SPE 2018 – Lecture 05

A Pragmatic Introduction to Continuous Integration

Dr. Daniel Schien

Daniel.schien@bristol.ac.uk

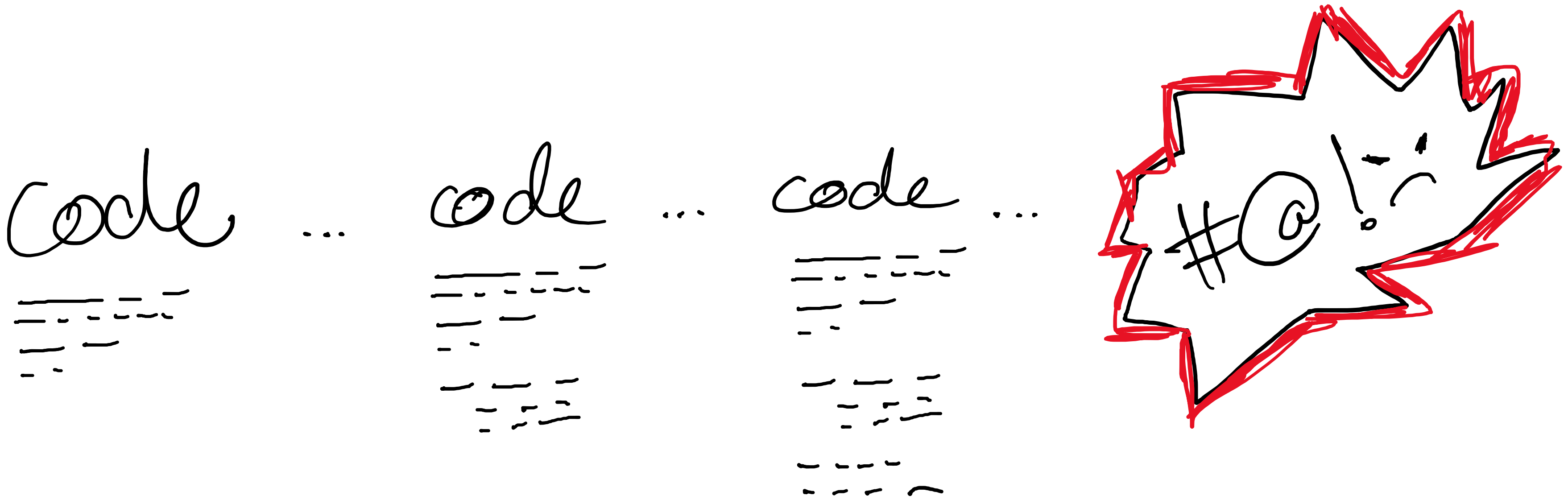
Recap

- Week 1 Introduction & The Open Project
- Week 2 Introduction to Agile & Agile Practices
- Week 3 CI & Validation and Verification
- Week 4 Requirements I & Requirements II

