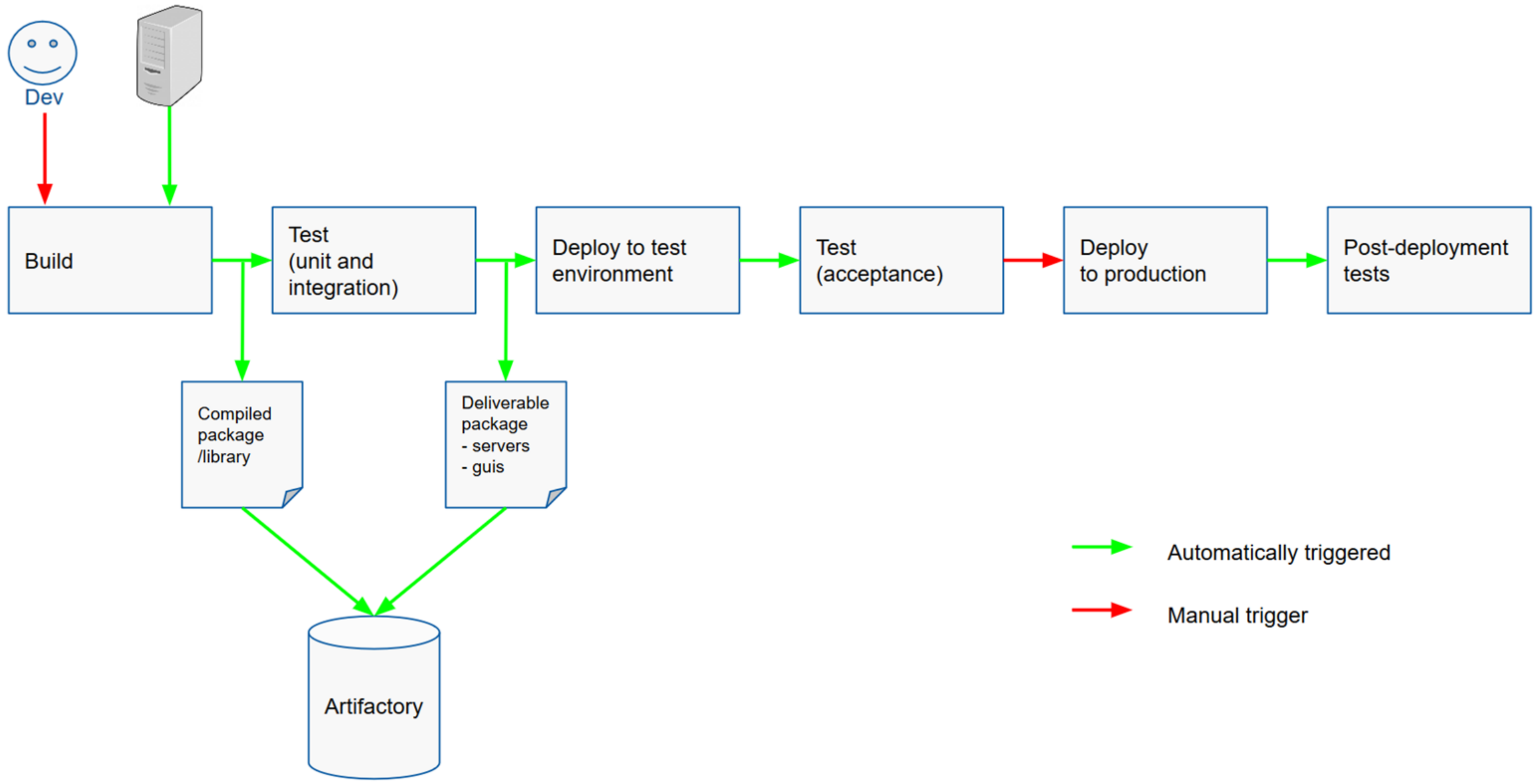
Software challenges

Refactoring
Improving the Design of Existing Code
Martin Fowler
Clean Code
A Handbook of Agile Software Craftsmanship
Robert C. Martin
Patterns of Enterprise Application Architecture
Refactoring to Patterns
Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

