JConf Mexico 2020

3 rd October 2020

Continuous Stupidities - Did DevOps fail?

The Author

Marco Schulz ElmarDott

studied at HS Merseburg, Germany, computer science and holds an engineers degree in software engineering. The main topics in his field of work are Software Architectures, automatism of the software development process and Software Configuration Management. Since more than 15 years he work in different large Web Application Projects. Currently he work as independent Consultant, Trainer and publish plently articles in several computer magazines. Mail: marco.schulz@outlook.com

+ Project Manager + Consultant + Writer + Speaker + Trainer +

Agenda

- What is DevOps?
- Semantic Versioning
- Releases, Deployments, Delivery
- 5 Ways to tear down team productivity
- Lesson learned

We do what we can!
But, can we what we do?

> “A fool with a tool, is still a fool”
> Gardy Booch

Process automation reduce the risk of failures, but **high complex processes are often hard to automate**.

DevOps: Tool or Culture?

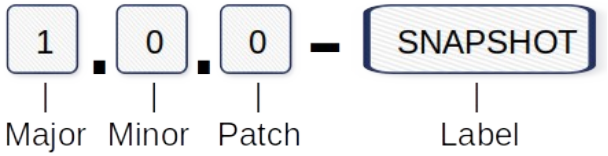
First time mentioned in the book “Continuous Delivery” by Jez Humble in 2011

Academics and practitioners have not developed a unique definition for the term "DevOps"

> “A set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality”
>
> Len Bass, Ingo Weber, and Liming Zhu

Semantic Versioning

Given a version number [MAJOR . MINOR . PATCH], increment the:

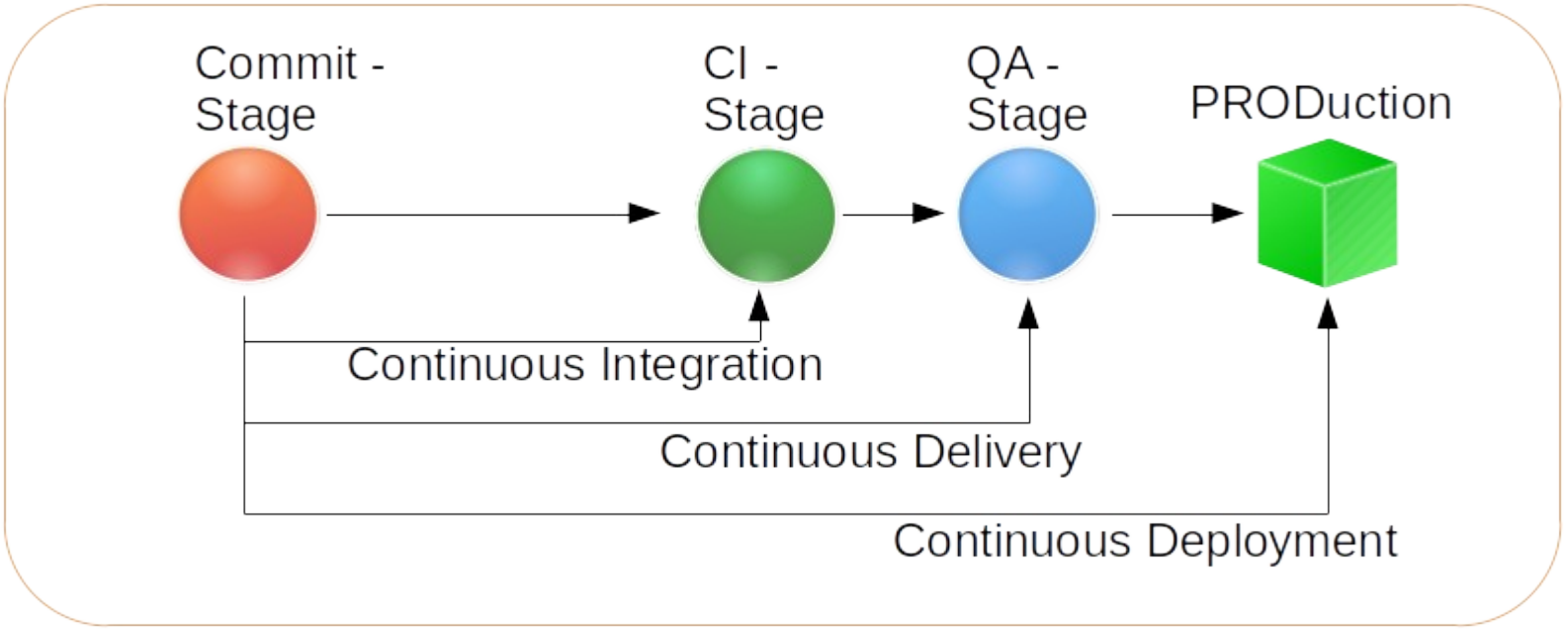


1. **MAJOR** - when you make incompatible API changes,
2. **MINOR** - when you add functionality in a backwards-compatible manner, and
3. **PATCH** - when you make backwards-compatible bug fixes.

Additional labels for pre-release and build metadata are available as extensions to the MAJOR.MINOR.PATCH format.

Source: http://semver.org

From Release to Delivery

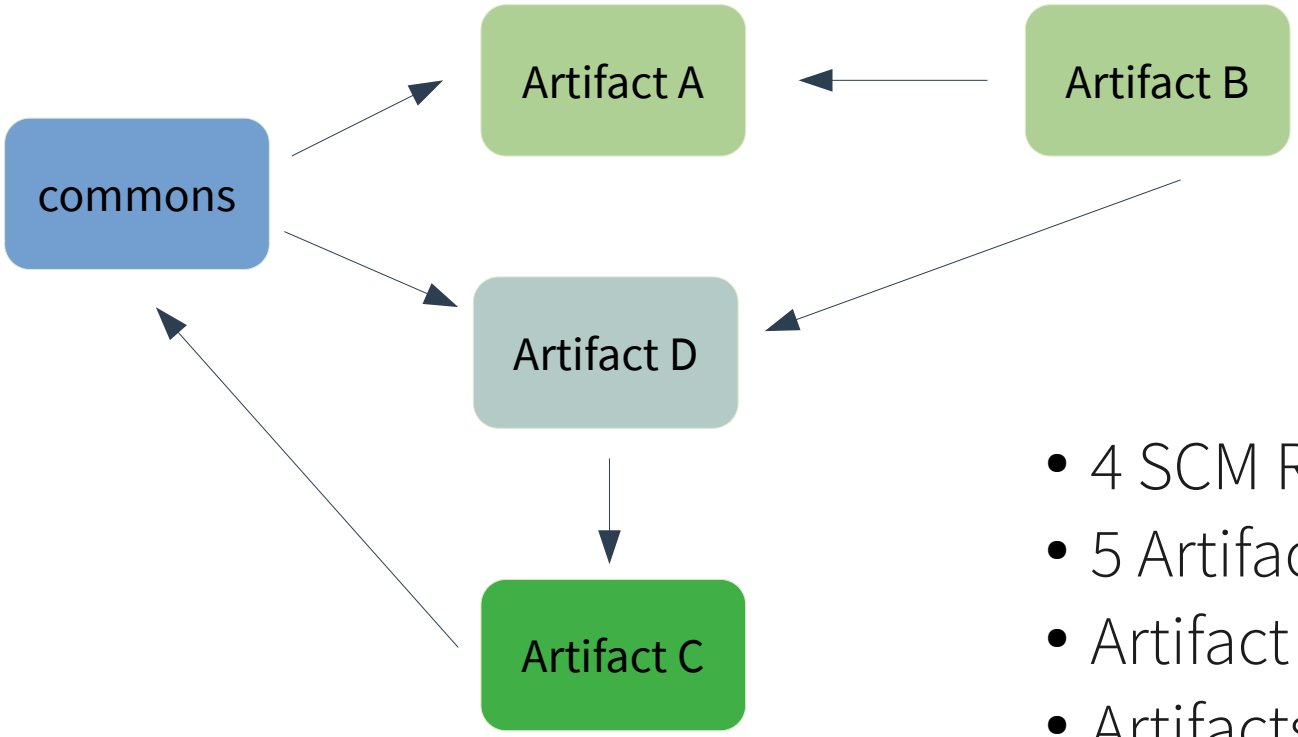


Release : package

Deploy : storage

Deliver : install

I. Build Logic vs. Architecture



- 4 SCM Repositories
- 5 Artifacts
- Artifact B define an API
- Artifacts A & B is multi module

How to fix your build?

