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

In this chapter I will bring forward the current idea's, findings and discussions related to the field of Continuous Delivery in software engineering.

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.

Chapter 4

What is Continuous Delivery

In this chapter I will discuss what people generally understand by the term Continuous Delivery.

4.1 Continuous Integration

Continuous Delivery is the natural evolution of Continuous Integration (CI). Practicing Continuous Integration is an absolute necessity before you can start with Continuous Delivery.

CI focuses on integrating different software branches into a main line. This generally occurs when developers make changes to the main line in their development environments.

To do CI one needs at least the following systems:

1. Revision Control System
2. Continuous Integration Server

Figure 4.1 depicts the dependency relationship between the CI systems.

4.1.1 Revision Control System

The RCS covers the integration of code branches into a main line.

4.1.2 Continuous Integration Server

A CI-server, sometimes referred to as the build server, automatically performs the build process when a code branch is integrated into the main line. This ensures

that the software in the main line can still be build according to predefined rules. Furthermore it ensures that the change doesn't depend on development specific environments, reducing the 'it builds on my machine'-problem. Preferably the CI-server also executes tests to ensure that previous functionality is still intact.

4.2 Continuous Delivery

Since Continuous Delivery (CD) builds on top of CI it reuses its systems.

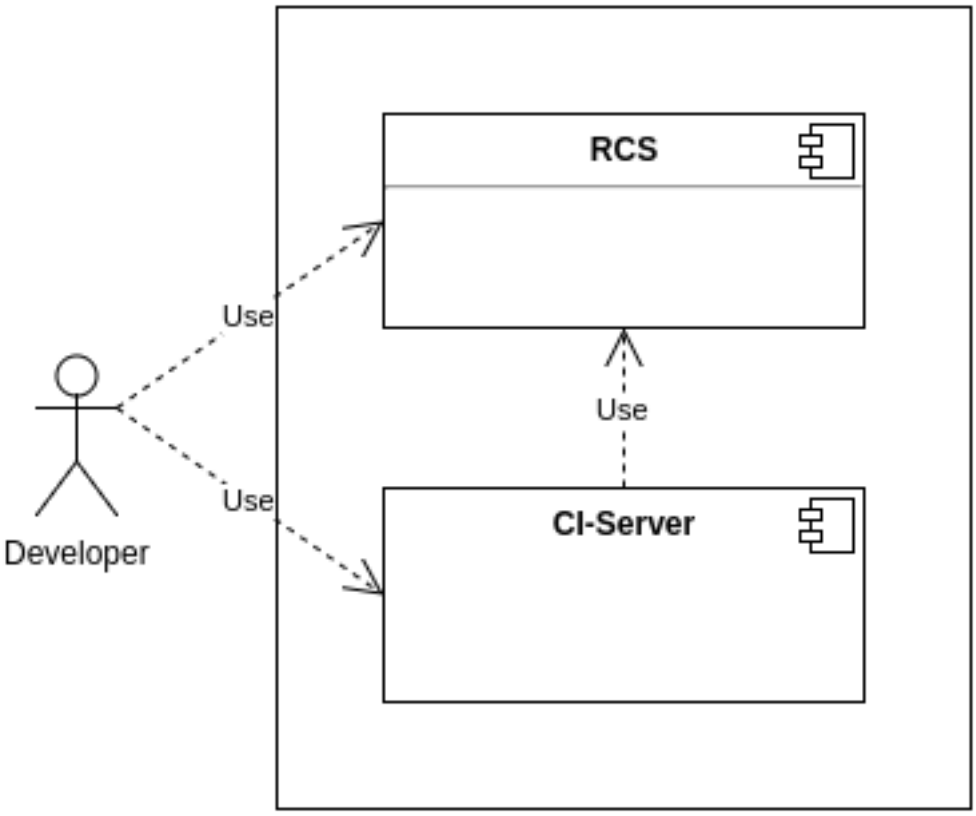

Figure 4.1: Overview Continuous Integration

Chapter 5

Software Development at the Development Organization

TBD

5.1 Project Roles

Within the development organization we distinguish the following roles.

Role	Description
System Administrator	Maintains the (virtual-) network and server infrastructure.
Project Lead	Usually non-technical. Responsible for project outcome.
Software Developer	Develops the software (!).
Functional Tester	Creates functional test specifications and executes them (manually).
Test Automation Developer	Creates automated repeatable tests.
Quality Manager	Ensures that the delivered software and other by-products adhere to the

5.2 Scenario's

This paragraph describes a set of common scenario's in the life cycle of a software development project. The scenario's are written with a particular stakeholder in mind and help us to understand the needs, wishes and problems in a structured way.

5.2.1 Scenario 1: On-boarding a new project

Stakeholder: All.

When a new project is taken on by the development organization a technical infrastructure needs to be setup in order to accommodate the development process. It includes the setup of a CI/CD-pipeline, access-management and creation of several (virtual) deployment servers.

5.2.2 Scenario 2: Accommodate different sized projects

The tooling in the CD-pipeline needs to be flexible enough to support projects of different sizes and architectures. One project can be just as simple as a website with a couple of form inputs. It can be deployed using a single web-server and requires a single database. On the other hand there exist projects that develop applications to administer larger parts of the governmental resident databases. These applications are usually deployed on clustered load-balanced environments and require a redundant database setup. One step further are the applications which are deployed as a set of independent services, requiring infrastructure integration like a message-bus, central authorization handling.

In order for the tests to be as realistic as possible, each setup requires a production like environment.

5.2.3 Scenario: Migrate to a different version of Java

Stakeholder: Software Developer.

When a new (bugfix)-version of Java is released developers need to update their development, continuous integration and deployment environments. Java needs to be updated on the developer's machine, CI-server and the different test environments.

5.2.4 Scenario: Upgrade the application server

…

5.2.5 Scenario: Application instance per tester

Stakeholder: Functional Tester, Test Automation Engineer.

When a Tester needs to ascertain functionality or check for (absence of) regressions it is very useful if an instance can be started with ease by the Tester. This ensures

that observed behavior is not impacted by actions of other's which is crucial to come to a proper judgment or test script definition. The Tester should be able to start an application instance with a specific test data set at will and by the push of a button.

5.2.6 Scenario: Parallel frontend test execution

Stakeholder: Test Automation Engineer.

5.2.7 Scenario: Install a plugin in Jenkins CI

Stakeholder: Software Developer.

5.2.8 Scenario: Upgrade Jenkins CI

…

5.2.9 Scenario: Switch to another tool

…

Chapter 6

Continuous Delivery at the Development Organization

In this chapter we will discuss the previous and current Continuous Delivery environment at the project organization. I use the scenario's described in the previous chapter to exemplify the possibilities and problems of both environments.

6.1 CI in a shared environment

This paragraph describes the previous CI/CD landscape at the development organization.

6.1.1 The situation

The systems needed for CI/CD are managed by an Ops team. All projects use a set of shared services. Figure 6.1 depicts the relationship between the Ops team and the development teams. The shared services are:

- Subversion
- Jenkins CI
- Jenkins build servers
- SonarQube
- Nexus
- Jira

Besides the shared services each project would be assigned one or more deployment servers. The deployment servers are managed by the Ops team.

The next chapter describes common scenario's that occur in a CI/CD environment on request of the development team. These scenario's describe the impact on the development team.

6.1.2 Scenario's detailed

TODO In this paragraph I will detail the aforementioned scenario's for this type of environment. Which scenario's can be implemented in this environment? Are more troublesome? Cannot be implemented? Require a lot of manual intervention/work/configuration?

6.2 CI in a distributed environment

This paragraph describes the current CI/CD landscape at the development organization.

6.2.1 The situation

Instead of managing the systems needed for CI/CD the Ops team manages a distributed environment in which teams are able to deploy applications at will and on demand.

6.2.2 Scenario's detailed

TODO In this paragraph I will detail the aforementioned scenario's for this type of environment. Which scenario's can be implemented in this environment? Are more troublesome? Cannot be implemented? Require a lot of manual intervention/work/configuration?

6.3 A new project

This paragraph conceptually describes what happens when a new project is embedded within the development organization. Besides organizational arrangements a technical infrastructure is setup to accommodate the development of the software application.

The following systems are employed:

- Gitlab
- Jenkins CI

- Jenkins build servers
- SonarQube
- Nexus
- Mediawiki
- Deployment servers
- Jira
- Releasemanager
- Quality dashboard
- Test reporting

The next paragraphs talk about the tasks that happen initially and tasks that recur more frequently.

6.3.1 Infrastructure setup

Initially every system used needs to be installed onto a target server. Depending on how you choose to do the installation, this might take some time.

6.3.1.1 Gitlab

Gitlab is used as a revision control server.

1. Install Gitlab
2. Configure authentication mechanism (LDAP)
3. Set roles and permissions for users

6.3.1.2 Jenkins and build servers

Depending on the project one or more build servers are needed. A build server has specific tooling on-board to be able to build the application. The following list details the installation steps.

1. Install Jenkins CI
2. Install one or more Jenkins build servers
3. Install specific tooling on the build server
4. Configure the build server in the main Jenkins server
5. Configure authentication mechanism (LDAP)

6.3.1.3 SonarQube

SonarQube is used to continuously monitor the quality of the source code.

1. Install SonarQube
2. Configure authentication mechanism (LDAP)

6.3.1.4 Nexus

Nexus is used to archive and distribute software artifacts.

1. Install Nexus
2. Configure authentication mechanism (LDAP)

6.3.1.5 Mediawiki

Mediawiki is used as a team collaboration tool.

1. Install Mediawiki
2. Configure authentication mechanism (LDAP)

6.3.1.6 Deployment servers

For the purpose of deploying the application in a production like environment a deployment landscape has to be setup. Depending on the application this can be as simple as a single server, or as complex as a clustered setup of a Java application server with a corresponding complex database setup.

6.3.1.7 Jira

Jira is readily available within the organization and doesn't need to be setup. However, it needs to be configured to accommodate the new project.

6.3.2 Project setup

After the infrastructure is setup the project team adds configuration to the tools to be able to build and deploy their application.

