ONE LOGIC

ONE LOGIC –
DATA SCIENCE FROM GARAGE TO PRODUCTION

Continuous Integration for Everybody

21.06.2018

Thomas Stieglmaier

About Me

Thomas Stieglmaier

B. Sc. Internet Computing / M. Sc. Computer Science @ Universität Passau

Working Student @ Software Systems Lab

Since 2 Years: Software Developer @ ONE LOGIC

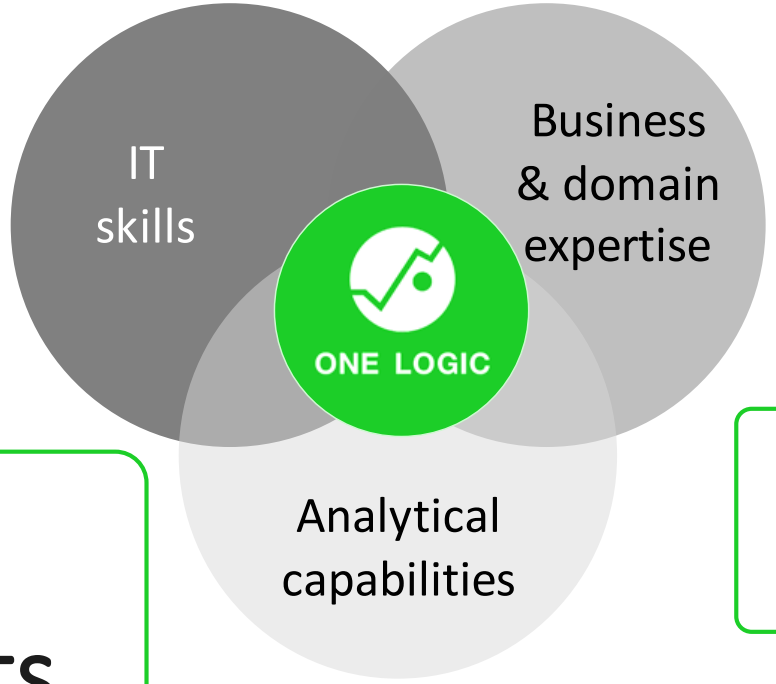
Thomas ♥ Software Engineering

About Us

Software Development

Java, C#, JavaScript
Android/iOS

Main Product: ONE DATA



Data Analysis

Using ONE DATA (partially with R / Python)

