infrastructure **Services**

Driving iT

Building a Continuous Integration Pipeline with Open Source Tools

February **2016**

Paul Mayne
Software Tools Engineer

Shane O'Hanlon
Software Tools Engineer

Open Source at Allstate

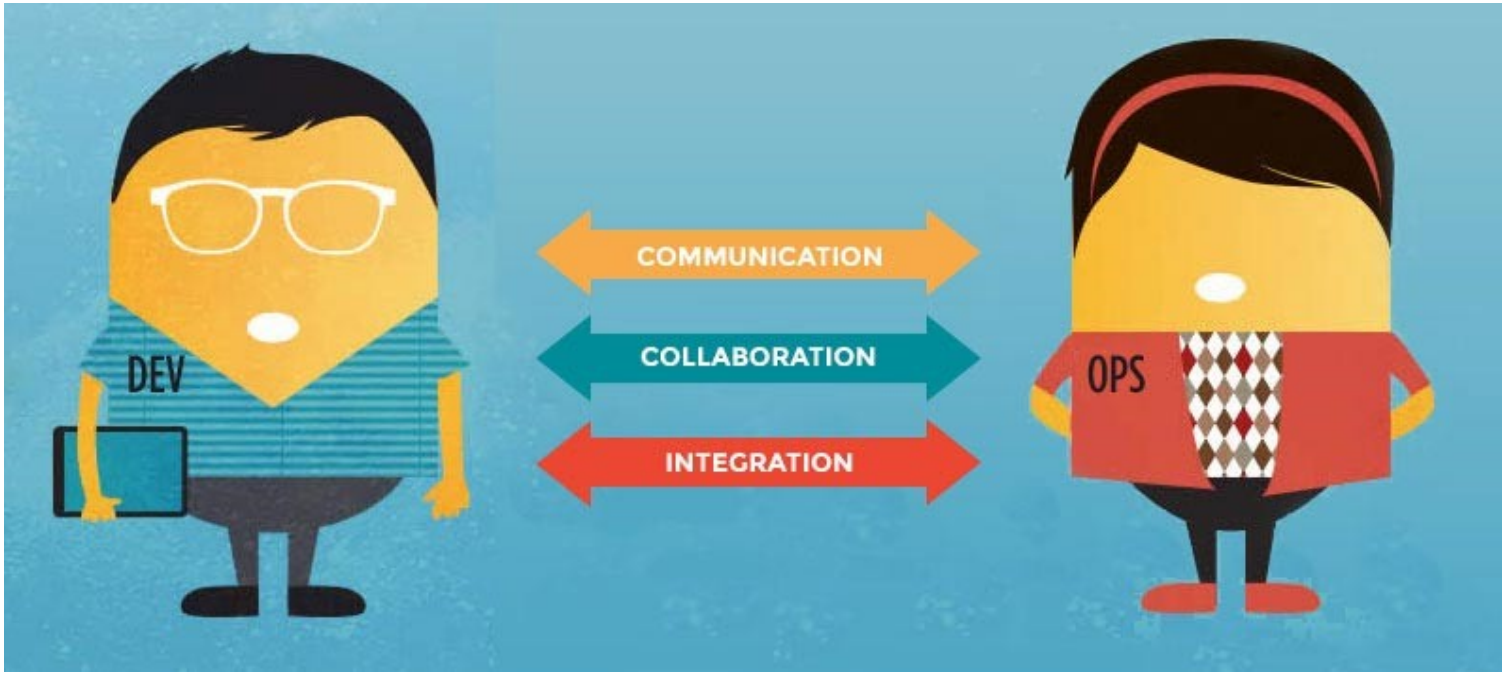
Allstate first considered the use of Open Source tools late in 2013 for the purposes of Continuous Integration & Continuous Delivery.

We know Open Source promotes universal access via a free licence model, but as much as licensing costs are a major factor, equally a key driver for Allstate is the fact that licensing costs actually narrow our ability to recruit new employees.

By shifting left towards Open Source we actually increase the pool of top talent to recruit from and reduce training costs.

A background in DevOps ...

What is DevOps? This is a difficult question to answer, but I will attempt to demystify this movement within the IT industry and how Open Source Tools play a major part in it.



As the IT industry adopted Agile **development** methodologies, this allowed for much faster iteration cycles but if the cadence of the deployment and release cycle is slower, a bottleneck is formed within **operations**. Enter DevOps with the development and operations working together using **automation** tools that allow for faster end-to-end cycles.

.. and Continuous Integration?

One concept of DevOps is the idea of considering the whole process as **pipelines** and using technology to automate wherever a bottleneck occurs so that the software can flow in a **continuous** manner.

Continuous Integration (CI): Is the development practice that requires developers to **integrate** code into a shared repository several times a day. As they check-in their code, it is verified by an automated build process, allowing teams to detect problems early.