1. A user for Jenkins needs to be created in Gitlab and configured in Jenkins so that Jenkins can checkout copies of the source code.
2. Code repositories are created in Gitlab for the corresponding applications.
3. Optional, import existing source code into Gitlab.
4. Configure jobs in Jenkins to build, test and deploy (for every application)
5. Configure the location of the SonarQube API in Jenkins

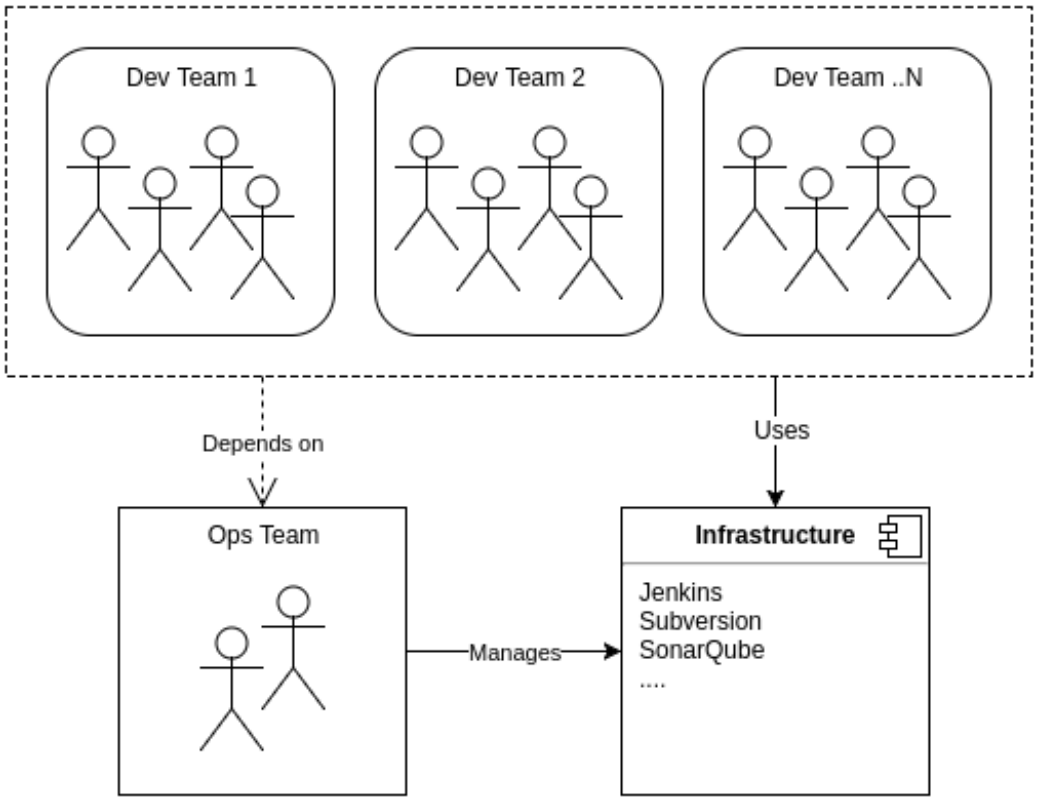

Figure 6.1: Relationship between Ops and Development teams in a shared environment

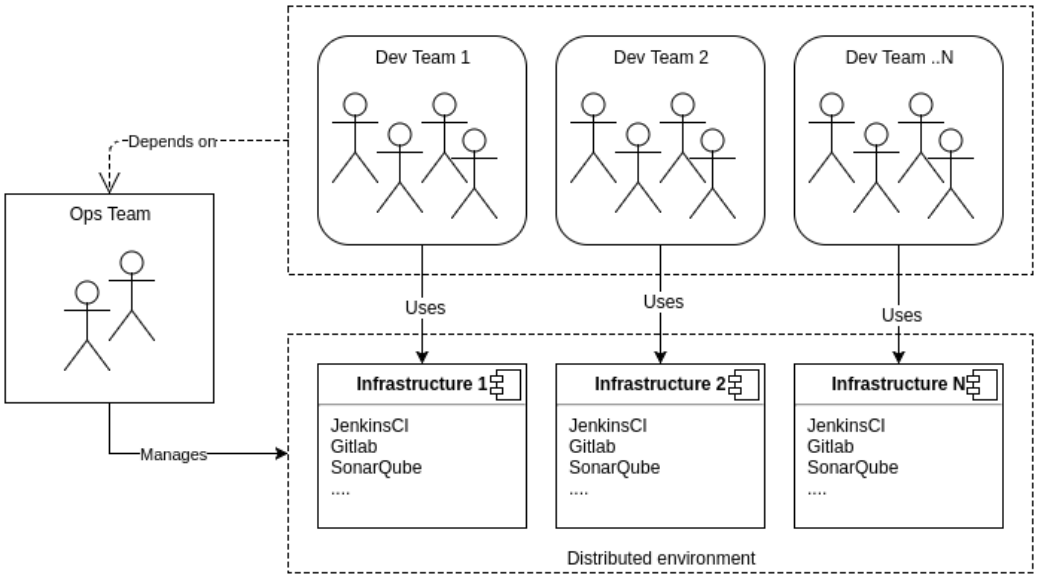

Figure 6.2: Relationship between Ops and Development teams in a distributed environment

Chapter 7

Stages of Continuous Delivery

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

Table 7.1: Demonstration of simple table syntax.

Right Le	ft	Center De	fault
12 12		12	12
123 12	3	123	123
1 1		1	1

7.1 Stage 1: CI in a shared environment

Characteristics

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

Systems

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

Server	Type	Depends on
Selenium		Deployment Server
Deployment Server		Nexus

Tools

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

Setup

- Setup is done by system administrators

Manual steps

Task	Depends on	Occurrence
Create build job	Jenkins	every new unit of development
Create deployment job	Jenkins	every new unit of development
Configure quality report	Jenkins, Quality reporting	every new unit of development
Maintain server configuration	Deployment Server	on configuration change
Trigger deployment	Deployment Server, Jenkins	on request of tester or stakeholder
Trigger automated tests	Deployment Server, Jenkins, Selenium	every iteration

Problems

Description	Has negative impact on
Resource sharing between all teams	Scalability
Changes and upgrades affect all teams	Stability
Teams can't change setup or install plugins	Flexibility, Usability
Teams can interfere with each other	Stability
Teams depend on sysadmins	Agility

Description	Has negative impact on
Deployment server changes are difficult to reverse	Flexibility, Scalability
Unable to deploy multiple instances of an application	Agility, Usability

7.2 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

7.3 Stage 3: — next evolution..

tbd..

7.4 Initial situation

! This needs to be placed elsewhere and rewritten !

Figure 7.2 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 7.3 shows the steps a developer needs to take in order to setup a single source repository and configure the continuous integration pipeline.

