LEAN SOFTWARE DEVELOPMENT

Part 1: from continuous integration to continuous delivery
Part 2: agile testing
Part 3: introduction to DevOps

Master Hes·so
!nnokick

Olivier Liechti, Spring 2016

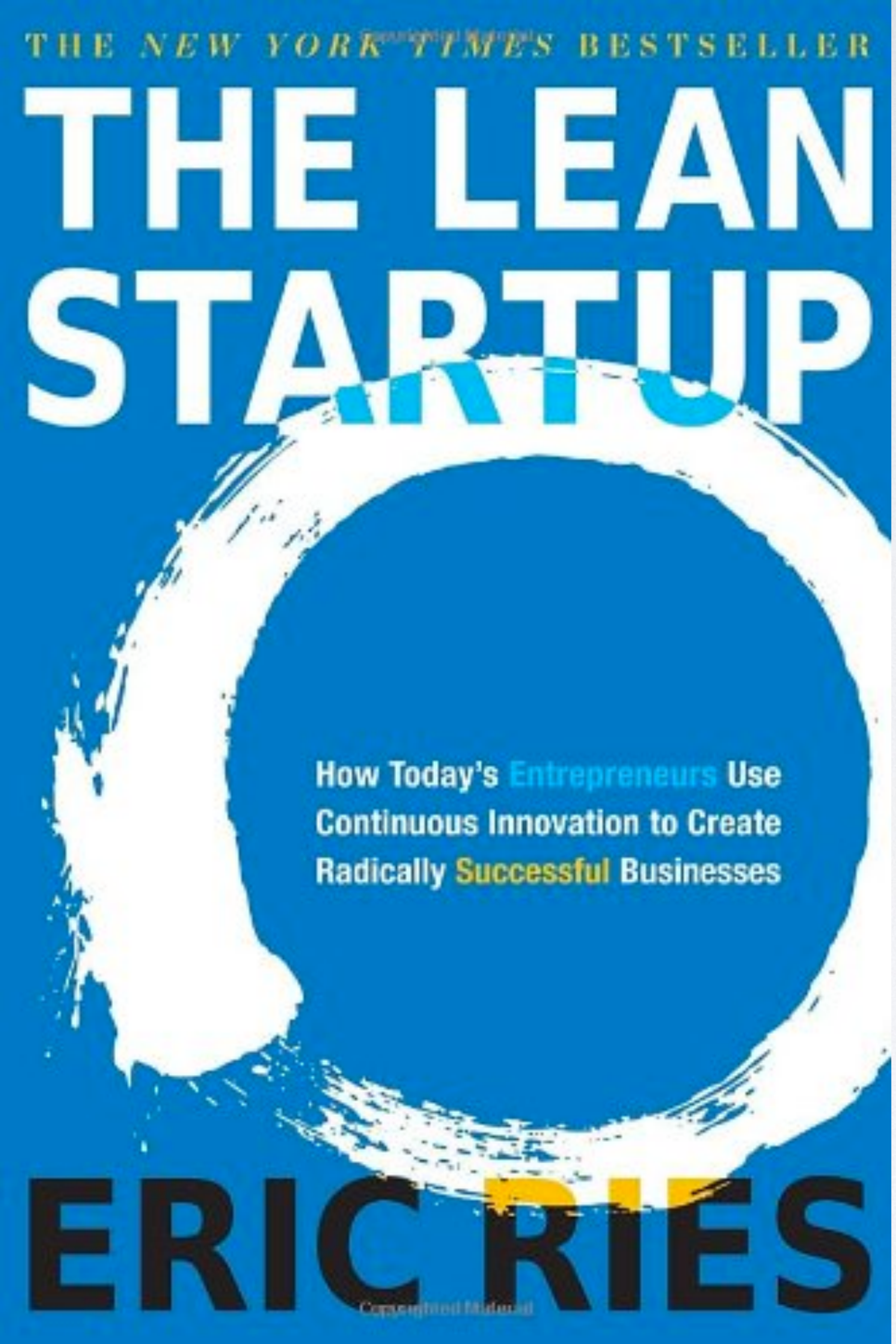
TODAY'S AGENDA

- **Continuous integration & delivery (20')**
- **Team work (20'), presentations & discussion (4 x 10')**
- **Wrap-up: putting it in practice (10')**