Le Menu

- Overview of Continuous Integration
- Live Demo

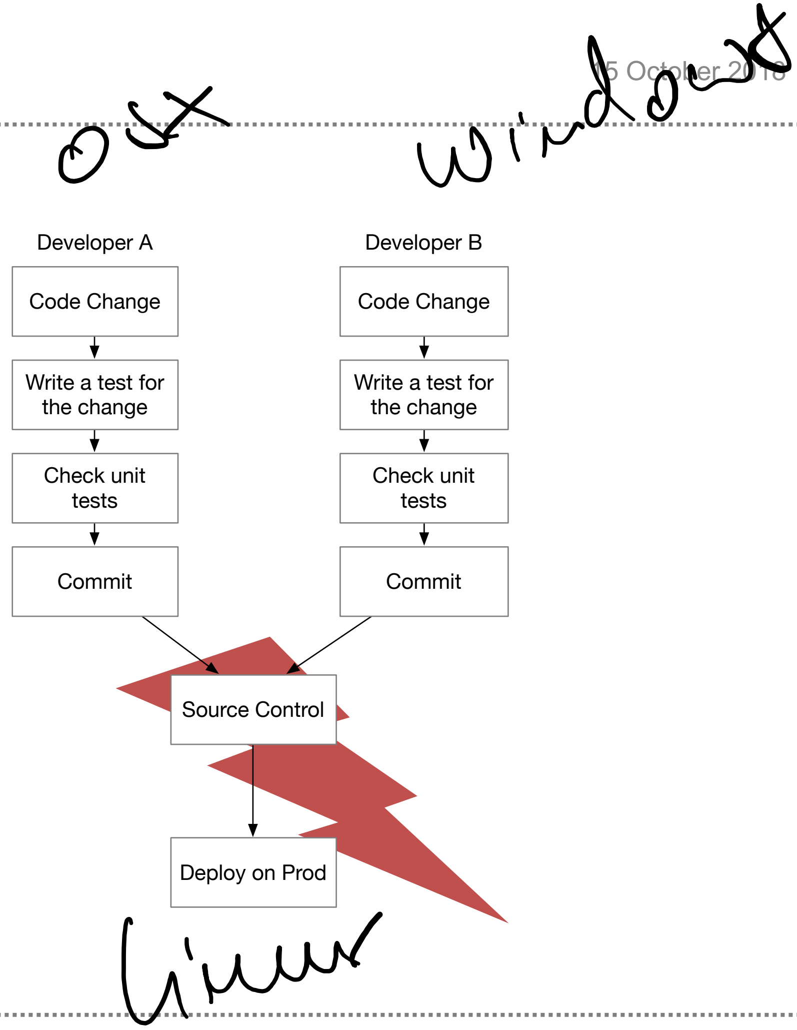
The Concepts


code ...
code ...
code ...
#@!

~~Never change a running system~~

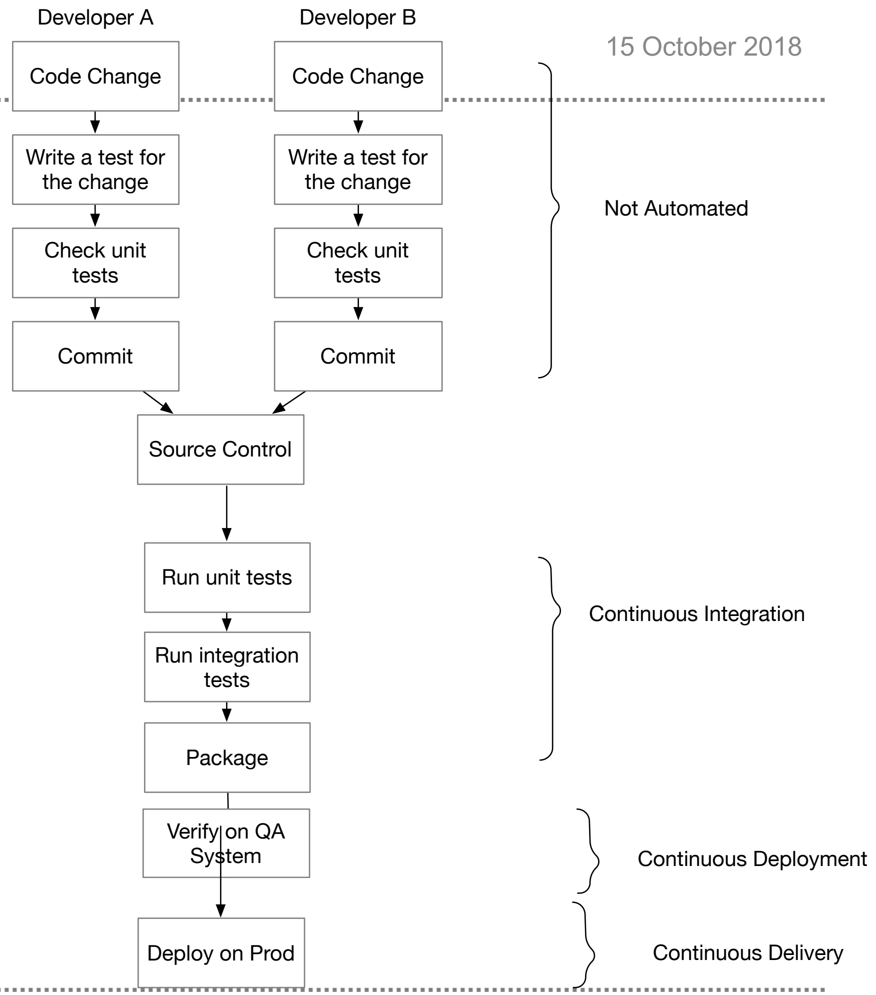
change is inevitable

Working in a team



Levels of Automation

- Continuous Integration
- Continuous Deployment
- Continuous Delivery



Advantages of CI CD CD

- Reduced costs
  - Testing is expensive
  - Manual regression test are extremely expensive
  - CI acts as a driver to increased test coverage
  - Avoid many trivial failures