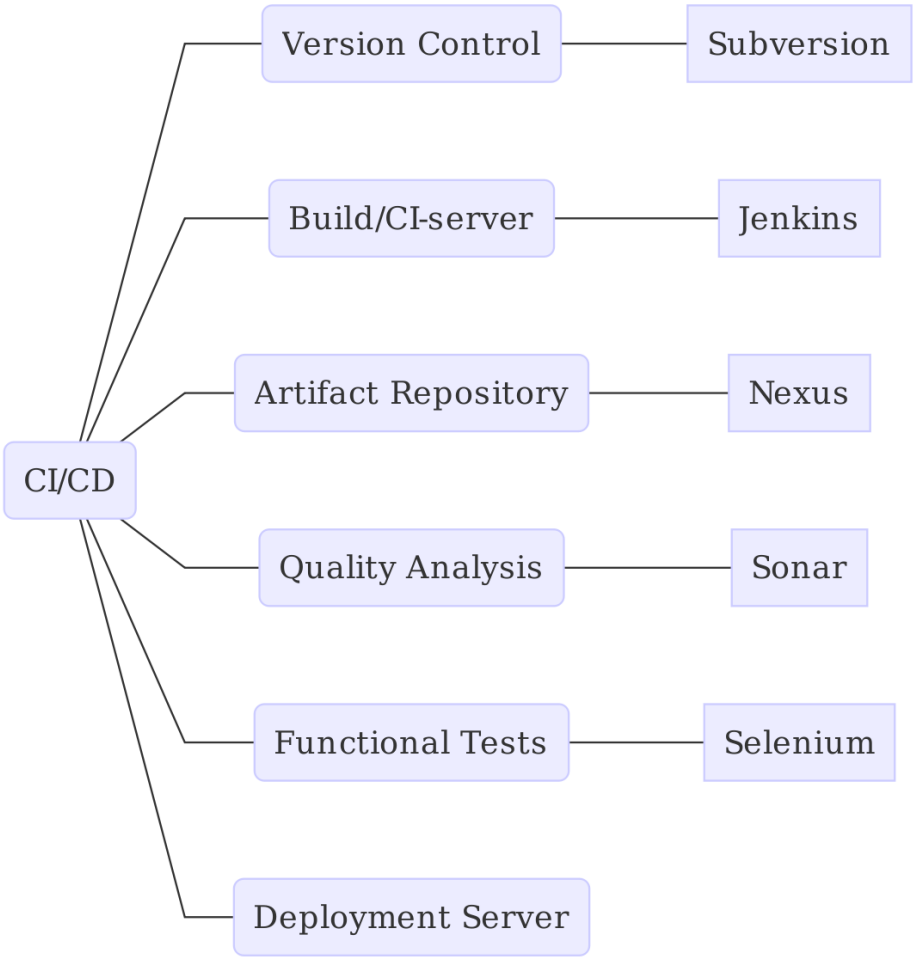

Figure 7.1: CI/CD Schematic Overview

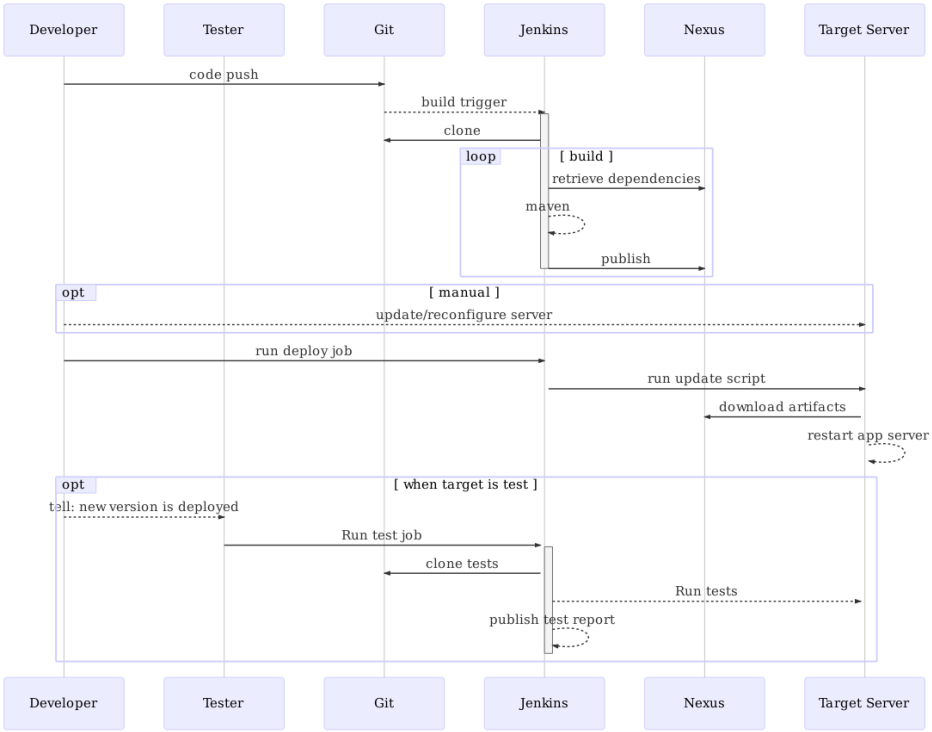

Figure 7.2: Basic CI

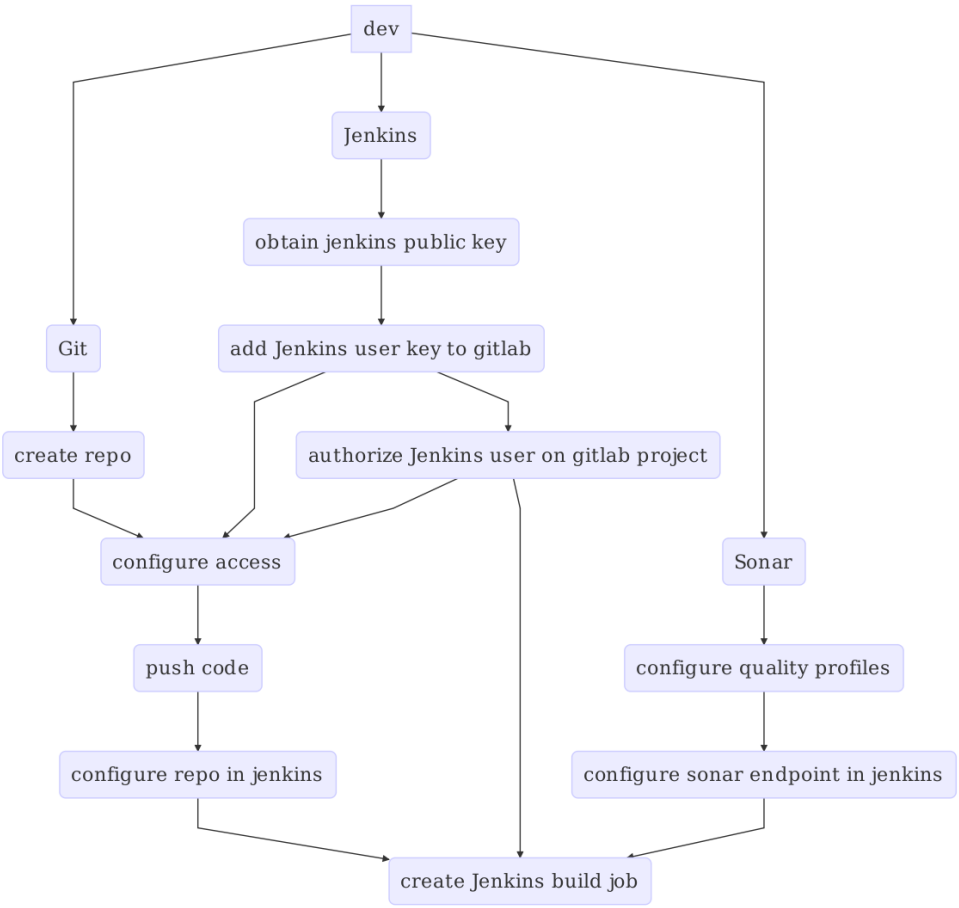

Figure 7.3: Basic CI setup

Chapter 8

References

Appendix 1: Team interviews

Team 1

Property	Value
Date	22-11-2016 10:00
Duration	20 minutes
Present	***I:*** interviewer
	R: developer 1
Team members	Two developers
Team size	Small
Project size	Large

Transcript

This paragraph contains the annotated transcript of the interview. One developer took part. Since the interview was in Dutch the transcript is also in Dutch. Annotations are in English.

I: waar wij als IQ-team geintereseerd in zijn is hoe de oplossing in de ontwikkelstraat wordt ervaren. Eventueel de problemen die daar mee zijn maar ook of jullie daar voldoende ondersteuning bij hebben. Of je door de organisatie gesteund voelt. Eigenlijk alles wat daarmee te maken heeft. en wat je er van vindt. Eventuele problemen. Jullie gedachtes.

R: Het meeste wat wij graag willen is dat het (ontwikkelomgevingen) gewoon beschikbaar zijn. En dat is het hier wel. En het is makkelijk. Alles kun je gewoon via Docker starten, dus dat maakt het wel heel erg simpel. Zo van, we hebben Git nodig. Toen we begin dit jaar met Bulk begonnen was het van hop, en daar stond Git. Dan moet je het nog wel een beetje inrichten, je moet nog certificaten maken en dat soort dingen. Dat was nog soms wel een beetje vogelen van hoe krijgen we dat nou vanuit jenkins dat certificaat er in en werkend, dat wilde in het begin nog niet want toen zat er nog een ander certificaat dwars. Dus daar zit soms wel wat uitzoekwerk. Toen ben ik bij jou zelfs langsgekomen en dan wordt er gewoon even naar gekeken, even inloggen op die server, dan kom je er meestal wel weer uit. Dus ja, eigenlijk ben ik wel tevreden daarmee. Er zijn altijd wel dingetjes natuurlijk.

```
- Continuous Delivery environment should have high availability
- Help is needed to finalize CD services configuration
```

I: Wat is nu jullie manier om een nieuwe release te testen, wat doen jullie? Wat is jullie manier?

R: In het begin hebben we in Jenkins een job gemaakt die dan een Docker image maakte die we vervolgens weer konden gaan starten. En dat konden we via Jenkins weer starten. Dat

bleek te onhandig omdat je vaak lokaal eerst wilt testen om te kijken of het goed is voordat je daadwerkelijk een build doet. Uiteindelijk zijn we terug gegaan naar scripts om Docker images te bouwen en om te kijken of het allemaal goed is en dan pushen we uiteindelijk.

I: Dus je runt eerst lokaal je applicatie?

R: Ja, omdat dat toch makkelijker is. Je kunt makkelijker testen en nog eventjes een nieuwe deployment erin doen. Dat gaat lokaal makkelijker. Daarna pushen we en dan starten we hem nog een keertje opnieuw op de Docker host zodat anderen er ook naar kunnen kijken. Bijvoorbeeld om te reviewen en de hele reutemeteut.

```
- Local builds are easier and quicker than building on CI environment
```

I: Je pusht hem gewoon vanaf je lokale machine dan? Dus je bouwt hem niet via Jenkins?

R: Nee, dat hebben we dus wel gedaan. We hebben nu ook zoiets van; dat moeten we nu ook wel weer gaan doen. R heeft wel meer moeite om het lokaal te draaien omdat z'n laptop dat gewoon niet zo goed trekt. Misschien moeten we toch wel weer gaan kijken om van die jobjes te maken. Maarja, de drempel om die jobjes te maken is dan wel weer hoog. Het zijn er namelijk best veel.

```
- Local builds require a performant developer machine
- Many CI jobs needed to build the software
```

I: Dat is best goed om te weten dat je daar tegenaan loopt.

R: Het is natuurlijk ook gewoon veel. Voor elke aansluitvoorziening hebben we ook een container. We zijn er eigenlijk een beetje laat mee om dat te gaan doen. Als je dat nu zou willen doen moeten we eigenlijk 40 van die dingen gaan maken. En jobjes. Dus tja.

I: Dit is wel specifiek voor jullie project. Je hebt echt ontzettend veel deelapplicaties.

R: Ja het zijn er veel, waardoor het ook niet meer leuk is om dat achteraf te doen. Opzich zou het wel handig zijn om dat wel te doen. Omdat als er iemand nieuw bijkomt die denkt van 'Hoe krijg ik hier nou een container gebouwd?' Dat is best wel even zoeken. Als je gewoon een jobje hebt waarbij staat 'bouw deze', dan is het van 'klik'. Dus in die zin is het wel nuttig om het wel te doen.

```
- Building on CI server would enable new team members to start quicker
```

I: Je hebt nu wel scripts maar je moet wel weten waar ze staan?

R: Ja, en je moet weten welke je moet runnen.

I: En je moet dus een pc hebben die krachtig genoeg is om het lokaal te bouwen.

R: Ja inderdaad.

I: Maar verder weinig problemen eigenlijk?

R: Ja, eigenlijk niet zo heel veel problemen. Met Docker wel af en toe dat een host vol was en dat 'ie dan onderuit gaat. Maar dat gebeurd niet zo vaak meer. We hebben de load iets beter verdeelt over de hosts die we hebben. Dat gaat nu redelijk. We hebben nu drie hosts. Dus dan past het meestal wel. Misschien als het nog verder gaat met dit project dan hebben we misschien nog meer hosts nodig.

- Docker host crashes occasionally
- Manual load distribution between Docker hosts

I: Was jij al bij het project betrokken toen het project nog niets met Docker deed?

R: Ja op het moment dat ik binnenkwam was er net een eerste twee Docker containers. En de rest stond toen nog op de virtual machines.

I: Dat is nu nog steeds zo toch?

R: Nee we hebben onderhand alles al wel omgezet. Behalve A-select, SIAM, dat is nog een fysieke server. Niet echt fysiek natuurlijk. De rest is allemaal in Docker. Dus opzich, moet ik zeggen, bevalt het allemaal wel goed. Je hoeft ook niet meer ergens in een wiki bij te houden van 'waar stond die server ook alweer', 'welke was het ook alweer'. Nu ga je gewoon naar het dashboard en start je de juiste applicatie. Copy-paste van ssh en je kunt er even in.

I: Dus wat dat betreft is dit ech een verbetering?

R: Ja.

I: Ik heb begrepen dat eerder het aantal virtuele machines beperkt was dat er als er bepaalde klanten kwamen om te testen dat het dan een heel gedoe was om de juiste data in te laden. Hoe is dat nu?

R: Nouja, we hebben nu een cache database ingericht. We hebben daarvan een backup gemaakt. Als je nu een applicatie start kan je die data weer inladen. Je weet gelijk als je hem start en Bulk gaat repliceren dan weet je precies wat er in zit. We laden een standaard dataset, dus je hebt altijd een goede set. In die zin, als het vernaggeld is of als er getest is dan herstart je het gewoon weer en dan staat de standaard testset gewoon weer klaar. In die zin hoeven we niet zovaak meer naar de dataset te kijken of anders configureren en testsets laden. Dat scheelt wel.

- Docker enables easier management of data- and test-sets.

I: Hoe kijk je aan tegen het gedeelte van de kwaliteitsrapportage?

R: Opzich is het handig. Kwaliteit en testrapportages komen er netjes uitrollen.

I: Leveren jullie die rapportages op aan de klant?

R: Kwaliteitsrapportage niet, testrapportage wel. De testrapportage willen ze ook graag hebben om te kunnen zien of er getest is en wat er getest is. In het mastertestplan staat ook de verantwoording, dit doet de ontwikkelorganisatie en dit doet de klant. De klant wil graag zien wat er gebeurd is en van 'kijk er is echt daadwerkelijk gestest'. Het is in elk geval een verhaal van wat er gebeurd is. Maar daar zijn wel wat vervelende dingen. Omdat we heel veel projecten hebben, als we dan een story hebben in Applicatie1. Daar hangen logical test cases onder. Maar die story raakt ook Applicatie2. Of over de hele linie moet er een aanpassen worden gedaan. Dan komt er uit de testrapportage dat een story niet gevonden kon worden. Want dan testen we Applicatie2. Een Applicatie1 test case kan dan niet worden gevonden, want die staat in een ander project. Dan moeten we een nep-story maken waar dan dezelfde logical test case onder hangt om het te laten kloppen. Dat is een beetje irritant.

- Managing logical test cases shared between projects increases administration load.

I: Overbodige administratie dus?

R: Ja, eigenlijk overbodige administratie. Maargoed. We zijn bezig om Jira wat te consolideren. We zijn uberhaupt bezig om alle applicaties aan te pakken en meer samen te voegen. We zijn ook bezig met 1 Jira project om daar alles onder te hangen. Dus dan zullen we hier niet zoveel last meer van hebben. De testrapportage opzich werkt uitstekend.