Wanted:
Working Students
& Junior Developers

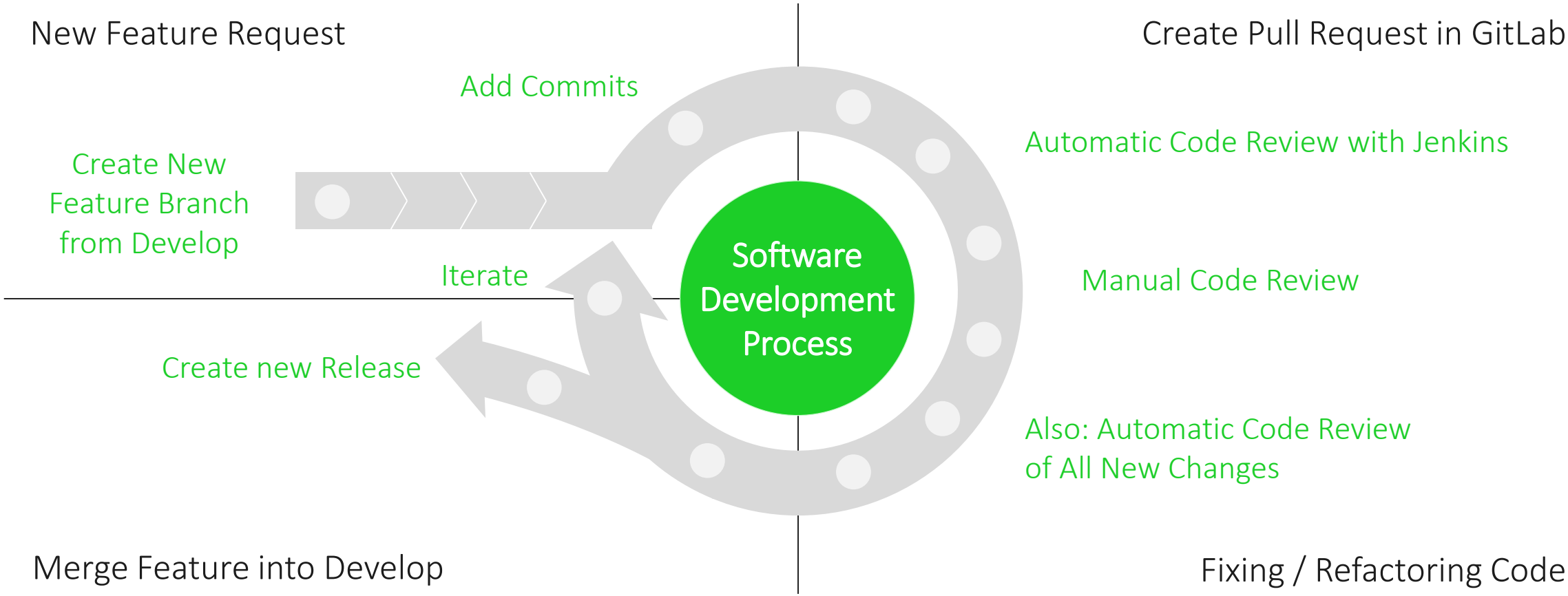
Overview

Easy reproducible builds

Nicer Code / Easier Code Reviews

Automating Builds + Code Reviews

Our Software Development Process (git flow)



Maven – A Tool for Software Project Management

IDE Integration

Project Description

Dependency Management

Code Generation

Code Quality Tools

- Junit / Coverage
- Spotbugs
- Errorprone

Deployment

```
<distributionManagement> groupId>
  <repository> apache.maven.plugins</groupId>
    <id>nexus</id> -compiler-plugin</artifactId>
    <name>Corporate Repository</name> >
    <url>http://nexus.url:8081/repository/maven-public</url>
  </repository> javac-with-errorprone</compilerId>
</distributionManagement>howWarnings> ></goals>
    <forceJavacCompilerUse>true</forceJavacCompilerUse>
    <source>${java.version}</source> otations</outputDirectory>
    <target>${java.version}</target> de_tests.xml</excludeFilterFile>
    <useIncrementalCompilation>false</useIncrementalCompilation>
    <compilerArgs> d>
      <compilerArg>-Xep:ParameterName:OFF</compilerArg>
      <compilerArg>-XepDisableWarningsInGeneratedCode</compilerArg>
    </compilerArgs> /goal>  iler-plugin</artifactId>
  </configuration> d>com.querydsl</groupId>
  <dependencies> factId>querydsl-apt</artifactId>
    <dependency> ion>${querydsl.version}</version>
      <groupId>org.codehaus.plexus</groupId>
      <artifactId>plexus-compiler-javac-errorprone</artifactId>
      <version>2.8.2</version>
    </dependency>
  </dependencies>
</plugin>
```

Code Quality – Software Environment

Use google errorprone as compiler instead of javac

Use static code analysis tools: SpotBugs / PMD / Checkstyle / …

Enforce Coding Standards:

- Project-wide code formatting guidelines: google-formatter / synced eclipse formatter settings
- System.out.println only in very specific subpackages (if at all)
- Do not use shaded dependencies of your dependencies
- Have proper javadocs for all public interfaces

Collect these metrics to see how your project evolves and where you need to improve.

Code Quality - Development

Use libraries that make your life easier

- Application Code
  - Lombok (removes boilerplate code)
  - Guava (immutable collection types, concurrency features, …)
- Unit Tests
  - Truth (fluent assertions)
  - Mockito (mocking framework)
  - EqualsVerifier (no self-written unit-tests for hashcode and equals!)
  - (ToStringVerifier)

Jenkins

GitLab

Continuous Integration - GitLab CI

Requirements:

- Support for build system
- Integration into our git infrastructure (GitLab, Github, …)
- A server for running the build/test tasks

Pros:

- Works instantly (AutoDevOps)
- 2000 CI minutes free for everyone
- Almost no configuration necessary

Cons:

- No UI showing detailed failure messages (besides the log)

Continuous Integration - Jenkins

Requirements:

- Support for build system ✓
- Integration into our git infrastructure (GitLab, Github, …) ✓
- A server for running the build/test tasks

Pros:

- Configurable UI showing detailed messages
- Analysis results queryable via REST calls
- Fine-grained CI configuration capabilities

Cons:

- Dedicated server necessary
- Initial configuration overhead

Gitlab CI vs Jenkins configuration

```yaml
image: maven:3-jdk-8

stages:
  - build
  - test

maven_build:
  stage: build
  script: mvn clean package -DskipTests

maven_junit:
  stage: test
  script:
    - mvn test
    - cat target/site/jacoco-ut/index.html

maven_findbugs:
  stage: test
  script: mvn verify -DskipTests
```

```
pipeline {
  agent none

  options {
    // all stages mentioned here have to be executed
successfully, in order to make "Merge When Pipeline
Succeeds" work.
    gitlabBuilds(builds: [
    '00 Pipeline Started',
    '10 Checkout and Merge Branches',
    '20 Clean Project',
    '30 Build & Code Metrics',
    '40 Collect Metrics',
    '50 Add merge request comments'])
    gitLabConnection('https://git.intranet.onelogic.de')
  }
…
…
…
…
```


Cool Story Bro, but I have Questions!

➔ ask me today, or
➔ write an e-mail to thomas.stieglmaier@onelogic.de