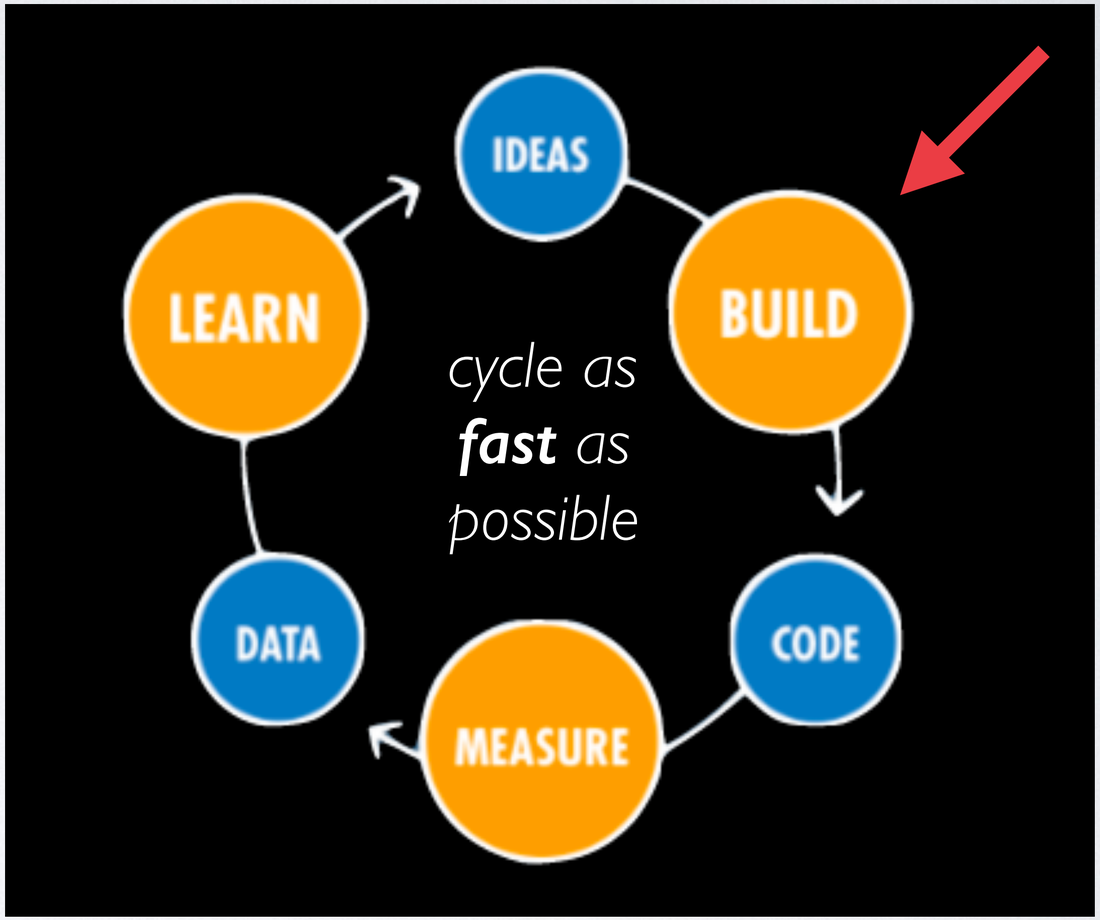
QUICK POLL

TOPICS

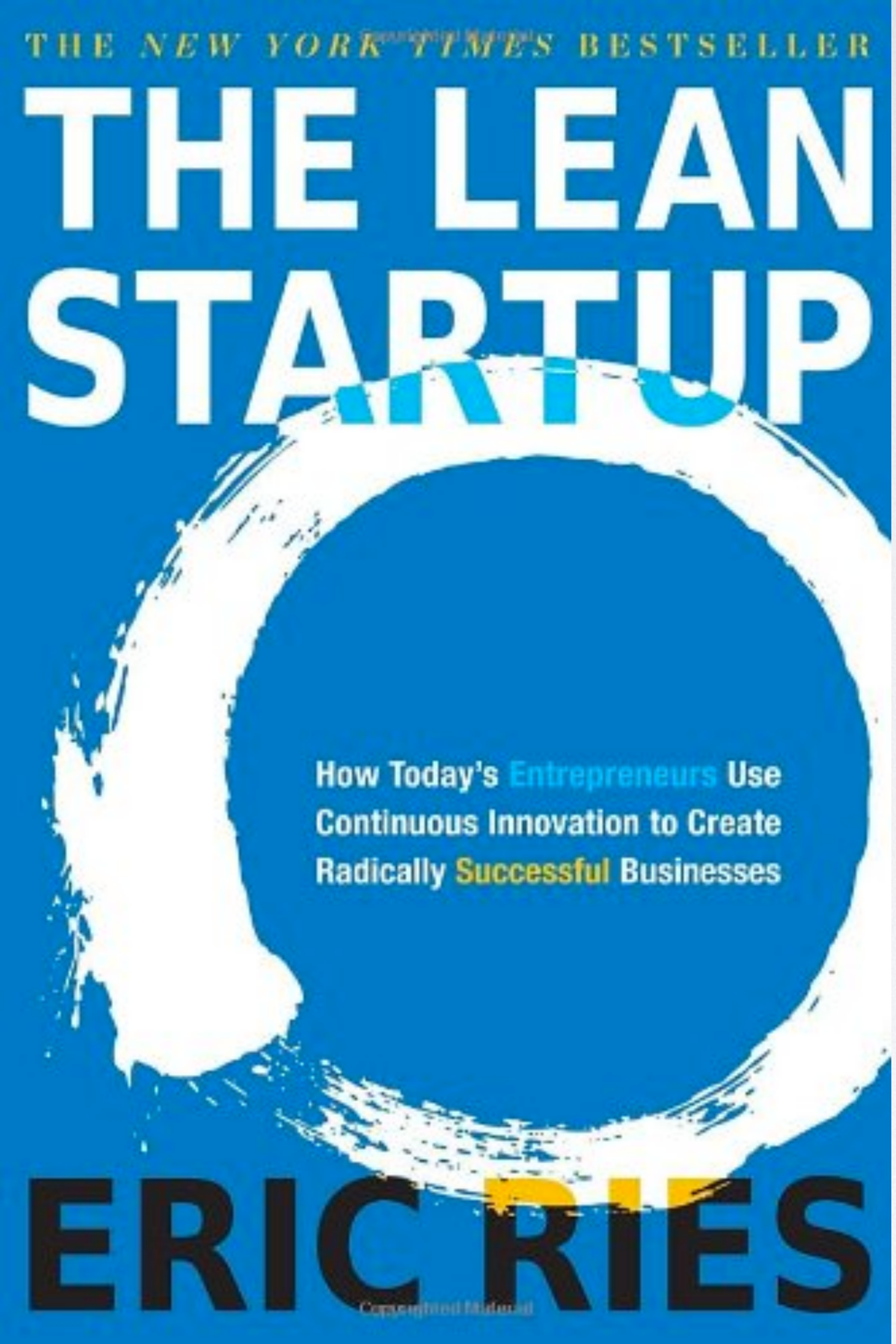
- **Motivations**
- **Continuous integration as a core agile practice**
- **Towards continuous delivery & continuous deployment**
- **CI/CD Pipelines**