I: Waarom leveren jullie de kwaliteitsrapportage niet op?

R: Ja, daar staan ook heel veel dingen in van 'is niet goed', of 'doet het nog niet'. Dat komt omdat we uberhaupt veel componenten hebben waar een jaar of twee jaar niet aangewerkt is en waar ik zelf nog nooit aan gewerkt heb. Daar staan dan issues op omdat we dan een nieuw kwaliteitsprofiel hebben. Dus tja, das is leuk maar hallo. Daar gaan we nu echt niet naar kijken, maar het staat allemaal wel in die rapportage. Dus als je dat dan oplevert wat zegt dat dan? Er staan veel rode dingen. De klant is er ook niet echt in geintereseerd. Hij wil weten dat er getest is. En ja, wat de kwaliteit voor de rest is. Hij gaat er vanuit dat het niet al te best is. Dus tja, en dat is misschien ook wel zo. Maar het wordt beter. We gaan nu een begin maken om er doorheen te lopen en dingen vernieuw. Dan gaan we wel weer voldoen aan de kwaliteitseisen.

```
- Quality report contains many violations
- Customer is not interested in quality report
```

I: Jullie kijken zelf wel naar de Sonar rapportages?

R: Dat is natuurlijk een onderdeel van de kwaliteitsrapportage. Dat deel is over het algemeen slecht. We doen er wel wat mee. Als er echt majors bijkomen. Op een bepaald moment hadden we een nieuw profiel. We zagen door de bomen het bos niet meer. Welke issues zijn er nou bijgekomen en welke waren er al? We kunnen dat allemaal niet fixen. Toen is er op een bepaald moment besloten om toch maar weer een ouder profiel te laden. Zo hadden we weer zicht op wat er bijgemaakt was aan fouten. Dat lossen we dan op, zorgen dat er geen majors bijkomen. Dat is het dan wel. Dus ja, dat werkt opzich wel. Maar voor ons om het bij te houden is het een beetje te veel.

```
- Team tries to prevent new major quality violations
```

I: Er zijn nu veel mogelijkheden om zelf applicaties en ondersteunende services te starten. Maak je daar gebruik van? Heb je zoiets van, ik wil een bepaalde tool gebruiken of op een bepaalde manier inzetten? Er is nu mogelijkheid om naar eigen inzicht tooling te starten. Het is mogelijk om af te wijken van de standaardtooling als Jenkins, Gitlab e.d.

R: Nee, ik vind het een goede toolkeuze. Jenkins is gewoon een goede tool. Gitlab is ook gewoon een goede tool. Dus ik heb in die zin niet de behoefte om iets anders te starten. De standaard toolset is gewoon een goede keuze. Als je iets nodig zou hebben daar rondom, dan kun je dat natuurlijk starten. Ik heb nog niet zoiets gehad van ik heb echt dit nodig.

```
- Default tool selection is sufficient
- Team didn't make use of possibility to shape their own CI/CD environment
```

I: Je kunt nu natuurlijk ook Gitlab gebruiken als CI-server, dus je zou alles kunnen overzetten vanuit Jenkins.

R: Dat is niet iets wat we leuk vinden om te gaan doen. en sowieso, als je alles over moet gaan zetten is dat geen pretje. Het is heel veel werk. Dus als het eenmaal draait in Jenkins dan is het goed zo. Maarja, er zijn wel dingen die we meer willen automatiseren met Docker containers. Stel je hebt een release gedaan van een component dan willen we ook automatisch de Docker

container updaten. Dat is heel leuk, dat willen we. Waarschijnlijk zullen we dat alleen doen voor de nieuwe componenten die we gaan maken. Dat houdt het ook overzichtelijk. Het is wel iets dat we graag willen. Je hebt dan niet zoveel stappen aan het eind van een story om alles helemaal rond te krijgen. We moeten dan componenten releasen. In de meeste gevallen hebben we zo'n drie componenten per story. En dan moet het nog in Docker gezet worden, de containers moeten gemaakt worden, pushen. Het zou wel makkelijk zijn dat als je een release maakt dat de containers dan automatisch geupdated worden. Maar dat zit er nog niet in. Het komt allemaal een beetje op hetzelfde neer. Het is veel te veel! Het kost veel tijd, waardoor we het dan toch laten zitten.

```
- Increasing build automation is seen as an expensive and time consuming investment.
```

I: Hoe zou je de basiskennis van al die tooling beschrijven? Zijn jullie experts? Hadden jullie al kennis van de tooling toen je aan het project begon, of heb je die tijdens het project opgedaan?

R: Nee geen experts. Voor dit project had ik ook al Jenkins gebruikt. Daarvan weet ik wel wat het ongeveer kan en welke plugins je nodig hebt. In die zin, als gebruiker hebben we de kennis wel. Echt geavanceerde dingen dan moeten we ook echt even gaan zoeken. In het vorige project maakte ik ook gebruik van Docker, dus ik kende Docker wel. Maar het is hier wel lekker opgezet met het dashboard enzo. Dat maakt het allemaal wel makkelijker om het te gebruiken. Je hoeft niet eens zo heel veel van Docker te weten om het te gebruiken. Dus ja, basis dingen weten we wel en iets geavanceerder ook wel. Maar we moeten ook wel even zoeken.

```
- Little Docker knowledge is needed to get started in current environment
- CI/CD setup with default tooling makes it easy to start using it
```

I: Mis je dat je niet direct op de host kunt kijken? Waar die containers draaien?

R: Nee, op de Docker host hoef ik eigenlijk zelf niet echt te kijken. Het enige is dat als er wat omvalt dat je dan nog wel eens zou willen kijken.

I: Is dat ook de reden dat jullie op de ontwikkelomgevingen de containers eerst bouwen en daar testen? Want als je het dus deployed via het dashboard en het werkt niet, dan weet je eigenlijk niet zo goed hoe of wat.

R: Ja, we bouwen natuurlijk lokaal. Als er iets mis gaat dan zoeken we dat lokaal uit. We hoeven dan dus niet op de host. Als je het gepushed hebt en je start via het dasboard een docker container, als er inderdaad iets mis zou gaan, maar er gaat eigenlijk nooit wat mis. We hebben wel eens gehad dat je pushed en dat de image toch niet aankwam. Als je hem dan start dan zie je bijvoorbeeld dat de wijzigingen er niet in zitten. Er was dan vaak een probleem met resources op de host waardoor de nieuwe image niet goed doorkwam. Maar voor de rest heb ik nooit de behoefte gehad om zelf op die host te kijken.

```
- Local builds are done to prevent problems on CI/CD environment.
- Problems on CI/CD environment are difficult or impossible (due to access restrictions) to debug.
```

I: En de Docker registry? Waar veel teams problemen mee hebben is dat de disk vol loopt. Er moeten dan oudere images verwijderd worden. Hebben jullie daar last van?

R: Nee. Dat is eigenlijk wel gek.

I: Hoevaak en hoeveel images pushes jullie?

R: In een sprint, stel dat we iets van vier a vijf stories hebben met wijzigingen. Dan hebben we verschillende dingetjes die we zullen pushen. En dat doen we ook wel een paar keer denk ik. Dus ja. Het zijn er niet meerdere per dag. Je draait natuurlijk alles lokaal. Daar testen we eerst en daarna pushen we pas. Per sprint pushen we misschien zes a zeven keer een image. Zoiets zal het zijn. Ik weet niet hoeveel je moet pushen om je registry vol te krijgen?

`- Team releases new image six/seven times per sprint.`

I: Nouja. als je Continuous delivery doet met meerdere pushes per dag dan is het snel vol.

R: Ja precies, elke keer als je een wijziging hebt dan wordt het gelijk daarheen gepushed. Ja, wat wij ook nog wel eens doen is nog niet pushen als we nog bezig zijn met een story. Soms willen we dan wel dat Jenkins er al tegenaan gaat testen. En dan deployen we gewoon in de Docker container een nieuwe war of ear.

I: Jullie gebruiken wel een regressietestset? Naast de unittests? Waarin is die gemaakt?

R: Ja, die is gemaakt in Java Selenium. Die draait inderdaad in Jenkins. Dus die test in principe op de testomgeving op Docker. Soms draaien we de testen lokaal. Dat werkt ook en gaat vaak sneller. En ook wel eens op Jenkins. Sommige testen duren drie kwartier, dus daar wil je lokaal niet op wachten. Maar we doen dan geen push, omdat we nog bezig zijn. We doen dan alleen een deployment van een nieuwe war of ear. Dan test Jenkins daar tegenaan.

`- Running integration tests locally is quicker than on CI/CD environment.`

I: Op de container?

R: Ja op de container.

I: Aah, op die manier. Sneaky.

R: Ja sneaky heh. Hahaha.

I: Jullie gebruiken het in dat geval eigenlijk als een soort virtuele machine.

R: Ja. Gewoon even de war of ear er op. De tests draaien en als dat allemaal goed is en de story is uiteindelijk klaar bouwen we wel netjes een image.

I: Maar is dat uiteindelijk minder moeite dan elke keer een image bouwen?

R: Ja dat duurt veel langer. Dit gaat sneller.

`- Team uses Docker container as VM to gain time advantage.`

I: Hoe vind je de ondersteuning vanuit de organisatie? Door de ondersteundende teams?

R: Goed, als je langsloopt word je geholpen.

I: Je zei eerder dat je al eerder ervaring met Docker had opgedaan. Er was voor jou niet een enorme leercurve op dit te gaan gebruiken?

R: Nee. Sommige scripts waren wel even wat anders dan ik gewend was.

I: Wat bedoel je met de scripts?

R: De dashboard applicatie definitie. Maar het spreekt redelijk voorzich, dus als je een beetje weet hoe het in elkaar zit en werkt dan is het heel simpel op te pakken. Niet zo ingewikkeld. In die zin had ik niet zoiets van 'help, wat moet ik nu'. Maar ook dan, als we vragen hadden dan konden we gewoon langslopen en kregen we vaak direct het antwoord. Dat is uitstekend. Positieve ervaringen.

```
- Support teams are easy accessible and help accordingly
```

I: Dat was het voor nu, als we nog meer vragen hebben dan komen we gewoon later terug.

R: Dat mag.

Team 2

Property	Value
Date	02-12-2016 14:00
Duration	57 minutes
Present	***I:*** interviewer ***B***: developer 1 ***J***: developer 2 ***R***: developer 3
Team members	20
Team size	Large
Project size	Large

Transcript

This paragraph contains the annotated transcript of the interview. Three developers took part. Since the interview was in Dutch the transcript is also in Dutch. Annotations are in English.

I: Waarom ik jullie hebt uitgenodigd is omdat wij als IQteam meer inzicht te krijgen in hoe de teams de huidige continuous delivery omgeving gebruiken, docker en alles wat daarbij komt kijken. We gaan alle teams af om te inventariseren wat de status is bij de teams en hoe ze de omgevingen gebruiken. Wat zijn de problemen, waar loop je tegen aan. Wat gaat er goed? Zijn er ideeen voor verbeteringen. Van alles en nog wat. Ik heb een vragenlijst waar we gedurende het gesprek doorheen gaan.

