Docker Driven Continuous Delivery

On the gaps between tooling

Jeroen Peeters

A thesis presented for the degree of
Master of Science

Supervised by:
Professor H. Dekkers

The University of Amsterdam
September 2016

I, Jeroen Peeters, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Abstract

tbd.

Acknowledgements

tbd

Table of Contents

Chapter 1

Introduction

tbd.

Chapter 2

Literature review

2.1 Introduction

tbd.

2.2 Tools

2.3 Process

2.4 People

2.5 Methodoloy

Chapter 3

The development organization

In order to understand the problems at the organization it is important to have a deeper understanding of the development organization's structure. The organization is a semi-governmental IT project organization who's mission is to help other (semi-)governmental organizations with IT project management and the realization of projects. They lead by example and help the customer to shape their project according to agile principles. In this thesis we are only concerned with the department responsible for software project realization. Within the Software Delivery (SD) department project teams build software in an agile way. Because some customers are still used to work according to a waterfall approach the department plays an important role in guiding customers. The SD project team helps the customer getting familiar with Agile/Scrum principles in order for them to steer and make decisions about importance of tasks. Before a project ends up at SD it usually follows a pre-development process in which some architectural decisions are already made. This is mostly because governments have to apply to standards and regulations. Usually the software realization team is not involved in this process since the team is not yet in existence. This procedure as described here may vary per project and customer, but it usually applies. When the realization team is formed most of the fundamental decisions have already been taken.

To be able to quickly react to customer needs the development organization relies heavily on external hiring for the duration of a project. Within SD all project members are externals. This gives the organization the ability to quickly scale up or down depending on the number of active projects. However, it also implies that knowledge is easily lost. The organization tries to move people between projects as much as possible in order to retain them. In order to move people more easily between projects and bring new people up to speed more quickly the development phase is standardized within the department as much as possible. The standardization is targeted at process, tools and development frameworks and languages. This standardization is something that can change over time and is defined by SD itself. It is possible for a single project to differentiate from the standard following the “comply or explain”-principle.

The standardized process is based on Continuous Integration and Delivery (CI/CD) principles. In the next chapter we will take a closer look at the CI/CD process.

Wat is code hoe organiseren we het nu laat afhankelijkheden zien tussen applicaties kan je laten zien hoeveel kennis er eigenlijk nodig is om dit voor mekaar te krijgen waarom is het zo.

basic ci setup -> laat zien waar problemen zitten.

specificatie template meta. wat moeten we vastleggen. van elk van de items een voorbeeld uitwerken.

Chapter 4

Stages of Continuous Delivery

In this chapter I describe the different stages of continuous delivery that the development organization went through.

4.1 Stage 1: CI in a shared environment

Characteristics

.	
CI/CD Environment	Shared
Maintenance	System Administrator
Deployment	Manual
Flexibility	Static

- Shared between teams
- Maintenance by system administrators
- Manual maintenance
- Static deployment servers

Systems

Server	Type	Depends on
Subversion	Version Control	
Jenkins	Build Server	Nexus, Sonar, Selenium
Nexus	Artifact Repository	
Sonar	Static Code Analysis	
Selenium		Deployment Server
Deployment Server		Nexus

- Subversion

- Jenkins
- Nexus
- Sonar
- Selenium
- Deployment server

Tools

Tool	Type	Used by	Depends on
Maven	Build	Dev, Jenkins	
Java	Language, platform	Dev, Jenkins	
Custom quality reporting	Reporting	Jenkins	Sonar, Selenium

- Maven
- Java
- Custom quality reporting

Setup

- Setup is done by system administrators

Manual steps

- Create build jobs, (once for every unit of development)
- Configure quality reports, (once for every unit of development)
- Manually maintain configuration of deployment server, (every time server configuration changes)
- Tester requests deployment of new version (direct communication)
- Trigger deployment job, (every time a tester requests a new version)
- Trigger automated tests
- *this list is incomplete*

Problems

- Limited resources per team
- Doesn't scale beyond N teams
- Changes/upgrades affect all teams
- Teams can't install plugins or make changes to the setup
- Teams can interfere with each other causing difficult to debug and random failures
- Teams request changes to the infrastructure from the system administrators
- Might take considerable amount of time
- Request can be denied
- Changes to deployment server are not repeatable
- Difficult to go back N versions of the application and server configuration

4.2 Stage 2: Automated CD in a distributed environment

Characteristics

.	
CI/CD Environment	Per team
Maintenance	Team
Deployment	Automatic
Flexibility	On-demand

- Each team has his own CI/CD environment
- The team is responsible for the environment (DevOps)
- Dynamic deployment cluster
- Application deployment is scripted
- System administrators maintain the deployment cluster
- Teams decide what their CI/CD landscape looks like

Systems

- Gitlab
- Jenkins
- Nexus
- Sonar
- Selenium
- Deployment server

Tools

- Maven
- Docker
- Custom quality reporting

Manual steps

- s1

Problems

- p1

4.3 Stage 3: — next evolution..

tbd..

4.4 Initial situation

! This needs to be placed elsewhere and rewritten !

Figure 4.2 shows the steps and interactions a developer has with build systems in order to deploy a change in the software to a target server.

Figure **??** shows the steps a developer needs to take in order to setup a single source repository and configure the continuous integration pipeline.