MOTIVATION



http://www.startuplessonslearned.com/2009/06/why-continuous-deployment.html



MOTIVATION

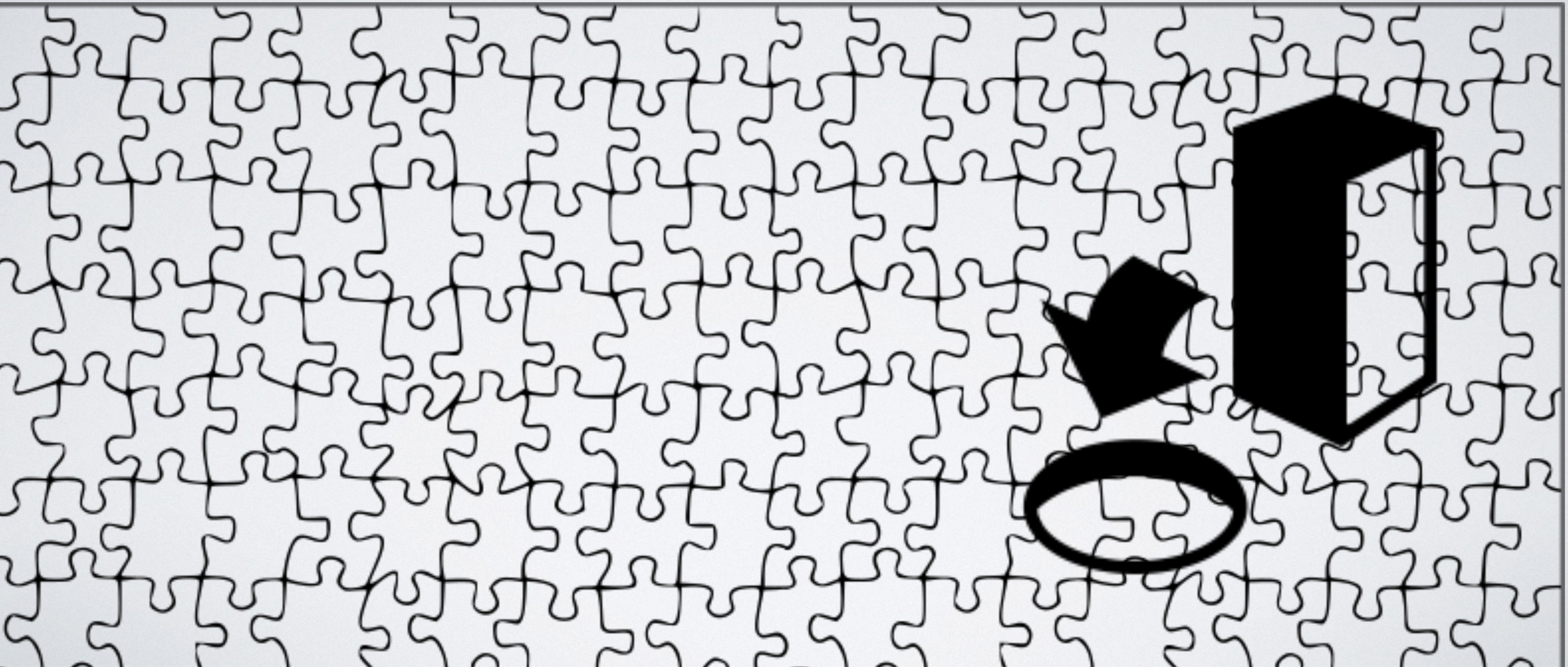
*How do we **keep a fast validated learning cycle**, even when building a complex software product, used by of people on a 24/7 basis?*