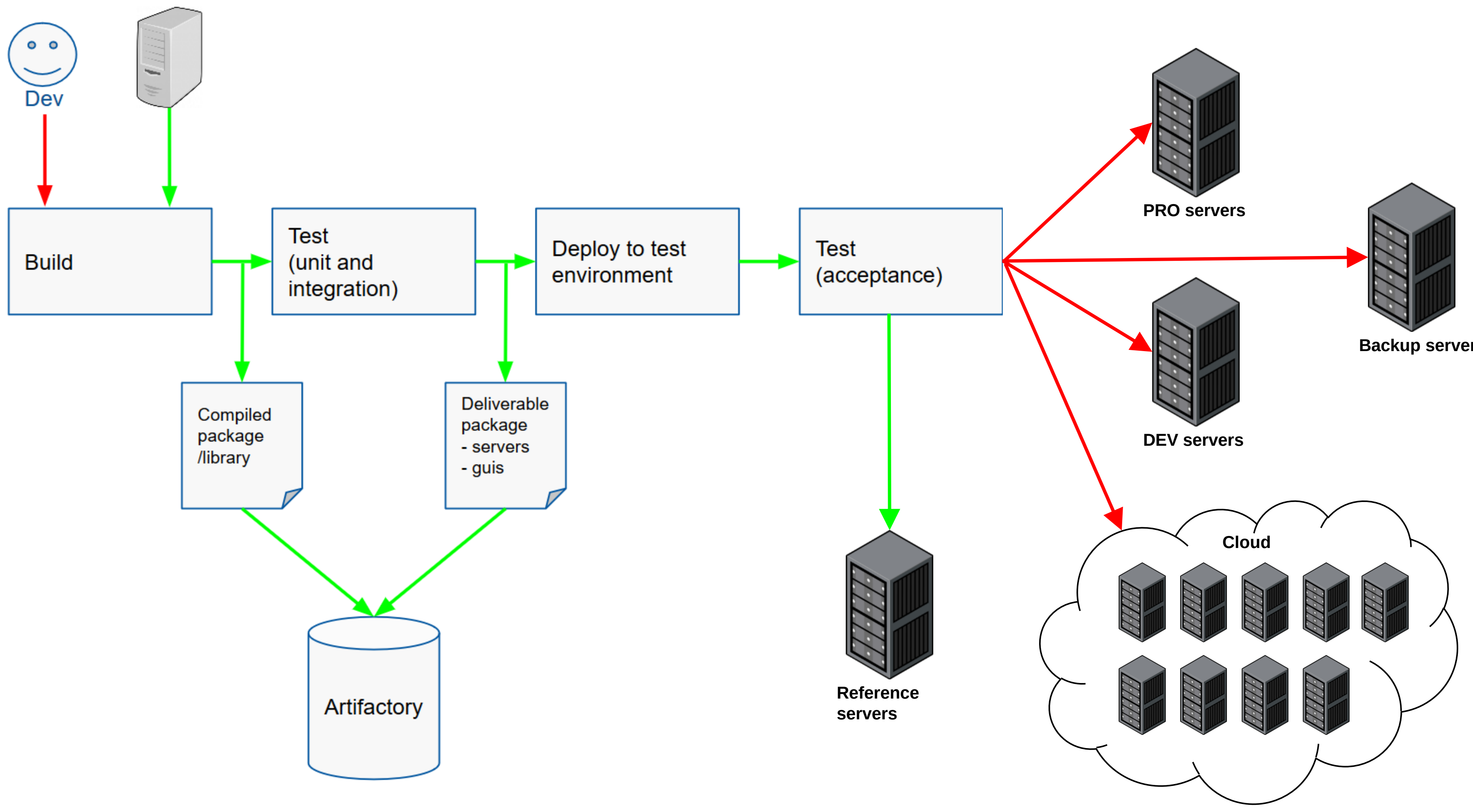
Brian Foote
Joseph Yoder
“Big Ball of Mud”

Can Continuous Integration/Delivery help us?

Continuous Integration



The challenge



Release

- Release == Risk
- Release == Fear
- Release == Working over time - frequent calls
- Release == Working during weekends
- Release == Long preparations
- Release == Boring repeatable tasks

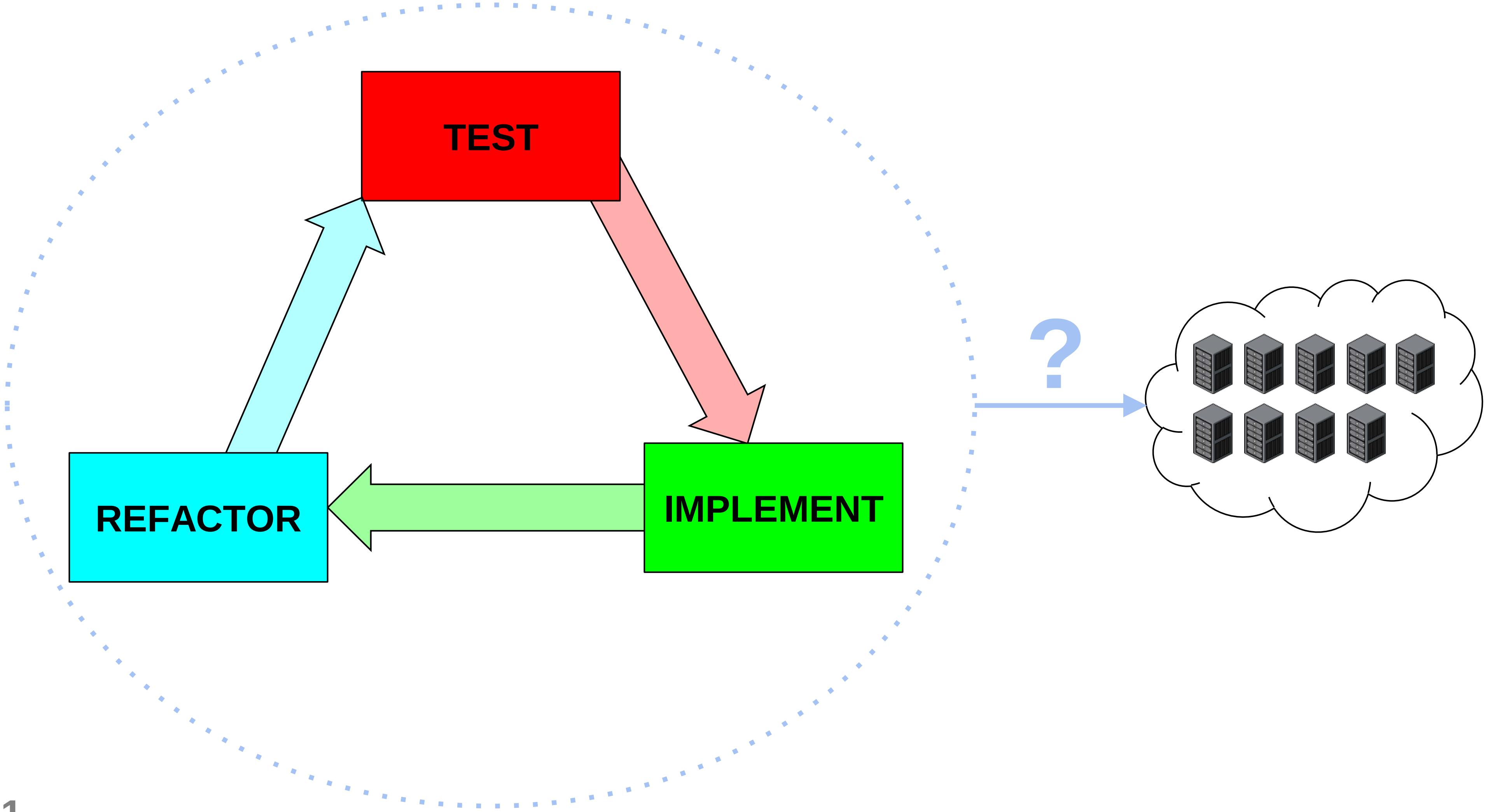
Can we do something with it?