More Advantages of CI CD CD

- Faster time-to-market
  - Increases organisational agility
- Higher quality software
  - Closed loop provides better feedback to developers on code quality
  - Coverage, typical problems, people
- Rapid feedback to developers and business

The Preliminaries

Secret Sauce I: Source Control

- Essential to CI
  - Central place to get the authoritative version from for the CI
- Lots of advantages
  - Annotated Version History
  - Branching
  - Backups
  - Tags
  - Enables collaborative development on code
  - Easy diffing

https://git-scm.com/book/en/v1/Getting-Started-About-Version-Control

Source Control II

- Different Systems
  - CVS
  - SVN
  - Git
  - Mercurial
- Cloud Services
  - Github
  - Gitlab
  - Bitbucket
  - etc

Secret Sauce 2: Automate Builds / Build Tools

ant

maven

gradle

- Organise your build commands into named tasks
- Tasks can be chained
- Typical tasks for a build tool
  - Resolve dependencies
  - Compile
  - Run unit tests
  - Package
- Additional tasks
  - Code Coverage of unit tests
  - Style-checking (linting)
  - Static code analysis

Maven

- Maven central
- local repository (.m2)
- Pom.xml config file
- goals "tasks" (e.g. exec:java)
- Phases - in sequence as part of the development life cycle
  - Compile, test, package, deploy

```
<project xmlns="">
  <modelVersion>4.0.0</modelVersion>
  <groupId>net.spe</groupId>
  <artifactId>lecture-05-ci</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>Lecture 5</name>
  <url>spe-hub.net</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.1.1</version>
      <scope>test</scope>
    </dependency>

  </dependencies>
</project>
```

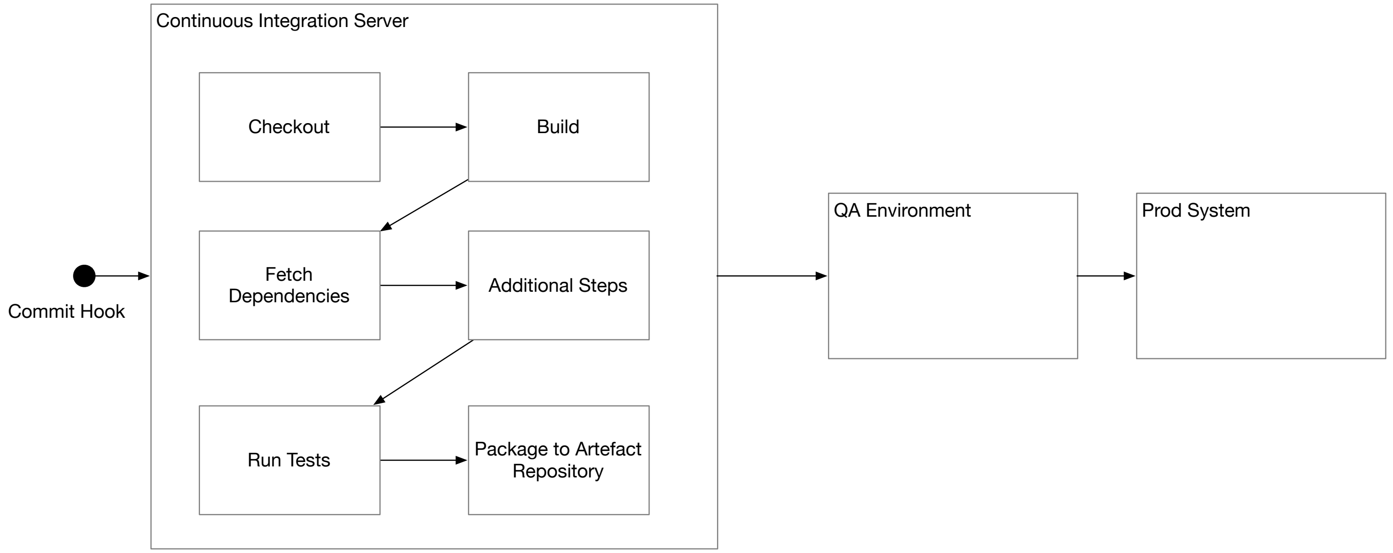
Final Ingredient: Continuous Integration Server