*How can **software engineering practices** and **tools** help us?*

CORE AGILE PRACTICES

- **Small releases**
- **Continuous integration**
- **Test-driven development**

CONTINUOUS INTEGRATION



INTEGRATION IS A PITA

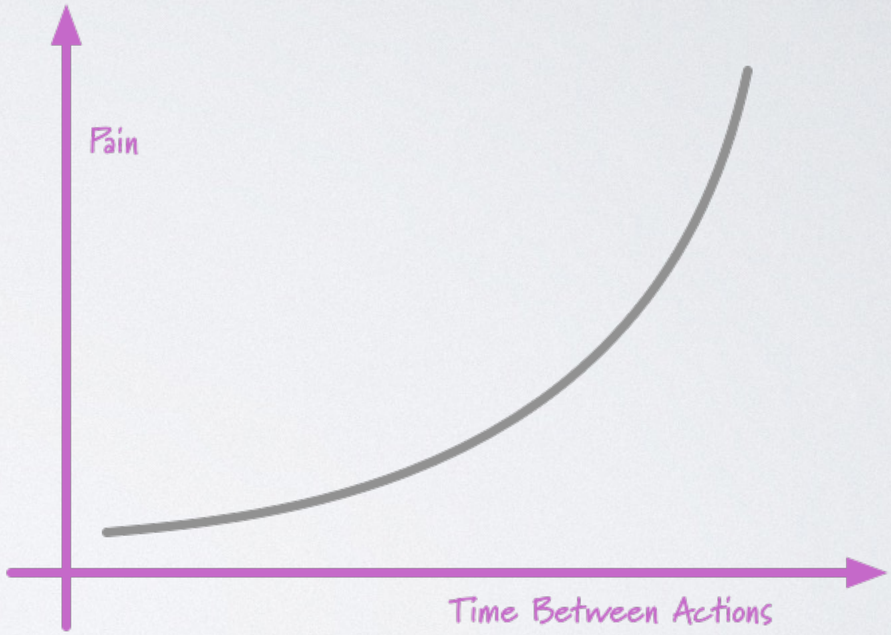
- **Team members work on several components in parallel**
- **The components need to be integrated to form a product**
- **Doing this integration is always painful**
- **But there are ways to reduce the pain**

IF IT HURTS...

... DO IT MORE OFTEN!

MAKE INTEGRATION VERY FREQUENT

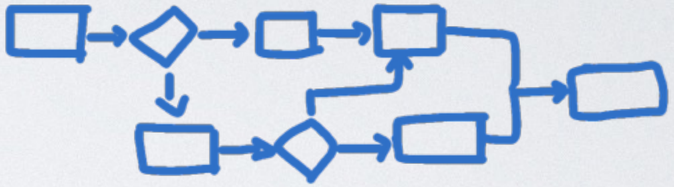
- Doing one big task at once is way more painful than doing **many small tasks**.
- **Frequent feedback** gives us a chance to make the same task iteratively less painful.
- Doing a task repeatedly **trains** you and makes you better at it. It also encourages you to automate (some of) the process.



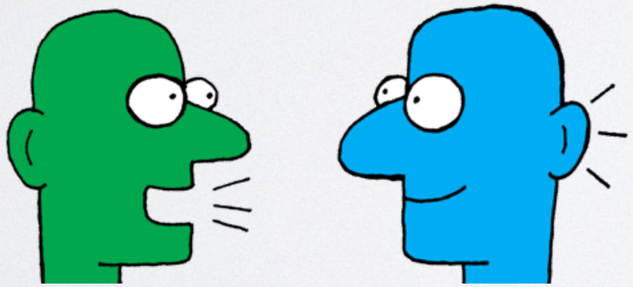
http://martinfowler.com/bliki/FrequencyReducesDifficulty.html

HOW DO WE DO IT?

Tools



Processes



Communication

Culture

Discipline

IN PRACTICE

- Maintain a code repository
- Everyone commits to the baseline every day
- Every commit (to baseline) should be built
- Automate the build
- Make the build self-testing
- Everyone can see the results of the latest build (feedback)

INFORMATION RADIATORS

http://www.infoq.com/articles/agile-software-cockburn-book-2ed

https://panic.com/statusboard/

https://www.atlassian.com/wallboards/information-radiators

CONTINUOUS INTEGRATION

Your Team is happy.