The Agile Manifesto

....
We follow these principles:
Our highest priority is to satisfy the customer through early and **CONTINUOUS DELIVERY** of valuable software.
....

@see: Agile Manifesto

Incremental development



The definition

“Continuous Delivery is a software development discipline where you build software in such a way that the software <u>can be released</u> to production <u>at any time</u>.“

@Martin Fowler

Principles

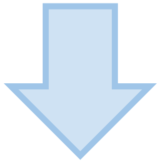
- **Continuous Delivery needs to be built on the solid Continuous Integration**
  - Frequent integration with others - (ideally) one main trunk
  - Confident tests protecting our development
  - Easy rollback should be possible

Principles

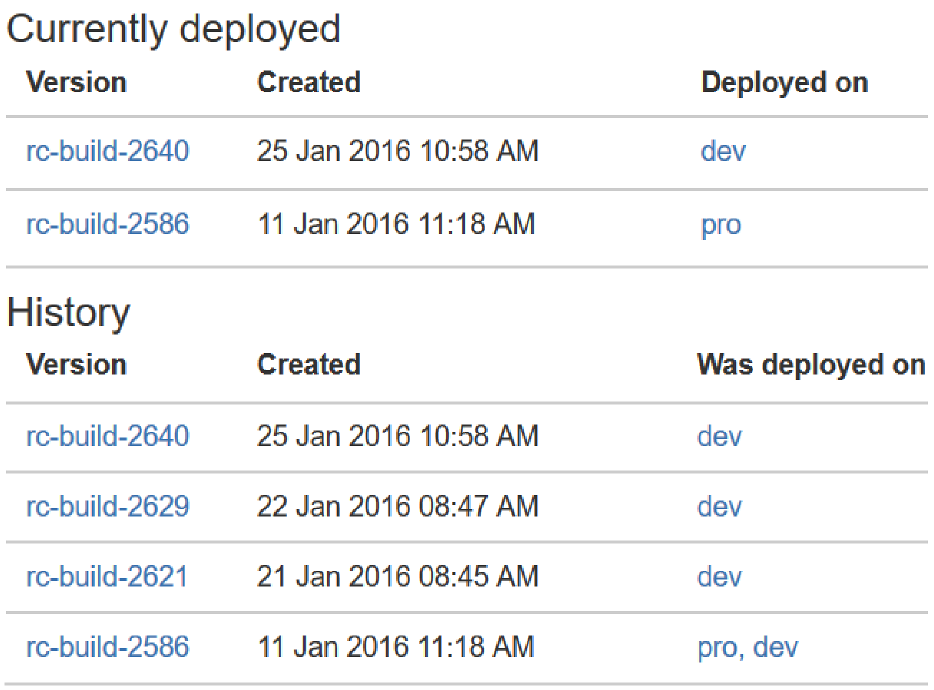
- Introduce as much **automation** as possible
- Test close to the production
  - Test environments and testing approaches should be similar to real life use cases

Principles

- **Avoid:**
  - unofficial releases
  - hot fixes without releasing
  - patches sent via e-mail