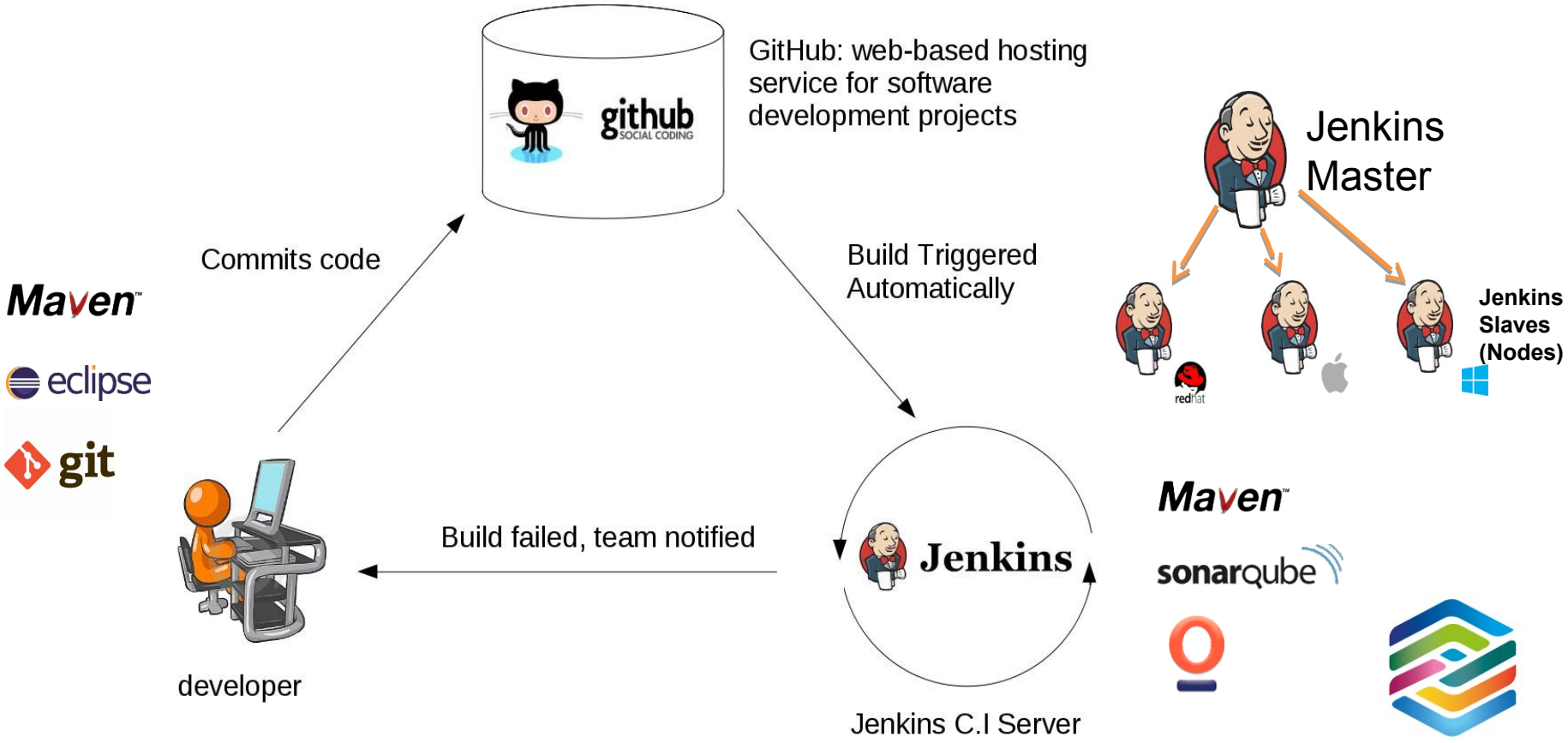
Continuous Delivery (CD): Is a software development discipline where you build software in such a way that the software is **delivered to a state** that can be released to production at any time.

Continuous Deployment (CD): Is the next step on from Continuous Delivery where any change that passes the automated tests is **deployed to production automatically**. Continuous deployment could be considered the end goal but for some companies they will be constrained by regulatory or other requirements.

There are no set rules but the more automation you put into your process the quicker you can deliver changes to production.

The Continuous Integration Pipeline

Now we understand some of the fundamentals of DevOps, we will take a look at a simple Continuous Integration pipeline to see what tools are involved. It's worth pointing out GitHub itself is not Open Source but is widely used to host public repositories to share Open Source software.