I: De eerste vraag is dan direct; welke problemen komen jullie tegen bij het gebruik van de huidige ontwikkelstraat, docker platform, in je dagelijkse werkzaamheden. Wat zijn belemmeringen waar je tegen aanloopt.

B: Resources. Physical resources, dus geen human resources. Diskruimte, ip-adressen. Maar ook beschikbaarheid. Performance, cpu. Dat soort dingen. Het is niet inzichtelijk. Je ziet niet wat de beperkingen van de resources zijn.

J: Details worden afgeschermd. Je hebt een host maar daar hoef je niet over na te denken. Maar op het moment dat er iets gebeurd weet je ook niet wat er gebeurd. Je weet niet waarom het niet opschiet of waarom het faalt.

I: En vind je dan het probleem dat je er niet zelf naar kunt kijken?

J: je hebt minder goed begrip van wat er eigenlijk gebeurd.

B: Ja.

I: Zouden jullie meer controle willen hebben? Zodat je zelf kunt onderzoeken waarom dingen mis gaan? En eventueel zelf kan fixen? Of zie je dat als iets wat het IQteam zou moeten doen?

J: Ik denk dat het handig is als we het zelf kunnen fixen. Nou… het probleem kunnen vinden in iedergeval.

B: Ja.

J: We willen beter inzichtelijk hebben wat de status van de host is.

B: Kijk jullie hebben bijvoorbeeld op je scherm al die notificaties. Zoveel procent diskusage, zoveel procent cpu, zoveel procent geheugen. Zo'n monitor zouden wij ook willen hebben. Ookal kunnen we er op dat momenet nog niets mee doen, je ziet gewoon aankomen dat er zometeen iets fout zal gaan.

I: Hoevaak komt zoiets dan voor?

B: Nou, laatste weken behoorlijk vaak.

I: Meerder keren per dag? Of 1x per dag?

B: Nou, als het 1x per dag faalt dan is het over. Dat is ook het probleem heh. Het is zo'n strategisch product geworden dat als het omvalt dan zit gewoon iedereen stil. Dat is dus in ons geval 20 man voor X-aantal uren.

I: Je bedoelt wanneer de Docker host omvalt?

B: Ja, dan kan bijna niemand iets meer doen. Er is geen Jenkins, geen conflunce, geen gitlab. niets. Dus je kunt helemaal niets doen. Twintig man zit dan stil. Het is vergelijkbaar met als we geen netwerk hebben. Of als we geen internet hebben.

J: Plus de aanloopuren. Het klapt er meestal niet in 1x uit, maar performance wordt steeds slechter. Dus dan ga je eerst uitzoeken waarom het niet werkt. Bijv. de ART faalt, dan ga je een timeout veranderen.

B: Ja het accumuleert. Normaal gesproken als je iets doet dan doe je dat rustig en je doet je werk. En dan is het twee uur later dan kijk je nog eens een keer. Als er dan iets begint fout te gaan dan ga je repareren. Dan ga je dit testen, en dat testen. En iedereen doet dat. Dus we zitten nu ineens met z'n zessen dingen te repareren die eigenlijk te maken hebben met timeouts, geen resources, alles is een stuk trager. Als de harddisk 80-90% vol zit dan krijg je een grote performance penalty.

I: Deze problemen die jullie hebben, hebben die vaak dezelfe oorzaak? Of is het telkens iets anders waardoor resources op zijn?

B: Er zijn twee oorzaken; diskruimte en ipadressen.

I: Maar waardoor loopt jullie diskruimte vol?

B: Images. Ik bedoel, we hebben in totaal 300Gig. Onze baseline zit op 200G. Dus blijkbaar is dat wat wij minimaal gebruiken. Bij 260G klapt het ding. We hadden dus heel weinig ruimte. Images die we continue maken, per stuk is dat 2Gig. Per set zijn dat 5-6 images. De meeste zijn kleiner, maar de grootste zijn rond de 2Gig. Dan praat je dus over per keer 10G.

R: Nu moet ik ook wel zeggen dat we net opruimen. Dat is wel iets wat we zouden moeten doen. We pompen maar bij en ruimen niets op. Net zoals met een kamer die je niet opruimt. Je kunt

er wel bij blijven gooien maar op een gegeven moment houd het op. Dus daar zit ook wel een schone taak voor onszelf.

I: Misschien als dit probleem bij meerdere teams zit, dan is het misschien iets dat we centraal kunnen regelen. Of eenvoudiger zou kunnen maken

R: waar ik zelf aan heb zitten denken, en wat ik zelf fijn zou vinden, is om in plaats van een centrale oplossing een decentrale oplossing te maken. Dat schaalt natuurlijk altijd beter.

I: In welke vorm zou je dat dan willen zien?

R: Nou mijn natte droom is om Jenkins lokaal te kunnen draaien.

I: Maar waarom kan dat nu niet dan? Wat weerhoud je daar nu van?

R: Eeehhm… nou, de tijd ontbreekt het aan. Maar ook configuratie. Configuratie is nu nog centraal, dus die zou je dan beschikbaar moeten maken. Maar dat zou volgens mij wel kunnen, is volgens mij niet zo heel moeilijk. En ja, je zou Jenkins in een Docker container kunnen draaien natuurlijk. Maar je zit met de configuratie, die moet je centraal distribuereren.

B: Maar Jenkins draait toch al in een Docker container?

R: Ja, nee daarom. Dus het is volgens mij helemaal niet zo moeilijk. Alleen moeten we kijken wat we dan met de configuratie moeten. Die zou je dan eigenlijk ook willen versionen.

I: Ik probeer alleen nog te begrijpen wat nu precies het probleem is dat je hiermee probeert op te lossen? Er is dus een probleem met de availability van de bouwstraat. De vraag is of dit de juiste oplossing is.

R: Ja ik denk het wel. kijk als er centraal op een gegeven moment iets klant en niemand kan meer Jenkins draaien, dan zit je.s

I: Jenkins hangt natuurlijk aan al die andere systemen. Je gaat nog steeds naar de centrale Docker repository pushen. Als die vol zit, dan kune je ook niet zoveel meer met je lokale Jenkins.

B: Ja, je zou wel verder kunnen. Lokaal heb je je eigen registry. En je hebt natuurlijk alle caches lokaal; Docker, Maven, NPM.

I: Een beetje achtegrondinformatie, alle services die nu draaien in de ontwikkelstraat draaien op 1 resource pool. Dus als er iets met die pool aan de hand is dan vallen al vrij snel meerdere diensten om en heeft iedereen daar last van. Misschien zou je meerdere resource-pools willen hanteren waardoor je de essentiele services kunt scheiden van de volatile deployments?

B: Maar die hebben we al gesplitst. We hebben twee pools.

I: Uuhm. Ja in jullie situatie is dat inderdaad zo.

B: Die stap is al gezet.

I: Ja oke. En waar het nu fout gaat is op de plek waar al je applicatie instanties draaien.

B: Blijkbaar beinvloeden ze nog steeds elkaar.

I: De IP-pool wel ja.

B: Maar storage ook. Die 300G is voor beide hosts. Als die vol is werken applicaties uit beide compute-pools niet meer.

I: Ja, jullie zijn nu gemigreerd naar shared data storage, dus dat klopt. Wat er nog niet geimplementeerd is zijn quotas die je kunt opgeven per applicatie. Dus op het moment dat je een service

start dan kan die alle resources gebruiken die beschikbaar zijn.

J: Dat is misschien iets waar naar gekeken moet worden. Het punt is een beetje; niet te groot team op een host. Of maximaal aantal gebruikers waardoor je power blijft houden.

I: Wat we dus missen is het limiteren van het aantal applicatie instanties die kunt starten en het limiteren van resources die elke instantie krijgt. Het voorkomt dat een enkele applicatie een host kan laten omvallen.

J: Ja, of als een host omvalt, automatisch herstarten zodat alle applicaties weer terugkomen.

I: Hoe gaan jullie nu om met het probleem dat de schrijfruimte volloopt?

B: Gewoon schreeuwen, vragen voor meer ruimte of hulp bij het opruimen en dan wachten…

I: Goed, over beschikbaarheid hebben we het dan al gehad. Even kijken…

B: We hebben wel een groot team, andere projecten zijn een stuk kleiner. Ons team is ook anders. Wij zijn volledig ge-Dockerized. Andere teams doen dat nog niet.

I: Jullie gebruiken dan ook geen andere virtuele machines?

B: Nee, nee. Al onze modules zijn allemaal in Docker. Daarom leveren wij dus per subsysteem een stuk of vijf of iets dergelijks. We hebben een paar van die subsystemen. Het aantal images, containers, is vrij groot als je het vergelijkt met andere projecten.

I: Dat is inderdaad wel interessant, daar gaan een aantal vragen over. Even over de grootte van het project. Om te kunnen vergelijken willen we weten hoe groot projecten zijn, hoeveel releases ze doen. Want je zegt ‘wij zijn een vrij groot project’, wat bedoel je dan?

B: We hebben twee teams, elke van 8 a 9 man. Plus overhead, zoals projectmanagement, performance tester en kwaliteitsmanager.

J: Stuk of tien developers totaal. Rest is tester.

I: aan hoeveel applicaties werken jullie? Of deelsystemen.

B: Stuk of zes. Plus alle tooltjes die er bij horen. De infra tool, graylog, activeMq, databases die erbij horen.

J: Wel wat meer denk ik zelfs.

I: Maar die bestaan allemaal uit 1 image, of meerdere?

B: Meerdere. Je hebt sowieso een applicatie en een database. En er zijn dingen die we delen, zoals de centrale logging. en ook de centrale queue. Alleen, voor de ART starten we dus een eigen queue op. Want die mogen niet in de weg lopen met andere ARTs.

14:20

B dus per ART heb je sowieso minimaal drie.

I: Wat bedoel je met minimaal drie

B Nou de applicatie, ART en de queue. Ja. Volegns mij is het volgens mij.

R we maken branches.

B Ja dat is ook iets anders. We werken met feature branches. dat wil zeggen dat elk z’n eigen branche waar die aan werkt en dat tikt ook aan.

I maar hebben we het dan ook over zelfde aantal image releases en pushes.

B Ja

I Ook op een dag, of doe je meerdere keren pushen naar de registry?

B Continu. Eeh zodra je dingen wilt gaan testen. Het is wel zo dat als die images niet gewijzigd zijn dan hebben ze hetzelfde ID, maar dan hebben ze een andere tag.

J Ja dat helpt nu best wel veel. Plus dat we kleinere images hebben gecreeerd.

B Dus daar hebben we het een en ander ook geoptimaliseerd.

I: Ja. Je noemde al de ART. Kun je beschrijven hoe het test process bij jullie d'r uitziet. Hoe start je dingen op, hoe is dat geautomatiseerd?

J: In principe bouwen we.. We draaien hem eerst lokaal, gewoon protractor met ART scripts tegen een instantie van het dashboard. Dus meestal starten we dan eerst gewoon zelf instanties van de applicatie. Dan bouwen ze hun ding, checken ze in en pushen ze. Dan leveren wij de art container mee. Daarin zitten alle ART's maar ook alle files op het te draaien. Dan doen we docker run en dat is dan de ART die tegen een instnatie draait. Zodat ze hem ook gewoon bij de klant kunnen draaien. Dan hoeven ze niet ons systeem te hebben. Dat ding wordt dan door jenkins gedraait via het dashboard. De pipeline start de instnatie op, op het moment dat ie er is draait de art. De results worden gepushed naar een of andere instantie op het dashboard. Iets met reporting, uuh ja. En dan stop het weer als het goed is.