Full history of releases



Currently deployed

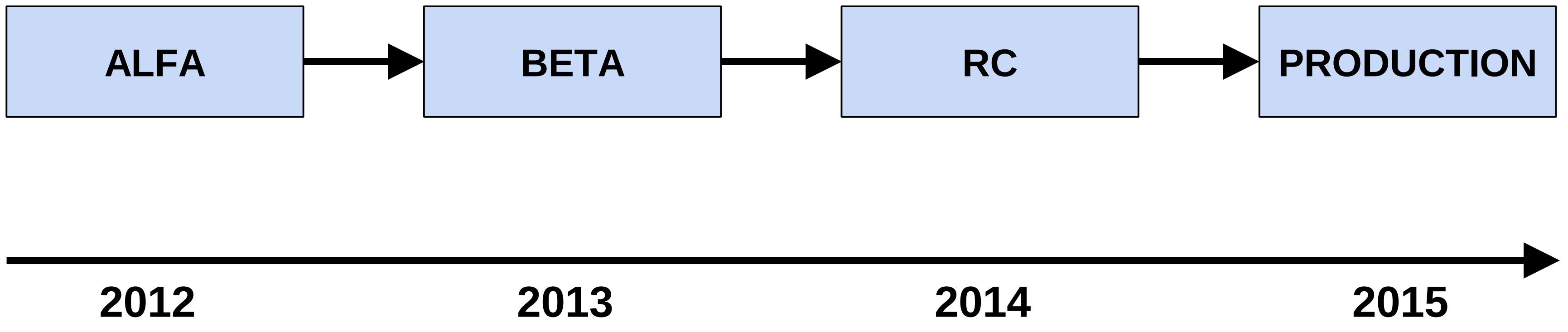
Version	Created	Deployed on
rc-build-2640	25 Jan 2016 10:58 AM	dev
rc-build-2586	11 Jan 2016 11:18 AM	pro

History

Version	Created	Was deployed on
rc-build-2640	25 Jan 2016 10:58 AM	dev
rc-build-2629	22 Jan 2016 08:47 AM	dev
rc-build-2621	21 Jan 2016 08:45 AM	dev
rc-build-2586	11 Jan 2016 11:18 AM	pro, dev

Principles

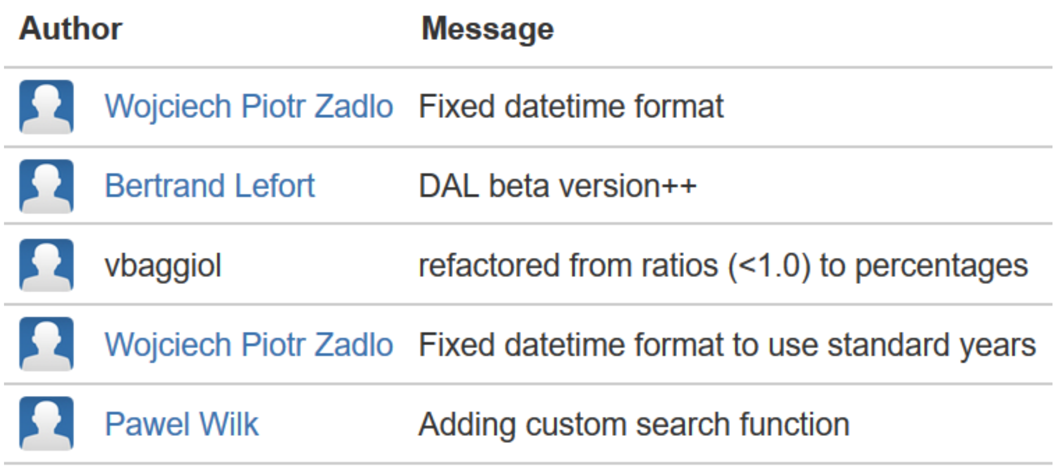
- **More releases => smaller releases**
- **Avoid release chains**



Why?

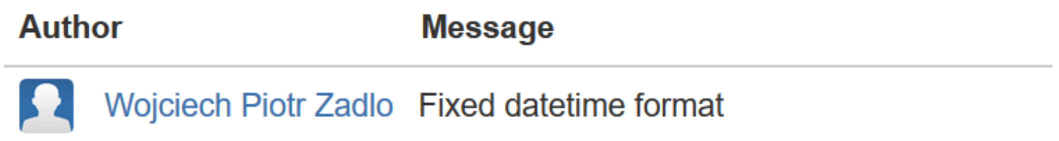
- Small release v. big release?
  - **Lower the deployment risk**
- As a developer, when do you consider the functionality done (completed)?
  - **Sense of progress**
- How do you validate/test application functionalities?
- Do we work on the right thing?
  - **Quick user feedback**

Why?



Author	Message
Wojciech Piotr Zadlo	Fixed datetime format
Bertrand Lefort	DAL beta version++
vbaggiol	refactored from ratios (<1.0) to percentages
Wojciech Piotr Zadlo	Fixed datetime format to use standard years
Pawel Wilk	Adding custom search function

v.



Author	Message
Wojciech Piotr Zadlo	Fixed datetime format

- **RELEASE A - 5 changes**
- **RELEASE B - 1 change**

Something goes wrong during the release...

In case of which release is easier to locate the problem?