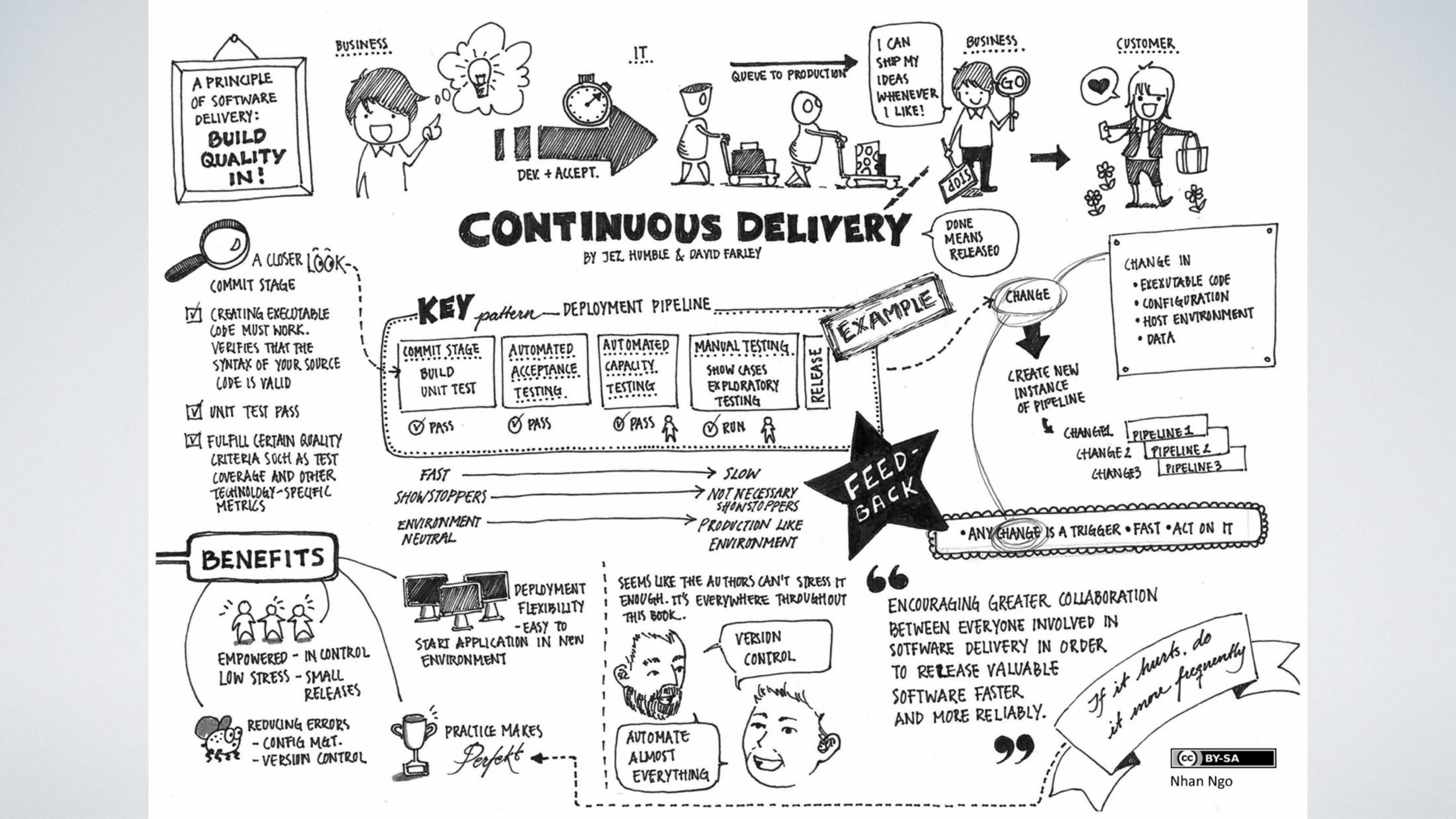
Everyone is pushing commits very frequently. We know that we are always able to build a complete software system with all the components. We even have some unit tests.

Of course, problems do happen. But we identify them quickly. Not 2 hours before an important demo or 2 days before launch.

By the way, setting up our continuous integration server was actually easy.