I: en dat is dan per deelsysteem?

J: Ja, per deelsysteem.

I: en dat hele proces loopt ook meerdere keren per dag?

J: Ja, dat loopt best wel vaak.

I: Hoe lang duurt het hele proces?

J: Kleine tien minuten om te bouwen, vijf a zes minuten voor een ART.

I: En dat is ongeveer voor alle deelsystemen gelijk?

B: Het is nog wel erg klein heh. Ik bedoel, wij zijn nog steeds aan het begin van het project. Al onze testen en ART's zijn nog in de beginfase. Het is nog niet het volledige product.

R: Het is ook niet echt stabiel.

I: Wat is precies niet stabiel?

R: Vooral de ART. Soms dan draai je hem een keer en dan gaat het goed, en de tweede keer gaat het fout.

I: Gaat het dan fout omdat er functionele issues zijn? Dingen die mis zijn in de applicatie, of technische problemen als in de applicatie komt niet online?

R: Heel vaak heefft het met timing dingen te maken. De ene container moet voorde andere worden opgestart. Vooral bij databases en dat soort afhankelijkheden. Dan duurt de ene weer iets langer dan de ander. Dan start een applicatie nog een keer op en heb je ineens twee berichten in je queue staan. Dat soort dingen.

I: Op die manier.

R: Dat heeft ook heel veel te maken met het feit dat het niet op elk moment even druk is. De

resources die beschikbaar zijn fluctuen en daardoor verschillen de testen. Dan krijg je timeouts vooral bij asynchrone processen.

J: Het vervelende is dat ie eerst een instnatie op het dashboard start, maar als dat te lang duurt dan timed 'ie out. Maar je gaat niet een kwartier naar het scherm xzitten staren, dus dan ga je iets anders doen. Half uurtje later <>, hij is gefaald, waarom? Dan ga je zoeken. Oh timeout, ok. Build now, opnieuw. En dan weer een kwartier later. En als ie dan weer een keertje faalt moet je eerst wat dingen gaan afsluiten op het dashboard die dan blijven hangen. Dus ja, dan ben je zo al een uur verder voordat je eindelijk weet of 'ie het doet of niet. En dat bij elkaar telt best wel op aan tijd en context switches. Je kunt niet doorwerken, je moet constant controleren of alles goed gaat. Dat vind ik heel lastig.

I: Ja, maar dat heeft er niet toe geleid dat je alles eerst lokaal draait om zeker te weten of het werkt?

R: Nou dat is het punt, zegmaar de <> <>.

B: Handmatig lukt het wel gewoon allemaal. Dat is het probleem niet. Het gaat om het geautomatiseerd testen, dat lukt dus niet. Dat was met vorig project toendertijd ook zo. Handmatig geen enkel probleem. Draai je zegmaar de testen op de achtegrond, <>. Fout.

J: De build service is toch altijd weer wat anders. We runnen die ART, hij doet het gewoon allemaal prima. Maar op de build server kan 'ie Chrome niet starten, sorry Firefox. Permission issues of zoiets dergelijks omdat ie een andere user meekrijgt. Ik weet niet precies meer wat het was, maar. En voordat je daar dan achter bent ben je zo een dag verder terwijl je daar eigenlijk niets hebt gedaan. Eigenlijk wacht op een build. Maarja hij moet eerst bouwen.

I: Dit is toch iets wat je dan maar 1 keer tegenkomt toch?

J: Ja maar je komt heel vaak zulk soort issues tegen. En dan ben je zo een week weg zonder dat echt iets gedaan hebt.

R: Vandaar dat ik ook zeg van, centraal zou het ook helemaal hetzelfde moeten lopen als lokaal. Maar omdat je lokaal niet kunt bouwen zegmaar, tenminse het kan wel en dat ga ik ook wel een keer proberen met een jenkins lokaal te draaien. Maar dat is gewoon een overgang. Altijd als je van omgevin veranderd en de omgevingen niet precies helemaal hetzelfe zijn dan krijg je weer een probleem.

I: Ondanks dat je zeg je, ook al draai je het via het dashboard en met Docker dan hangt het ook nog een keer van de load af van de machine. Het ligt dus niet zozeer aan de configuratie van de omgeving maar van de load?

J: Ja

R: Ja, onder andere de beschikbaarheid van resources, Ja ja ja.

I: Er is nu al een splitsing in de resource pool voor ontwikkelstraat en applicatie deployment. Misschien zou het beter zijn om ook een resource pool te maken voor ART's?

R: Ja, zoiets ja. Dat je in iedergeval….

J: Of zorg dat er snellere disk io is ofzo. Ik zie dat dat eigenlijk heel traag is waardoor je ook heel ander gedrag krijgt dan op je lokale machine. Een build duurt op Jenkins acht minuten, bij mij lokaal 1 minuut.

I: Ja dat is wel een fysiek limiet omdat dat ligt aan de onderliggende hardware. Dus daar is weinig aan te doen op de korte termijn.

B: Een goed voorbeeld is namelijk de Oracle database. We starten Oracle op en we vullen dat ding met referentie data. Op mijn PC duurt het laden ongeveer 30–35 seconden. Ik heb meegemaakt

dat als ik dat op het dashboard draai dat het dan vijf minuten duurt. Kijk twee keer zoveel, <>. Maar het verschil tussen 30 seconden en vijf minuten dat is wel echt een heel groot verschil. En het kan zijn omdat op dat ogenblik weinig resources aanwezig zijn. Kan zijn dat het traag is. Misschien gaat het over het netwerk.

J: Dat is dus lastig, want je weet dus niet waarom het traag is.

I: Die Oracle database hadden we getest op een host die rustig was. En dan had je hetzelde effect. Dus dat zijn echt gewoon hardwarematige limieten die je raakt. We draaien bijvoorbeeld niet op SSD's. We gebruiken wel enterprise level hard disks, maar die zijn niet zo snel.

B: Enterprise SSD's zijn ook veel te duur.

J: Maar dit is ontwikkelstraat gebeuren, we hebben geen enterpise grade disks nodig. Wat mij betreft ga je naar de mediamarkt en gebruik je een laptop als host.

I: Oke, de kwaliteitsrapportage, gebruiken jullie die om de kwaliteit te monitoren en te verbteren voor het projecT?

B: Ja, en sommige mensen zijn behoorlijk fanatiek.

<>

B: Dus het wordt zeker gebruikt.

I: En hoe gaat dat? Hoe kijk je daar tegenaan? Tegen het gebruik van dat soort rapportages om de kwaliteit te monitoren?

B: Ik denk niet dat je dit aan een van ons drieen wilt vragen.

<>

I: Maar gaat het dan om de kwaliteitsmanagers?

R: Ja, de kwaliteitsmanagers maar ook inderdaad het B en I bij ons in het team. Die zijn echt <>, ..

I: En waarom zijn die zo fan daarvan, zijn dat ontwikkelaars?

B: Die willen echt 100% hebben. Terwijl voor de meeste van ons voldoende hebben aan 80%.

R: 80% is heel mooi.

I: Ja.

B: dus het komt echt van binnenuit.

J: zij zien het echt als een sport om het goed te krijgen. En om 100% te halen, dat is dan gwoon een dingetje.

I: Maar vind je dan dat het bijdraagt aan de kwaliteit? Als je dingen ziet om de kwaliteitsrapportage, denk je dan van daar moet ik echt iets mee of eerder van daar kunnen we niets aan doen? Of dat zijn regels die we onzin vinden.

J: Wat mij betreft mag er nog wel een beetje een filtering overheen nog. Soms is het rood omdat een regressietest faalt, opzich goed hoor. Maarja dat is logisch als je net toevallig die… of iets heeft een paar dagen niet gedraaid, ja als je nix gepushed hebt naar master ofzo dan is 'ie rood. <>. Dat is niet technisch maar dan moet je wel allemaal weer uit gaan zoeken. Enne, bijvoorbeeld van die dependency checks en dergelijke moet je elke keer weer checken of daar iets is gewijzigd. Van mij mag het wat pragmatischer. Vind het goed voor de lange termijn maar je wordt wel wat

door daar elke dag naar te kijken, ja er is niets gewijzigd. Ja er is niets gewijzigd. Ja het is nog steeds rood. Maja..

R: Ja goed, het punt op een gegeven moment is natuurlijk er is een heel groot verschil <> gat tussen kwaliteit die daar gemonitord word en echte kwaliteit, zegmaar. En dat is dat gat is enorm. En ja goed. Vaak wordt dan wel. Ja het is kwaliteit op het laagste niveau. Het is gewoon hoe netjes een muur gemetseld is maar het is niet of het gebouw is dat nog enigzins functioneert. Zegmaar. Maargoed dat heb je met alle rapportages. Zeker als het percentaegs zijn, Iedere manager gaat daar helemaal op nat zegmaar. . Zovan, dat moet 100% zijn. Weetjewel. Maargoed daar doe je verder niets aan. Dat zegt niets over de of die rapportage tool niet of wel nuttig is. Ja die is nuttig. Het probleem is alleen op een gegeven moment van hoeveel tijd ga je er in steken om dat allemaal overeind te houden. Ten kosten van sommige structurele zaken die je zou kunnen oplossen. Zegmaar in die tijd. Hoe is de structuur van je applicatie op hoger niveau. Daar zou wat meer focus op moeten zijn in plaats van dat lage niveau.

J: Het is nuttig maar erg bureaucratisch.

R:

J: Maargoed, die tool op zich doet het wel goed volgens mij, die kwaliteitsrapportage tool.

I: En Sonar is daar natuurlijk een onderdeel van, daar kijken jullie ook zelf naar? De issues die daar uit komen?

J: Ja, ja.

I: En dat is dan ook een driver om dat te verbeteren over het algemeen.

B: Ja.

R: Jawel.

I: Hebben julie daar ook afspraken met jullie product owner over gemaakt. Zovan we willen elke sprint, weet ik veel, iets doen om kwaliteit te verbeteren of om achterstallige kwaliteits issues op te ruimen, dat soort dingen.

R: Ja soms dan beginnen we wat lager. Aan het begin van het project begonnen we op vijf procent. We hebben gezegd elke sprint gaan we hem omhoog zetten die grens. Dat hebben we wel gedaan.

I: En dat hebben jullie ook afgesproken met de product owner of is dat iets dat jullie zelf tussen de bedrijven door hebben geprobeerd?

B: Het komt wel meer van ons intern, vanuit Ontwikkelorganisatie.

I: Ja, dat snap ik maar je moet natuurlijk die tijd die je er aan besteed ergens verantwoorden. Want dat gaat ten koste van andere dingen.

B: <>

R: Wat ik wel heel interessant is en wat ik geopperd heb. Jongens ga nou eens monitoren hoeveel procent van je tijd je bezig bent met het onderhouden van van van geautomatiseerde testen. Versus tijd die je aan productiecode spendeerd. Maar.. maargoed.. Mijns inziens is die verhouden een beetje zoek. Dat zou je nou eens een keer moeten monitoren. Dat is interessant.