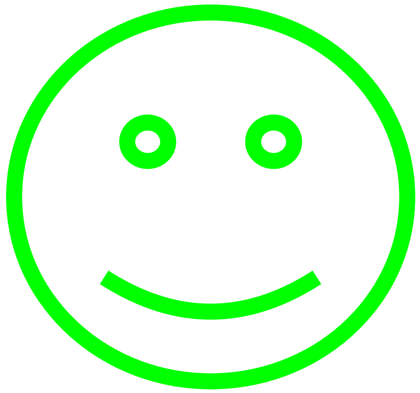
Why?

“***50% or more of functionality delivered is rarely or never used.***“
@Jim Highsmith, Adaptive Leadership

Why?

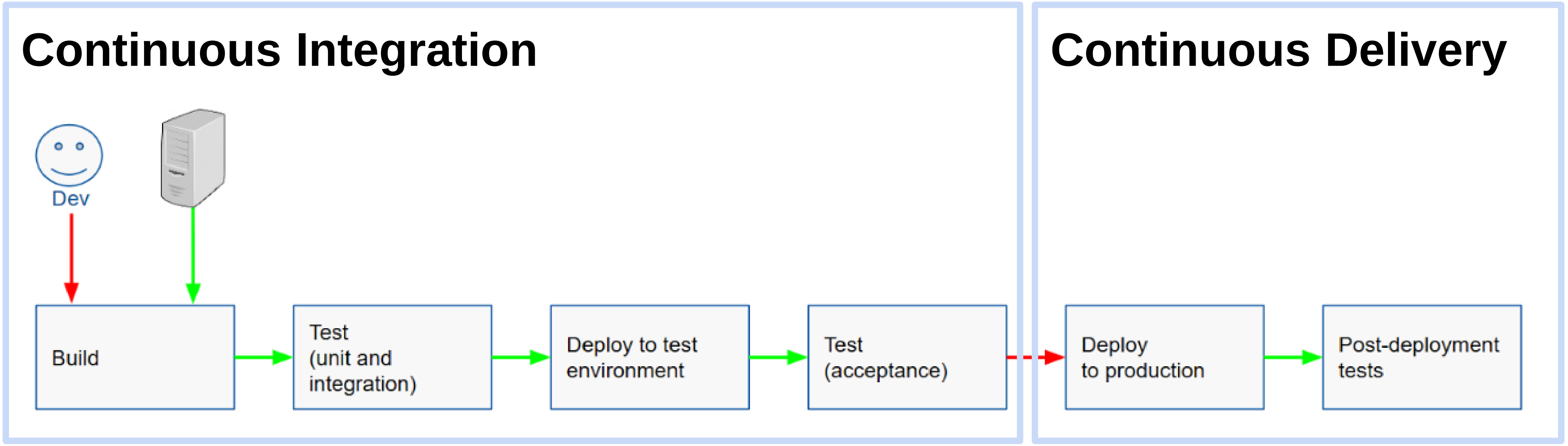
- **We can do the same work, but:**
  - Working less
  - In more relaxed environment
  - Avoiding risk
  - Having more constant workload
  - With high level of confidence

- **“*I wish we did it the old way*”**
  - @noone



How?

- Continuous Delivery is the natural continuation of the Continuous Integration processes



How?

- Continuous Delivery can be implemented using most build servers (both open source and commercial tools available)