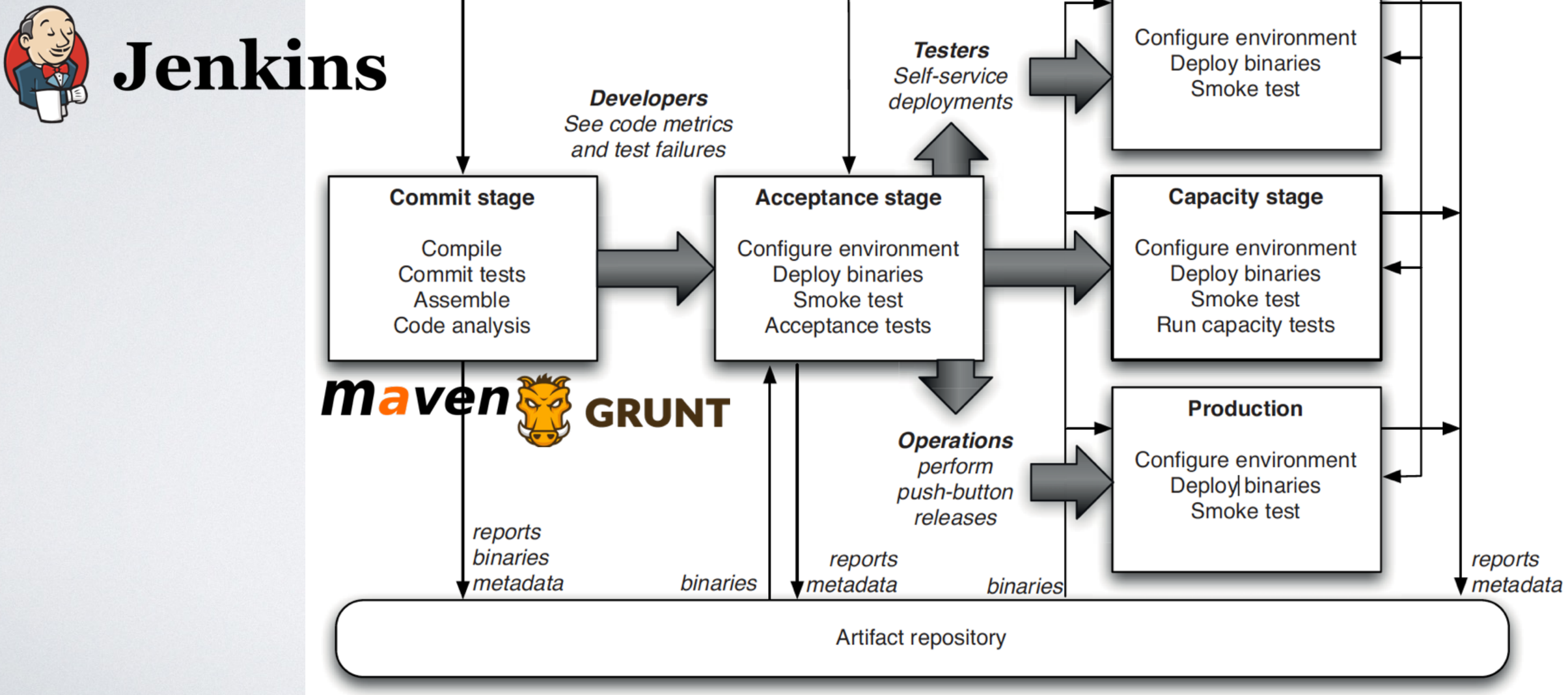
DON'T STOP THERE

- **Continuous integration is only a first step**
- **Your team might be happy, but your customers don't care**
  - A working software binary built on a daily basis and its components work together.
  - Your customers don't have access to this binary.