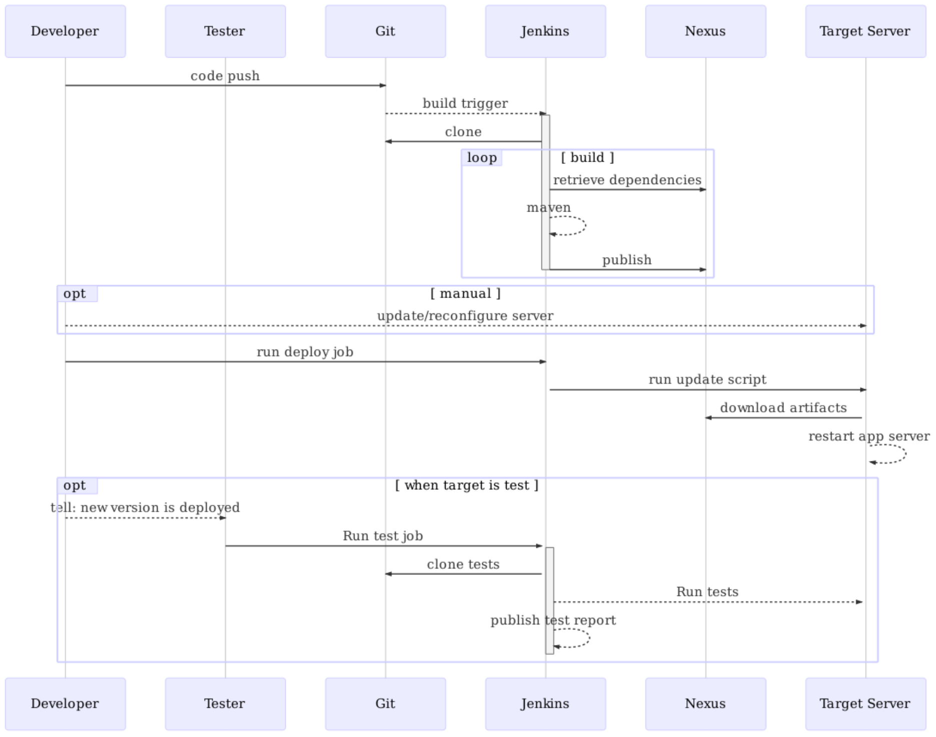

Figure 4.1: Basic CI

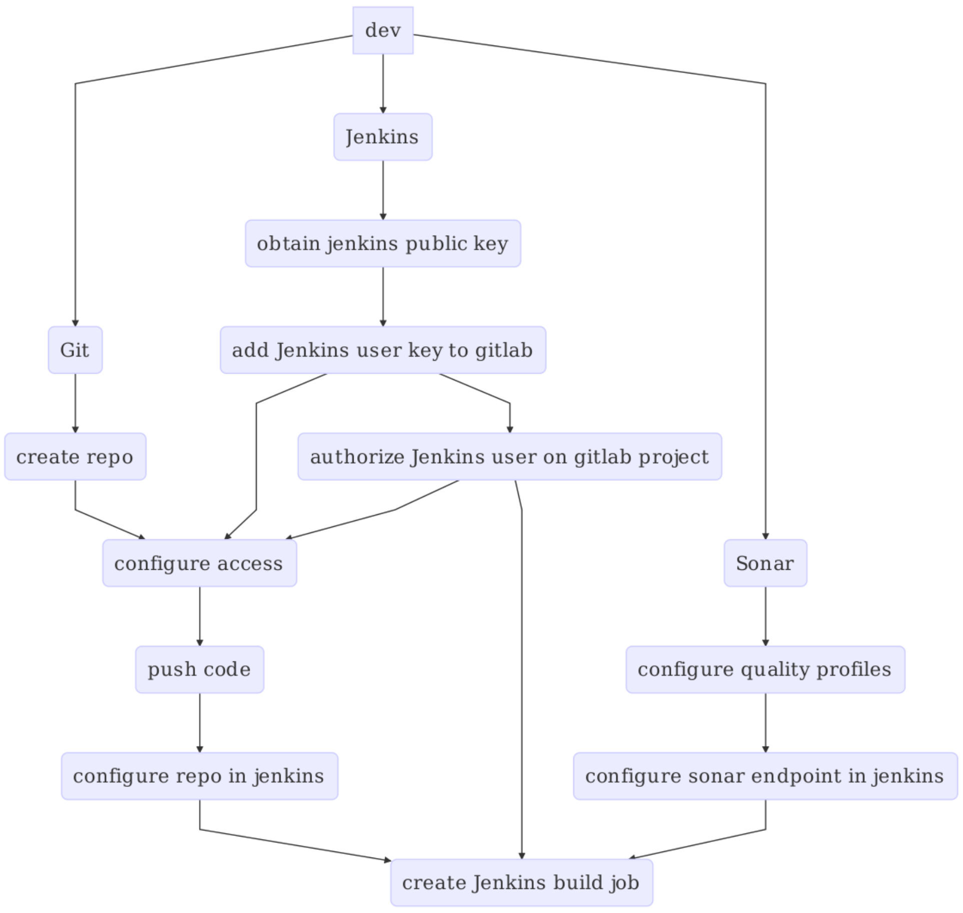
dev
Jenkins
obtain jenkins public key
Git
add Jenkins user key to gitlab
create repo
authorize Jenkins user on gitlab project
configure access
Sonar
push code
configure quality profiles
configure repo in jenkins
configure sonar endpoint in jenkins
create Jenkins build job

Figure 4.2: Basic CI setup

Chapter 5

Problem description

Explain the problem with CI/CD that the company was faced with.

-

The problem description follows naturally from the observations and/or interviews. The problems are categorized according to their origin.

Describe the problems

Chapter 6

Observations / Interviews

Do interviews with a couple of teams to find and clarify on problems.

References