TeamCity

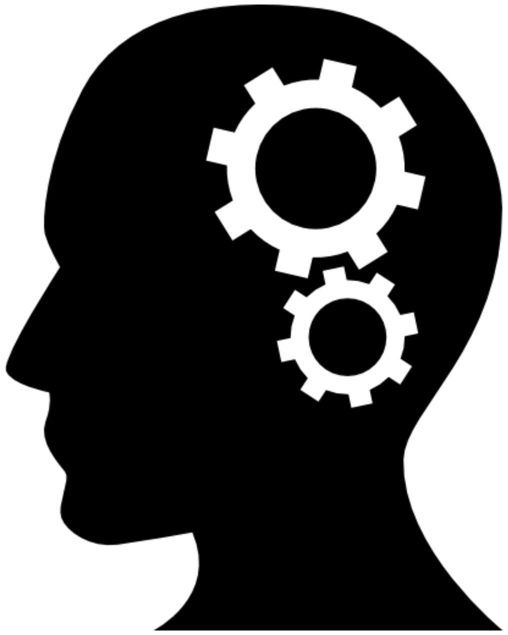
Jenkins

Bamboo

Travis CI

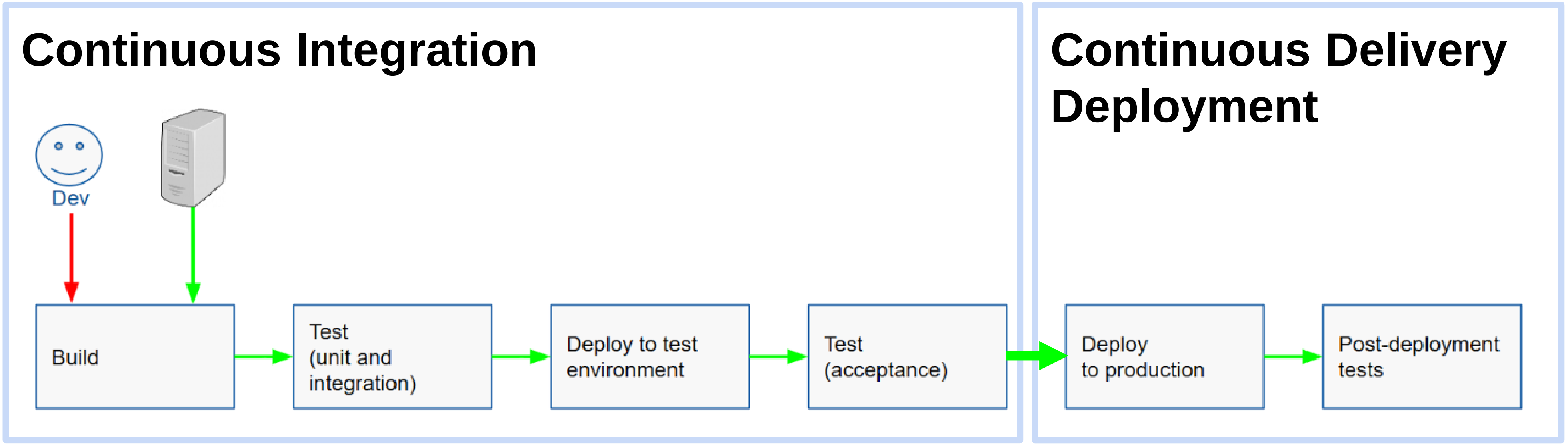
How?

- Continuous Delivery is not a complete set of tools which needs to be used
- It more a way of thinking about the development!



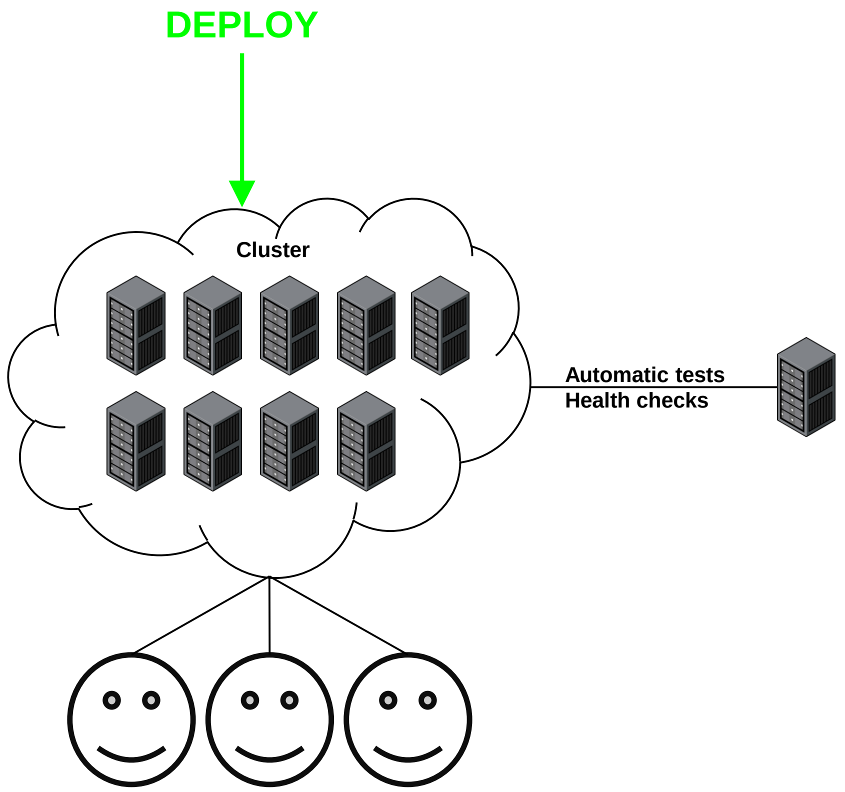
Continuous Deployment

(another level of hardcore ...)



Continuous Deployment

(another level of hardcore ...)