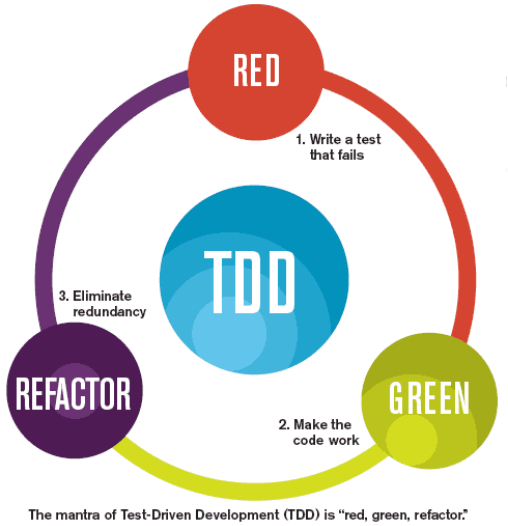
Test-Driven Development

Test-driven development (TDD) is a software development process where first the developer writes an (initially failing) test case that defines the new functionality, then the code is written to pass that test and finally refactors the new code to acceptable standards.

Development (Testing) Tools

Apache Maven is a software project management & comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.

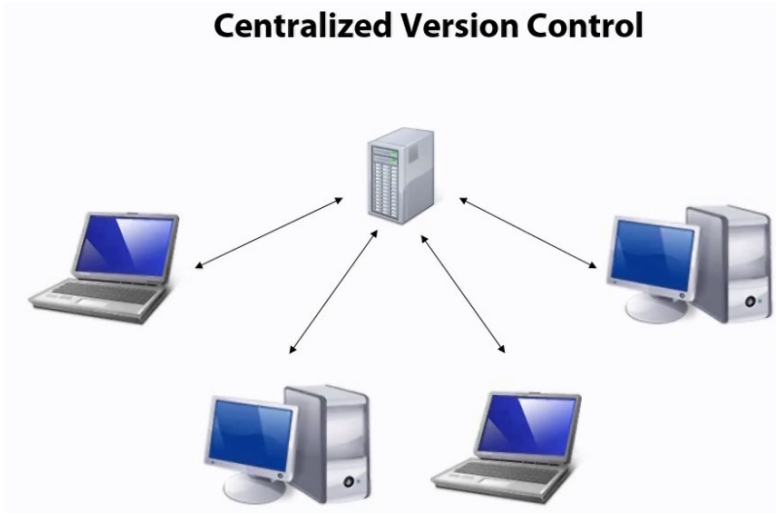
- IDE integration with M2Eclipse/M2E
- Testing and Coding Rule Engines
  - *Junit*
  - *Checkstyle*
  - *FindBugs*
  - *PMD*
  - *Mockito*
  - *JaCoCo*
  - *Macker*

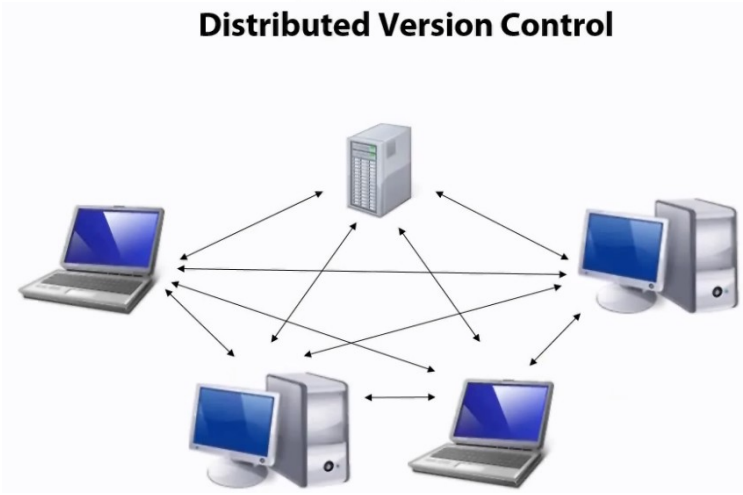


The mantra of Test-Driven Development (TDD) is "red, green, refactor."

Setting the Scene: What is Git?

Git is a Open Source distributed version control system.
Centralised Version Control (CVC) vs Distributed Version Control (DVC)





Git has the idea of local (personal) “shareable” repository

- Commit as often as possible

Some additional operations versus CVC

- Like: Push and Pull

Where are your Build Artifacts?

For most of us, the answer is our builds are **“out in the wild”** because, historically, versioning has been focused on your source code which is fully reproducible from the source code management system. However, the binaries typically lack any traceable or reproducible context.

> If you need to replay last week’s or any build from source code management you may end up with different results due to the dynamic nature of your build process.

artifactory

As DevOps brings about rapid speed to market, allowing us to be more disruptive within the business sector, then traceability and reproducibility provide the assurances for the deployment section of the pipeline and the rollback mechanism. i.e. fix fast.

Technical Debt

Technical Debt refers to the eventual consequences of any system design or development work that takes place within a code base. If left unchecked, technical debt increases software entropy impacting the **code quality** and **maintainability**.

It’s important to understand that Technical Debt can not be avoided and, just like financial debt, needs to be repaid.

Other benefits of managing your Technical Debt:

- Allow new projects to grow faster
- Enable faster iteration cycles
- Assess the impact of planned changes

sonarqube

infrastructure **Services**
Driving iT

Demo:

Continuous Integration Pipeline with Open Source tools