J: Ja het is echt een verschikkelijk kostbaar iets.

R:

J: Plus in combinatie met de hele kwaliteits, dat je de . Alles bij elkaar.

R: Want ja je unit test moet 100%. Je heh, integratietesten dat is bijna hetzelfde inderdaad. ART 50%.

I: Ik heb wel eens gelezen dat dat ongeveer 50/50 moet zijn. Zou je zeggen van dat gaat echt ver er overheen? of?

J: Heel ver er overheen.

R: Ik denk de ART inderdaad wel,

I: En komt dat dan doordat, weet ik veel. De tools die jullie gebruiken of dat de ART tool heel ingewikkeld is. Of is dat omdat je superveel dubbele testen aan het schrijven bent?

R: Ja ik denk vooral op een gegeven moment dat 'ie gewoon onstabiel is die ART.

I: Het is gewoon het uitzoeken van…

R: Ja meestal wel. Dan shit nu is 'ie rood en net was 'ie nog groen. Hoe zou dat nou toch komen. Ja dat is best wel moeilijk om achter te komen vaak. Vooral als het een timing issue is. dus Het geautomatiseerd testen vooral de ART. De unit testen ok. Die kosten misschien best wel veel tijd om te schrijven maar die zijn daarna altijd stabiel. Zolang je maar binnen de applicatie proces blijft met je geautomatiseerde test. En niet asychroon gaat werken, zegmaar. Dan is het wel redelijk stabiel. Dan is het gewoon toegevoegde waarde. Maar inderdaad zo'n ART test met een externe database en timing gebeuren.

I: Zie je dat het instabieler wordt naarmate de omvang van code en testen groeit? Of is dat vanaf het begin eigenlijk al zo geweest?

R: Ja dat is vanaf het begin af aan zo geweest volgens mij toch?

I: Zou je zeggen dat als de ART faalt, dat dat vaker het geval is door die synchroniciteits issues of performance issues dan dat er daadwerkelijk een functioneel probleem is?

R: Ik denk dat ongeveer 95% omgevingsproblemen zijn ja.

I: Dat is erg interessant want dit wil je natuurlijk niet.

J: Het is ook als je iets wijzigd aan het landschap. Als je iest verranderd dan vallen de ART's weer om en dan moet het bijgewerkt worden. Voordat het allemaal weer goed is dan is er wel weer iets veranderd.

I: Ja

J: ze zouden eigenlijk wat later meoten beginnen met de ART's.

B: Even iets anders…. Weet je zeker dat de opname loopt.

I: Ja ja ja ja.

I: Heel goed dat je er aan denkt. Ja hij loopt nog. Nee hij loopt nog. Dank je wel.

B: Ik heb al een keer meegemaakt dat je er na een half uurtje achterkomt; hij heeft toch niet opgenomen.

I: Nee het werkt nog, als het scherm uitgaat dan doet 'ie het nog.

I: . Ja over test cases en het administreren daarvan. Logical testcases enzo. Jullie gebruiken daar ook Jira voor? Hoe verloopt dat?

R: Oeh, dan moet je denk ik even een afspraak maken met Niels. Dat zijn echt de testers die daar over gaan. Daar doen wij niets aan heh?

B: Het meeste wat wij doen is draaien en als er iets fout gaat dat proberen te fixen.

I: Ja precies.

B: De hele administratie daaromheen, sowieso ook de BIRT rapportage en Jira dat doen wij niet.

I: Oke, dan gaan we die vragen skippen. dan zal ik contact met hun opnemen.

B: Volgens mij is dat ook hetzelfde als met andere projecten.

I: Nou, nee. niet altijd hoor. Maar jullie hebben echt dedicated testers in jullie team?

B: Maar die administratie is hetzelde.

I: Bij andere teams zijn er ook ontwikkelaars die testen en dus ook de test administratie bijhouden. Dat is dus niet perse overal hetzeldfe. En ook hoe dat is ingegericht kan ook wel verschillen. En de problemen die daar eventueel uit voortkomen.

B: ok

I: Oja, ik had ook nog een vraag eerder. Als er nou een nieuw teamlid joint. Een ontwikkelaar. Hoe krijg je ze dan eigenlijk. Want het is denk ik best wel complex misschien de hele ontwikkelomgeving en hoe dat allemaal samenwerkt. Hoe krijg je ze dan up to speed? Hoe laat je ze kennismaken met de hele omgeving?

B: Hier heb je Docker.

R: Docker compose up

I: Maar zou je zeggen dat dat helpt? Gegeven eerdere ervaringen met andere projecten. Helpt de opzet zoals we die hier hebben om sneller met een proejct up to speed te komen of maakt het totaal niet uit? Of werkt het zelfs averechts?

R: Ja, Heb je het dan vooral over de inrichting van je omgeving.

I: Bijvoorbeeld.

R: Niet de kennis opbouw. Daar hebben jullie volgens mij niet direct…

I: Nouja, kijk als je bijvoorbeeld geen Docker hebt dan. Nu gebruiken we docker. Dus Docker kennis is wel handig. Terwijl als je dat niet hebt dan hoef je die kennis niet te hebben.

B: We hebben allemaal een eigen IDE, de is vrijblijvend. Dus iedereen heeft z'n eigen keus. Directory structuur is vrij. Operating system is vrij.

I: Maar heb je de server ook gewoon geinstalleerd en zelf de configuratie gemaakt of gebruik je Docker images?

B: Ja docker images. Wat we doen is namelijk ja. Je kunt lokaal nog steeds buiten Docker je applicatie draaien. Maar dat is echt vaak gewoon alleen om kleine dingetjes te testen.

J: Ja

B: Met maven builden we al images. Dus met extra parameter heb je allemaal images. en dan run je je Docker instantie, je hele set.

I: En is dat een fijne manier om te werken? Want ik kan me voorstellen dat je soms ergens mee bezig bent en dan wil je heel snel achter elkaar testen of het al werkt.

B: Shortcuts maken. Dus eh. Een andere manier bedenken om te versnellen.

R: Ik had wel het idee inderdaad dat de eerste paar keer dat ik echt gewoon nadat ik unit en integratietesten had geschreven en ik wil het gewoon even proberen dan draai ik het wel gewoon in een applicatiecontainer. Meerdere keren een container opstarten en debuggen dat schiet niet op.

J: Dat is juist eigenlijk ook het grootste vervelende aan hoe we dat met Docker containers doen. Want als je aan 1 deelsysteem bezig bent moet je gelijk drie andere starten. Drie andere containers. Vaak moet je dan ook even een versietje updaten of even pullen. Of wat dan ook. En dan moet je die even bouwen. Of. Voordat je hem uiteindelijk draaiend hebt en echt kunt testen wat je doet ben je zo een kwart… zo een tijd verder.

B: Wat ik doe is, ik kopieer een war file naar een docker container. Hot deploy. In een draaiende container. Maar dan nog steeds. Je moet een hele war bouwen, je kunt niet rechtstreeks code daar wijzigen.

J: Dat doe ik dus wel door in IntelliJ gewoon explode war en dan een hot reload.

I: Eigenlijk heeft iedereen daar ook z'n eigen oplossing voor. Hoe dat lokaal het beste werkt.

R: Ja

B: Ja

R: Wat ik zelf nog ontzettend fijn vind is dat ik met een lokale DNS werk. Dat werkt ontzettend goed. Met Docker DNS.

J: Bij mij doet 'ie het niet meer.

I: Zou daar hulp bij nodig zijn, het lokaal de omgeving inrichten, of? Misschien kijken hoe iedereen het doet en daar een tiplijst van maken?

B: Maar dat doen wij dus ook. Je kijkt bij iemand anders. Oh, je hebt het op die manier. Nou ik vind die van mij handiger.

J: Maar dat ligt er helemaal aan aan welke deelsystemen je werkt of wat de hoofdmoot van je werk is. als je puur REST API werk doet dan is veel interessanter om even een war in je eigen service container te draaien. Ben je vooral met de frontend bezig dan ben je vooral geintereseerd in NPM. Dan zou je Docker niet eens draaien.

B: Precies, GP doet dat ook niet. Die heeft gewoon z'n eigen appicatie draaien, daar test hij alles op.

R: Wat ontzettend zou helpen is als je iedereen gaat dadelijk z'n eigen laptop mee moeten brengen (notitie: Ontwikkelorganisatie gaat over naar BYOD). Enne, Docker werkt gewoon nog steeds veel en veel beter odner Linux. En, het probleem is zegmaar vanwege security richtlijnen moet je een encrypted distributie hebben. En ik heb nog niet kunnen vinden hoe nou een dual boot kunt maken waarbij je de nieuwe linnux distro die je er langs zet encryp.t als jedaar een standaard oplossing voor bedenkt dat je een enorme boost geeft aan de productiviteit. Nu zit de ene op Mac ander op windows en de ander zit op Linux. zegmaar. Ik zie dat met Mac ontwikkelen zitten ze met de DNS te kloten. Daar kun je volgens mij geen lokale DNS op draaien. Ja andere mensen werken op Windows, dat is ook neitecht fantastisch samen met Docker. Dus als je mensen zou willen helpen, dan zou je daar een oplossing moeten bedenken.

I: Oke, genoteerd.

R: Dat is misschien meer iets voor systeembeheer, maar ik benoem het hier.

I: Of misschien een dekstop-as-a-service? Een remote desktop achtig iets.

J:

I: Goed. Nu dat je dat dashboard hebt, daar kun je mooi je eigen applicatie starten in de ontwikkelomgeving. Jullie weten wel hoe het er eerst aan toe ging, dat de ontwikkelomgeving meer gestandaardiseerd was eigenlijk. Je kreeg gewoon een standaard Jenkins op een bepaalde manier geconfigureerd en daar kon je zelf vrij weinig aan doen. Nu heb je veel meer vrijheid. Eigenlijk alle vrijheid om diensten te starten die je nodig acht tijdens het ontwikkelrpoces. Maken jullie daar ook gebruik van? Of is de standaardtoolset, Jenkins, Gitlab, Sonar, Nexus met Docker registry dan erbij, is dat voldoende?

J: Eigenlijk wordt er eigenlijk geen gebruik van gemaakt. Die instanties draaien en er zit nooit meer iemand aan. Keer proberen een nieuwe versie te draaien, maar maakt het wat uit. Je zit zo snel in het vaarwater van de rest van het team.

I: Maar ook bijvoorbeeld Jenkins plugins. Jullie hebben nu alle vrijheid om plugins te installeren.

J: Ja dat durven we ook niet want dan zijn we bang dat er de pipeline omvalt.

R: Nouja..

I: Jullie maken gebuik van Jenkins 2 en Jenkinsfile toch?

J: Ja.

I: Maar dat is ook iets, dat is door iemand anders bedacht zegmaar?

B: Ja. Door R. Door R opgezet.

I: Daar doen jullie nu dan niets aan? Dat draait?

B: Onderhouden. Want er zijn uiteraard. Er komen nieuwe subsystemen bij. Er zijn wijzigignen die wij willen hebben, dat het zich net iets anders gedraagd. Dat soort dingen.