A PRINCIPLE OF SOFTWARE DELIVERY: BUILD QUALITY IN!
BUSINESS
IT
DEV. + ACCEPT.
QUEUE TO PRODUCTION
I CAN SHIP MY IDEAS WHENEVER I LIKE!
BUSINESS
GO
STOP
CUSTOMER
CONTINUOUS DELIVERY
BY JEZ HUMBLE & DAVID FARLEY
DONE MEANS RELEASED
A CLOSER LOOK
COMMIT STAGE
CREATING EXECUTABLE CODE MUST WORK. VERIFIES THAT THE SYNTAX OF YOUR SOURCE CODE IS VALID
UNIT TEST PASS
FULFILL CERTAIN QUALITY CRITERIA SUCH AS TEST COVERAGE AND OTHER TECHNOLOGY-SPECIFIC METRICS
KEY pattern — DEPLOYMENT PIPELINE
COMMIT STAGE
BUILD
UNIT TEST
AUTOMATED ACCEPTANCE TESTING.
AUTOMATED CAPACITY TESTING.
MANUAL TESTING.
SHOW CASES EXPLORATORY TESTING
RELEASE
PASS
PASS
PASS
RUN
EXAMPLE
CHANGE
CHANGE IN
• EXEXUTABLE CODE
• CONFIGURATION
• HOST ENVIRONMENT
• DATA
CREATE NEW INSTANCE OF PIPELINE
CHANGE1
CHANGE 2
CHANGE3
PIPELINE 1
PIPELINE 2
PIPELINE 3
FAST
SLOW
SHOWSTOPPERS
NOT NECESSARY SHOWSTOPPERS
ENVIRONMENT NEUTRAL
PRODUCTION LIKE ENVIRONMENT
FEED-BACK
• ANY CHANGE IS A TRIGGER • FAST • ACT ON IT
BENEFITS
EMPOWERED - IN CONTROL
LOW STRESS - SMALL RELEASES
DEPLOYMENT FLEXIBILITY - EASY TO START APPLICATION IN NEW ENVIRONMENT
REDUCING ERRORS
- CONFIG MGT.
- VERSION CONTROL
PRACTICE MAKES Perfekt
SEEMS LIKE THE AUTHORS CAN'T STRESS IT ENOUGH. IT'S EVERYWHERE THROUGHOUT THIS BOOK.
VERSION CONTROL
AUTOMATE ALMOST EVERYTHING
ENCOURAGING GREATER COLLABORATION BETWEEN EVERYONE INVOLVED IN SOTFWARE DELIVERY IN ORDER TO RELEASE VALUABLE SOFTWARE FASTER AND MORE RELIABLY.
If it hurts, do it more frequently
BY-SA
Nhan Ngo

	Continuous Integration	Continuous Delivery
Who	Developer centric	Whole team Business/QA/Dev/Ops
Output	**software package**	**feature available to users**
Scope of quality control	Narrow: Unit / integration tests	Broad: unit, integration, UX, perf, security, etc.
Complexity / cost	Low	High to very high
Impact on agility	Medium	High to very high

QUALITY - PIPELINES - THE LAST MILE


Jenkins
Source code
git
Env & app config
Version control
Env & app config
UAT
Configure environment
Deploy binaries
Smoke test
Testers
Self-service deployments
Developers
See code metrics and test failures
Commit stage
Compile
Commit tests
Assemble
Code analysis
Acceptance stage
Configure environment
Deploy binaries
Smoke test
Acceptance tests
Capacity stage
Configure environment
Deploy binaries
Smoke test
Run capacity tests
maven
GRUNT
Production
Configure environment
Deploy binaries
Smoke test
Operations
perform push-button releases
reports
binaries
metadata
binaries
reports
metadata
binaries
reports
metadata
Artifact repository

TEAM WORK

- **Get in groups and choose a project, which included a software component. Discuss and prepare a 5' presentation (no need for slides).**
  - **Introduce the context:** product, team, stage
  - **Is continuous integration an established practice in the projet?** If **yes** explain what is done, what works well and what are the limits. If **no**, explain why.
  - Same question for **continuous delivery**.
  - What are **1 or 2 questions** that you have on the topic (can be technical or not)?

WRAP UP

- **How do we get started?**
- **Tools**
- **Resources**

GETTING STARTED: BABY STEPS!

- **Step 1: put your code in a version management system**
  - If you don't know git, invest the time to learn it. Don't rely on trial and error... Pick a simple branching strategy.
  - GitHub and BitBucket make your life easier.
- **Step 2: build your components automatically**
  - The tooling will depend on your technology stack: maven for Java, Grunt/gulp for JS, etc. These are established tools with huge communities.
  - Setup a build server. Whether is is a machine in your lab or a VM in the cloud, you need a place where automated builds are run. Jenkins is one of the popular choices.
- **Step 3: test your components automatically**
  - We will talk about that next week!

TOOLS


Jenkins

Bamboo

TeamCity by JetBrains

go Continuous Delivery


Travis CI

maven


GRUNT

GitHub


Bitbucket


GitLab

RESOURCES

- **continuousdelivery.com**
- **www.thoughtworks.com/continuous-delivery**