Build Logic can’t fix architecture gaps

- Build Logic is written by developers – not by administrators
- Follow the KISS principle
- Dependency chains marks monoliths
- Orchestrated builds are evil
- Don’t do everything at once

II. Single Point of Failure

- Choose the proper tool for your programming platform
- Know common problems
- Avoid own solutions when standards exist

Example: Version Number

src/main/resources/org/europa/together/configuration/**version.properties**

```
01: version=${project.version}
```

---

src/main/java/org/europa/together/utils/**Constraints.java**

```
029: public static final String MODULE_VERSION = getVersion();

099: private static String getVersion() {
100:     String filePath
101:         = "org/europa/together/configuration/version.properties";
102:     PropertyReader propertyReader = new PropertyFileReader();
103:     propertyReader.appendPropertiesFromClasspath(filePath);
104:     return propertyReader.getPropertyAsString("version");
105: }
```

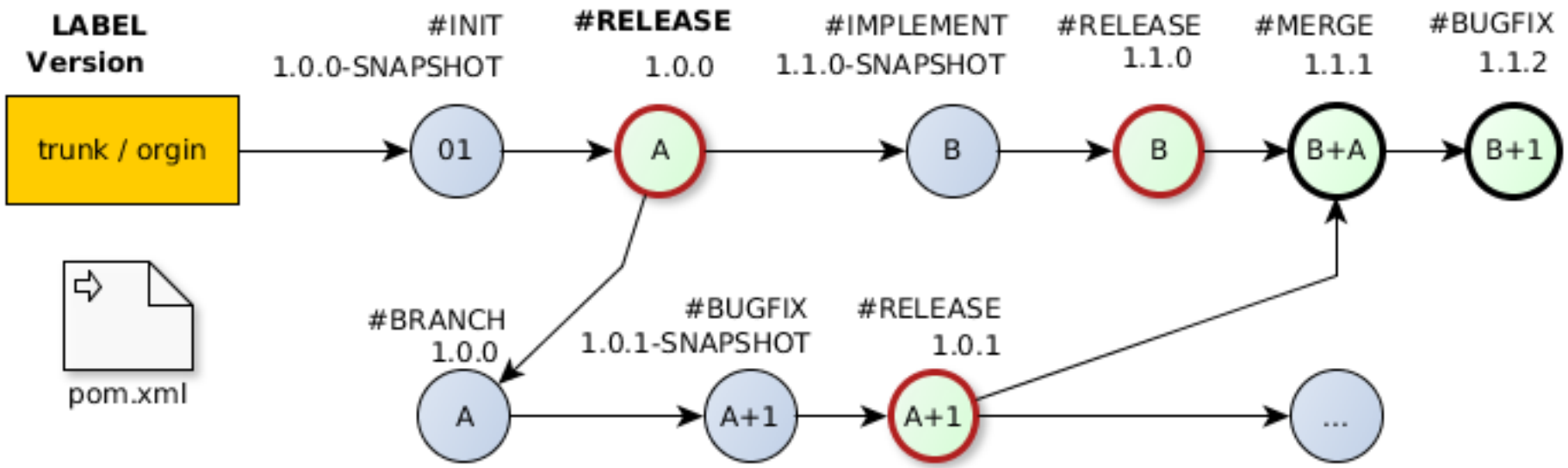
Source: https://github.com/ElmarDott/TP-CORE

III. Source Control Management disaters

> ”We are a team of four senior developers (by which I mean we’re all over 40 with 20+ years each of development experience) and not one of us has had a positive experience in the past with branching the mainline...The branch is easy - it’s the merge at the end that’s painful.”

Shaun Phillips, Jonathan Sillito, Rob Walker, 2011, Branching and merging: an investigation into current version control practices