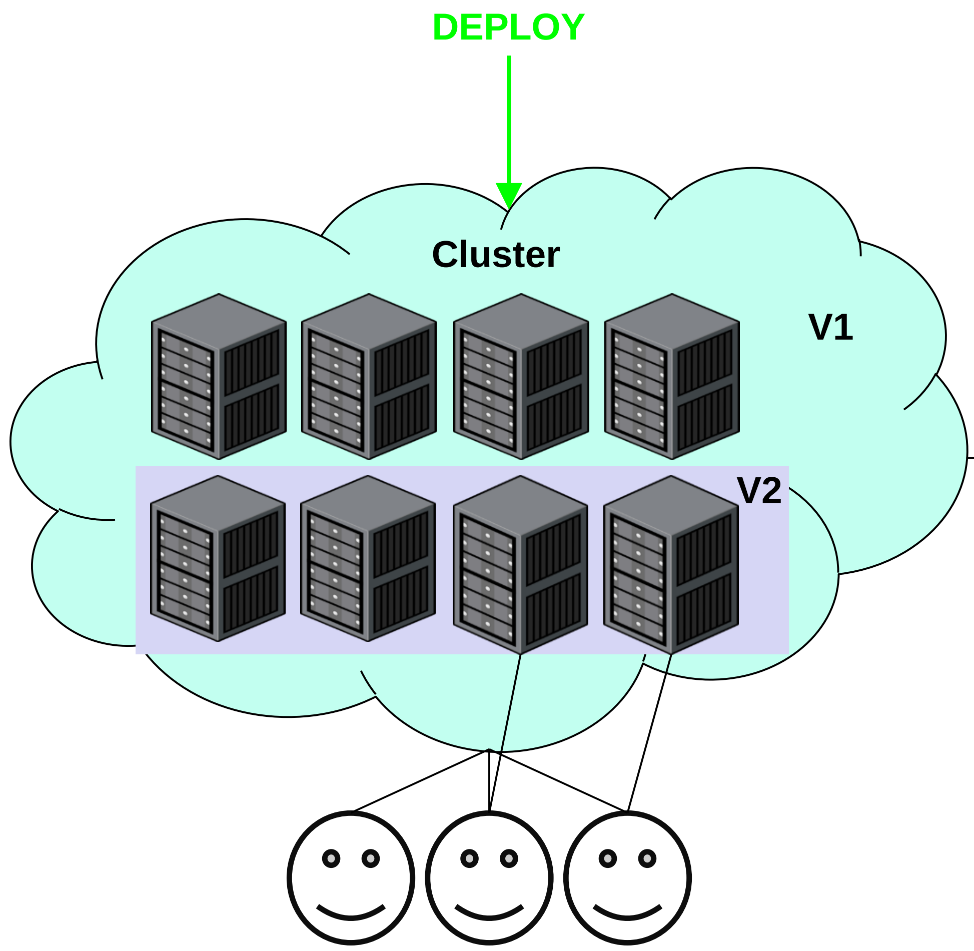


Required:

- High level of confidence in our software
- Application health checks (app condition monitoring)

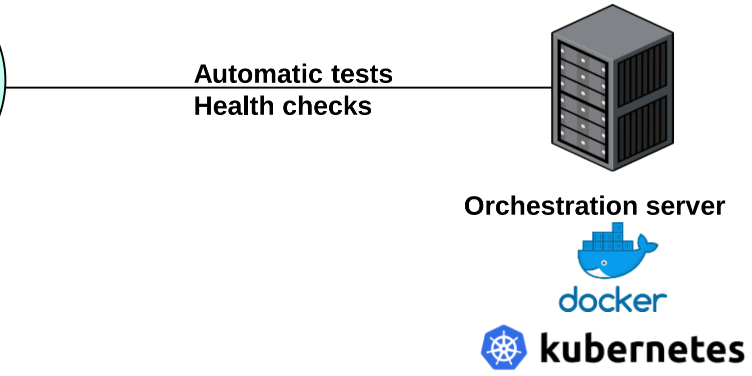
Continuous Deployment - patterns

(another level of hardcore ...)



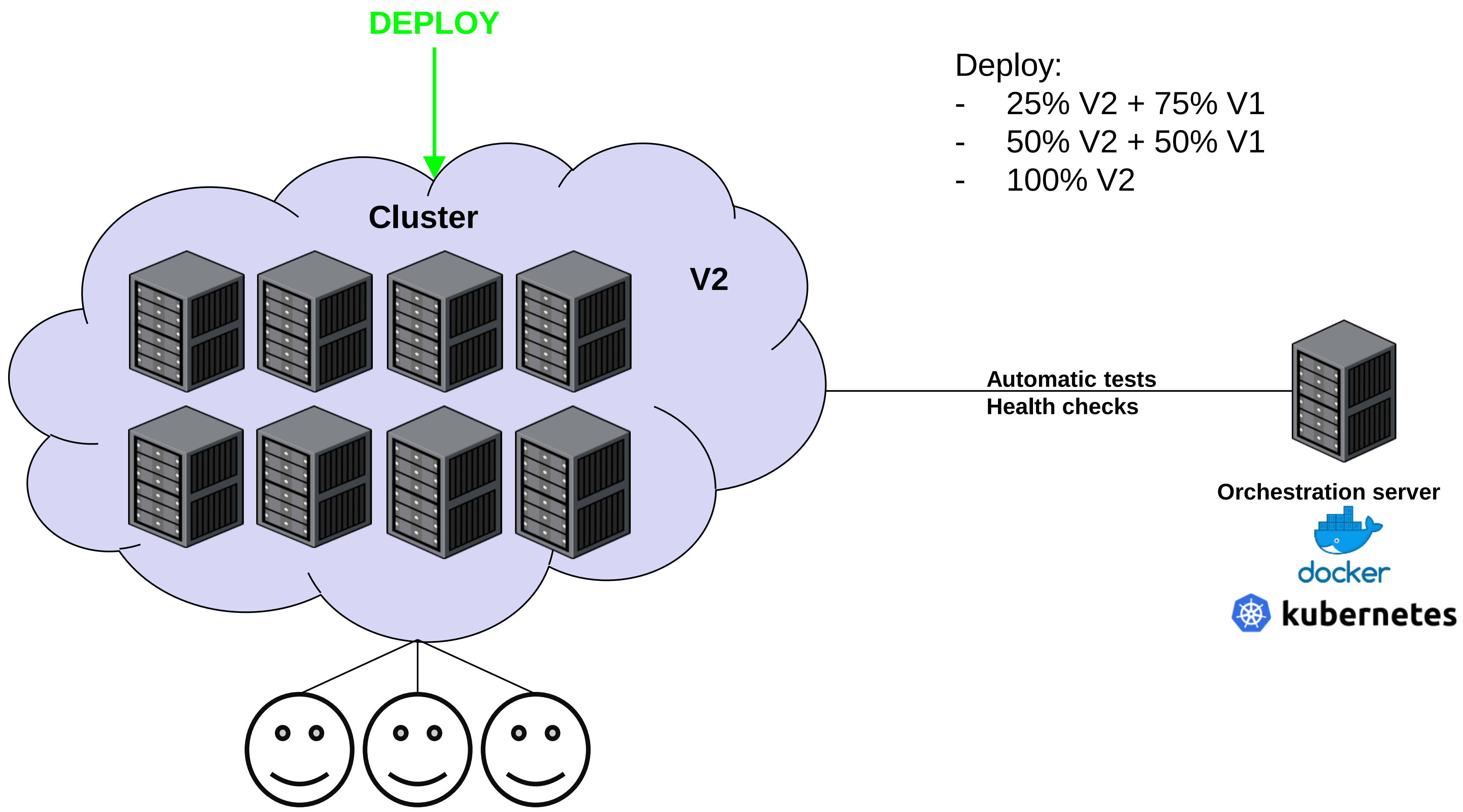
Deploy:

- 25% V2 + 75% V1
- 50% V2 + 50% V1



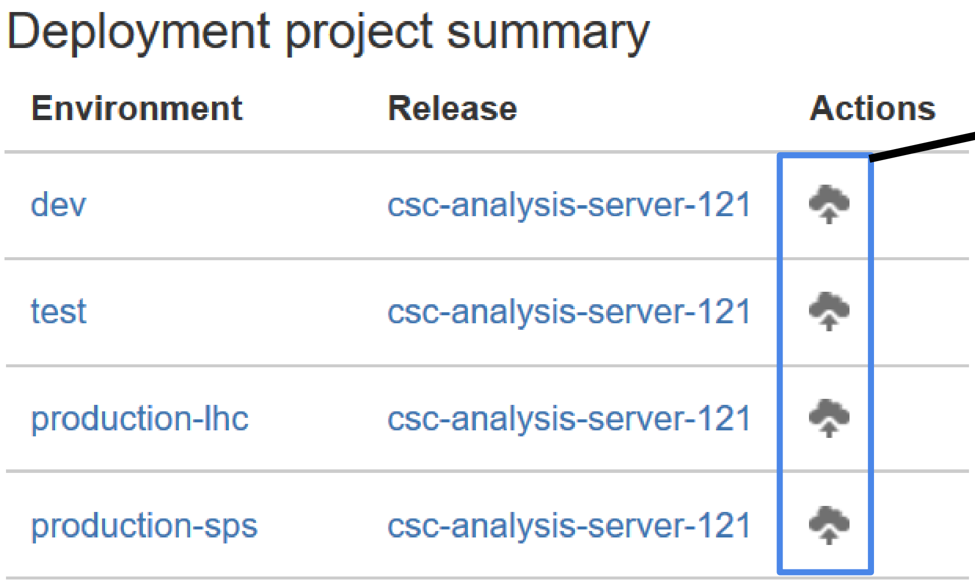
Continuous Deployment - patterns

(another level of hardcore ...)



Deploy:
- 25% V2 + 75% V1
- 50% V2 + 50% V1
- 100% V2

Do we do it right?



Deployment project summary

Environment	Release	Actions
dev	csc-analysis-server-121	☁
test	csc-analysis-server-121	☁
production-lhc	csc-analysis-server-121	☁
production-sps	csc-analysis-server-121	☁

The 'deploy' button
How confident are you?

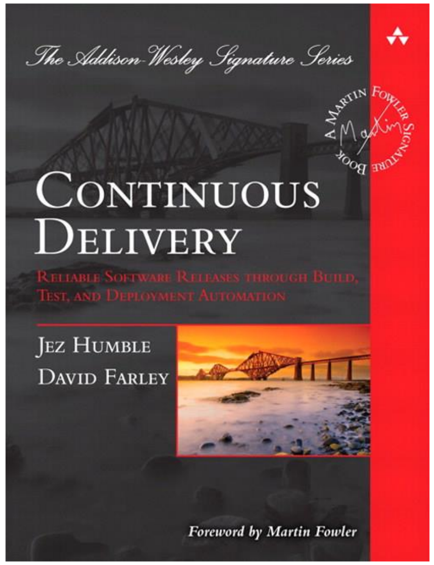
Answer these 3 questions

- Is it easy to deploy?
- Is it easy to rollback?
- Clients not affected?

If you answer yes to all questions, you are doing the Continuous Delivery right!

Summary

- Continuous Integration with Continuous Delivery can give us high level of confidence
- It imposes the incremental development
  - We build the right thing
  - It's easy to locate problems
  - It helps developers see the progress
  - **Clients receive functionalities quickly**
- It's fashionable
- It's a standard


The Addison-Wesley Signature Series
A Martin Fowler Signature Book
Continuous Delivery
Reliable Software Releases through Build, Test, and Deployment Automation
Jez Humble
David Farley
Foreword by Martin Fowler

Thanks a lot!

Any questions?