- Integrates with VCS server
- Watches for changes
- Scripted build - Runs always exactly the same tasks
- Build history

The Workflows

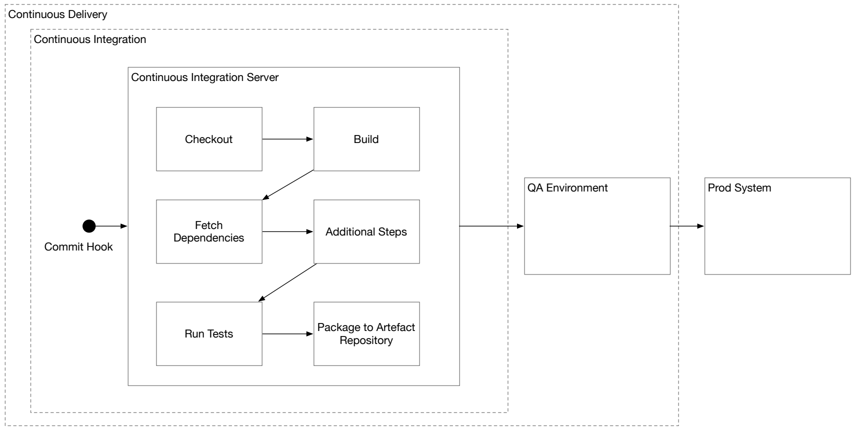
Continuous Integration

- Every code change on the central repo triggers a build



Continuous Delivery

- Automatic testing in the QA environment
- Organisational consequences
- No manual tests any more
- End-to-end tests automated
- There are no release branches
- Passing builds in the QA environment are ready to release to prod

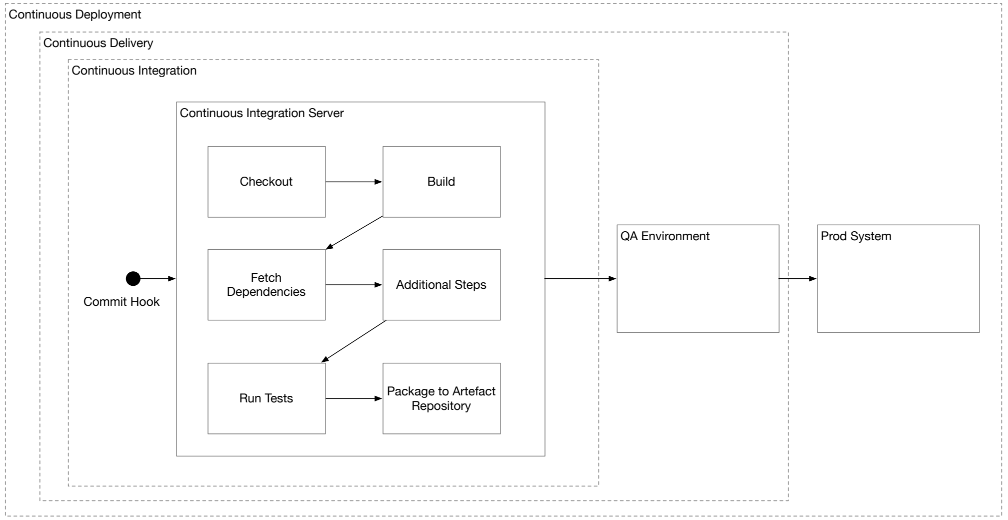


Continuous Delivery

- Deployment to prod is still manual
  - Regulatory
  - Manage critical time periods
  - Coordinate with other departments

Continuous Deployment

- Next step on the organisational maturity ladder
- Operations teams



Canary deploy

- Phased rollout to subset of users
- Monitor
  - Server metrics
  - Application metrics
  - Enables A/B Testing
- Rollback

Demo Time

- mvn archetype:generate -DgroupId=net.spe -DartifactId=mvn-test -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

Thank you for your attention