Secrets of merge conflicts



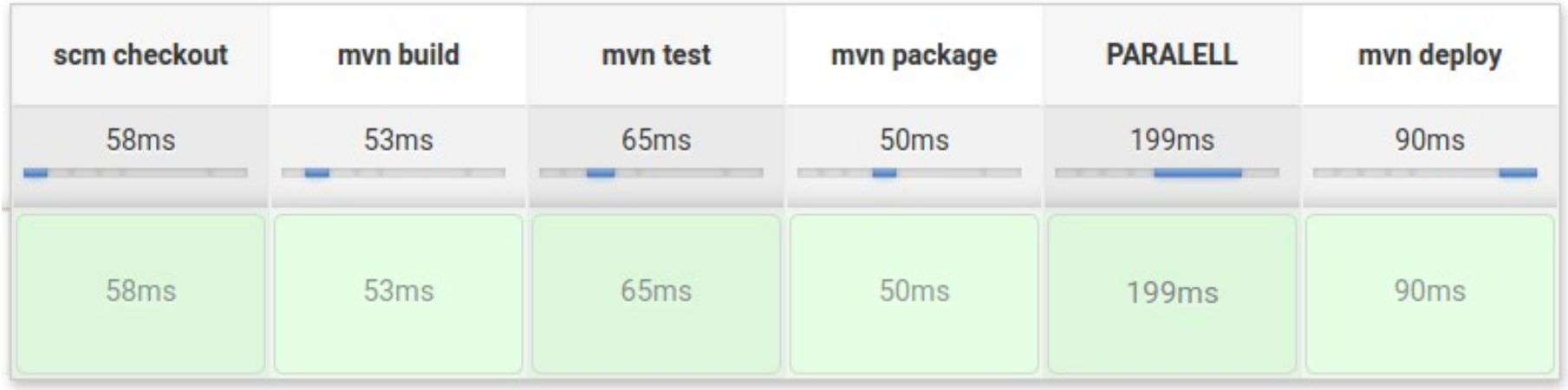
IV. Automation Server myths

Jenkins pipeline plugin with groovy DSL

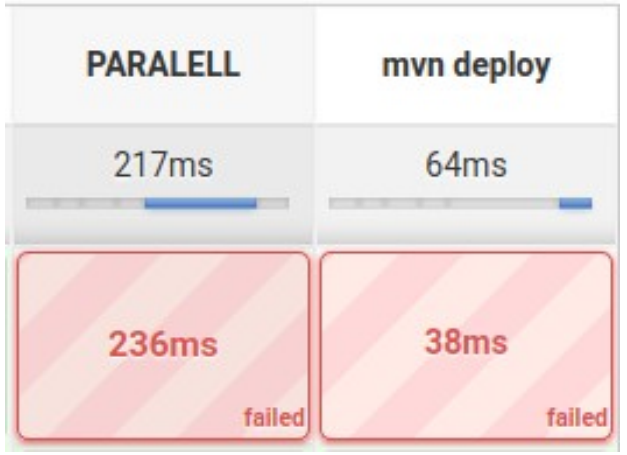
scm checkout	mvn build	mvn test	mvn package	API Doc	sonar checks	OWASP checks	mvn deploy
118ms	88ms	82ms	90ms	81ms	63ms	73ms	51ms
118ms	88ms	82ms	90ms	81ms	63ms	73ms	51ms

- CI Servers aren't management tools
- Don’t exaggerate with containerization
- Execution steps should unique
- Build logic have to run also on local machines
- Workspace clean before execution not after

Parallel universe




```
stage('PARALLEL') {
    steps {
        parallel (
          "API Doc" : {
            echo 'API DOC'
          },
          "sonar checks" : {
            echo 'SONAR'
          },
          "OWASP checks" : {
            echo 'OWASP'
          }
        )
    }
}
```



V. the truth about testing

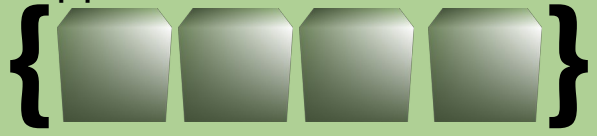
MVN build lifecycle

test	**verify**
Unit Tests	Integration Tests
surefire-plugin	failsafe-plugin
ClassTest.java	ClassIT.java

BDD Tests

JGiven

application



Integration Test for team collaboration

Jenkins

other test aspects:

- UI test
- End 2 End test
- Penetration test

Lessend Learnd

- A well structured architecture simplify your life
- SCM is a collaboration tool – not a quality gate
- Pull Request should not a solution for enterprise teams
- Don’t create high sophisticated own solutions
- Follow common standards
- Focus on problems not on tools
- Prefer always a decentralized solution instead of centralism
- Avoid perfectionism

Failures becomes as problems if you can’t detect them fast.

Thank you / Danke / Gracias