I: Ja.

B: Dus we onderhouden het wel, maar het is wel opgezet door R.

I: Hoe zouden jullie je kennisniveau van Jenkins inschatten? zijn jullie experts?

B: Jenkins as is, nee. Wat we onderhouden is inderdaad die Jenkinsfile en hoe de pipeline zich gedraagd. De rest van Jenkins laten we staan. Maven plugins, welke versies. Ik weet waar ze staan. Voor het laatst heb ik drie maanden geleden gekeken.

J: Ja er zijn een hele boel die moeten geupdate worden. Tenmisnte die kunnen geupdate worden. Maar dan ben ik bang dat het omvalt.

B: Wat dat betreft is onze productieomgeving veel belangrjiker dan tweaken dat het up to date blijft of wat dan ook. Voor ons is het gewoon productie. en productie is belangrijker dan features. Zo zie ik dat.

I: Maar wat vind je van het feit dat de verantwordelijkheid voor het onderhouden van de hele ontwikkel en bouwstraat bij het ontwikkelteam ook neer te leggen in plaats van bij een beheerteam die dat onderhoud.

J: Echt superfijn, maar dan voor een klein project. Voor een groot project wordt het te groot waardoor je niet… Of dat moet je 1 iemand hebben die dat doet. Dan heb je in principe je beheerclubje binnen je grote team.

R: Ja bij ons is de discussie nu een beetje van. eerst waren er mensen extern voor. Zegmaar. die zijn eigenlijk weg gegaan zegmaar. Fijner zou zijn geweest dat ze niet weg zouden zijn gegaan. Maar gewoon in het team verder zijn getrokken. Veel mensen hebben nu het gevoel van ik kom niet meer aan programmeren toe. Ik ben eigenlijk 80/90% van m'n tijd bezig met het

onderhouden en beheren van de buildstraat.

B: Beheren, echt systeembeheer.

R: Ja dus.

I: Wat voor systeembeheer taken zijn er dan zoal?

B: Onderhoud van die pipelines. Onderhouden van de omgevingen. Ontwikkelen nieuwe feautres. Als je iets wil, vroeger konden we dat uitbesteden aan R. Opzetten van nieuwe omgevingen of tooling. We zijn aan het experimenteren met Nexus drie. En nu op dit ogenblijk, alsjeblieft niet er aanzitten. Want dan valt het om.

I: Het idee is natuurlijk, als ontwikkelaar, weet je het beste hoe je je build pipeline ingericht wilt hebben en hoe je dat kunt streamlinen. Dus je hebt nu ook de vrijheid om te zeggen van, Jenkins, ik wil dat niet gebruiken. Ik wil een andere tool gebruiken. Of ik wil alles in Gitlab gaan doen. Die vrijheid is er.

J: Maar dan moet je eerst als team wat op kunnen bouwen. Het ging van nul naar honderd in het begin en daardoor is het boven ons hoofd gegroeid en nu. We zijn ook heel veel bezig met fixen fixen fixen! Brandjes blussen In plaats van als team dat je er langzaam naar toe groeid waardoor het beter past. Dat is een beetje.

I: Dus in het begin is er teveel van buitenaf gepushed in een bepaalde richting?

J: Zeker, dat idee heb ik tenminste.

R: Ja ik heb daar zelf niet zo heel veel moeite mee. Ik ben zelf ook iemand die, ik voel me meer een generalist. Maakt me niet zoveel uit of ik nou aan het programmeren ben of met Dockerfile aan het werken ben. Of aan de build pipeline. Zegmaar. Maargoed, andere mensen vinden het veel fijner om gespecialiseerd ergens mee bezig te zijn.

I: Ja

R: Ik denk wel dat het goed is dat een team z'n eigen toolset kan kiezen. Want dat testx zegmaar dat is erg opgedrongen van buiten. Tenminse, zo heb ik dat ervaren. Vooral het werk met Excel sheets heeft echt tot problemen geleid. Het argument want ik daar voor aanvoerde zegmaar werd van tafel geveegd. Tot op het moment dat het niet meer werkte. Als je een teampje van vier hebt, en je hebt 1 tester. die alleen maar die testen beheerd. prima. Maar als je een groot project hebt en je gaat met branches werken, dan is het hel op aarde op een gegeven moment.

J: Als we geen ART zouden hebben met TestX, maar gewoon dat de developers gewoon met protractor end-to-end tests zouden schrijven dan waren we een fractie van de tijd kwijt geweest denk ik.

R: Ja, ja. Nu hebben we YAML als formaat. Dat werkt volgens mij ook wel aardig.

J: Als ik zie hoe eenvoudig in principe een protractor script is, is het veel makkelijker dan zo'n hele testX met functions, dingetjes, excelletjes. Dan zit de configuratie een gedeelte in testx en gedeelte reporter, gedeelte protractor, webdriver, selenium. Allemaal bij elkaar, en kijken wat is nu wat en hoe speelt het samen. En welke versies. Vind ik best wel ingewikkeld.

I: Jullie hebben dedicated testers in het team, wat volgens mij geen technische mensen zijn? Het hele TestX is natuurlijk verzonnen zodat dat niet technische mensen testautomatisering kunnen doen. Vind je dat dan dat je dedicated testers hebt eigenlijk niet fijn? Had je liever het testwerk gewoon onder de developers. En dat je dan bijvoorbeeld twee extra developers had gehad?

J: Ja dat is lastig. Mijn persoonlijke mening is als je dus dedicated testers moet hebben. Dan zou een developer ook helemaal niks te maken moeten hebben met die ART. Want daar heb je dus die functionele mensen voor.

I: Ja inderdaad, maar in de praktijk is dat dus niet het geval?

J: Nee.

I: Want wat moeten jullie dan doen voor hun? Moeten jullie TestX uitbreiden? Of wat is jullie werk?

J: Uiteindeljik moeten we de ART's werkend houden. Docker infrastructuur onderhouden, de compose files voor ze onderhouden, af en toe een ART fixen.

R: Vooral de compose files . En wat ook nog wel veel tijd neemt voor die ART is dat we inderdaad lokaal heb ik een Docker Compose configuratie gemaakt. nu onderhouden we een tweede applicatie definitie speciaal voor het Dashboard. Ja goed omdat je best wel veel componenten hebt is die dubbele configuratie onderhoud dat vreet gewoon tijd op. Zegmaar. Dus daar zou een stukje Docker Compose ondersteuning wel helpen.

B: En we hebben ook nog een testomgeving, ketentest. Dat is anders. Dat is look-alike net als bij . En dus dat is heel anders. Dat is niet een dashboard. Het zijn wel Docker Images en containers maar die registeren zichzelf. Die port forwarding en dat soort dingetjes. Die draaien volledig buiten alles om.

I: Draaien die bij ? Of die draaien wel bij de Ontwikkelorganisatie?

B: We hebben een kopie van de omgeving hier. Een versimpelde versie van de klant, zegmaar.

J: Het betekend dus dat je per deelsysteem zes compose files hebt. Docker Compose plus een override, Art, Art-override, ketentest en ketentest override. Die moet je allemaal bijhouden dat is als er wat veranderd.

I: En de applicatie definities voor het Dashboard?

J: Ja die worden automatisch geconverteerd en geupload. Dus daar moet je af en toe wel eens iets aan doen maar niet zoveel. Alleen als je iets custom wilt draaien. Een hele boekhouding best wel.

I: Laatste vraag is dan, hoe kijk je aan tegen de ondersteuning vanuit de Ontwikkelorganisatie voor het werk dat je doet? Bijvoorbeeld ondersteuning bij de ontwikkelstraat, testwerkzaamheden. Wat vind je van de kwaliteit van de hulp. Ik heb jullie horen zeggen dat jullie in het begin van het project misschien een bepaalde richting zijn opgeduwd. Bepaalde tooling, een bepaalde manier van werken. Er zijn hulptroepen weggenomen, wat vind je van de verdere support? Is dat te weinig?

R: Ik vind dat het wel prima is. Kijk <> als je nou een voorbeeld neemt. We hebben zegmaar lange tijd een tekort aan resources gehad. Dat had misschien wat eerder opgepikt moeten worden. Maar dat is een kwestie van naar jullie toe gaan en een afspraak maken. Dat moet denk ik van twee kanten komen. Ik heb me daar verder ook niet zoveel mee bemoeid. J wat vind jij er van?

J: Resources had gewoon twee maanden geleden al opgelost moeten zijn. Dat heeft teveel tijd gekost. Hoeveel uren we daar als team zijnde mee kwijt zijn geweest. Dat is echt gigantisch.

R: Maar hoe kwam het dat het nou zo lang heeft geduurd?

J: Ja, volgens mij omdat we op toestemming moesten wachten. Heeft met kosten, organisatorisch of met veiligheidsredenen te maken.

I: Volgens mij zijn er ook gelijdelijk in stappen extra resources vergeven. Maar het klopt dat het eerst goedgekeurd moet worden. Daarnaast, hoeveel ruimte je ook geeft, als ongebruikte resources niet meer worden vrijgegeven dan zul je altijd een punt bereiken waarop er tekort is.

J: Op een gegeven moment bereikte we een punt waarop we een paar keer per week moesten

opruimen.

R: Ja een paar keer per week inderdaad.

I: Volgens mij is er nu ook een proces ingericht zodat jullie dat makkelijker zelf kunnen doen.

B: Vroeger was het zo dat we aan het eind van de sprint alles moesten opruimen maar dan houden we dus een stuk of 300, nee 200 over van de 800 images. We moesten jullie vragen om dit te doen.

I: Dat betekende ook dat jullie altijd afhankelijk waren van ons. Dat is nu dus niet meer zo.

J: Het is heel stom geweest om niet eerder naar jullie toe te komen over die Jenkins workspace. Die is ook een paar sprints lang onvoldoende groot geweest, waardoor we drie keer per dag moesten opschonen. Nu die ruimte 50Gb is, is het zo veel beter.

R: Maar misschien ook beter en naar de juiste personen escaleren en niet zelf lang aanmodderen. Jullie hebben toestemming nodig, maar we kunnen ook zorgen dat er vanuit ons meer druk op gelegd wordt.

J: We denken ook te snel, dit is wat we hebben en daar moeten we het mee doen in plaats van kunnen we niet meer krijgen. Kunnen we niet meer krijgen?

<>

I: Goed. Dank voor het gesprek.

Team 3

Property	Value
Date	09-12-2016 15:00
Duration	44 minutes
Present	***I:**** interviewer
	R: developer 1
Team members	3
Team size	Small
Project size	Small

Transcript

This paragraph contains the annotated transcript of the interview. Three developers took part. Since the interview was in Dutch the transcript is also in Dutch. Annotations are in English.

I: Dat wordt natuurlijk opgenomen.

R: Ja, Ok.

I: Euh… of er inderdaad euhm ja hoe, hoe ziet je project eruit? Dus hoeveel mensen werken daaraan en wat voor applicaties. Dat soort //ja// zaken.

R: Euh op dit moment euh… zijn we met het eerste deelproject bezig. Euh KV-versie 1.O. Euhm… dat bestaat uit drie euh ja, werk producerende teamleden. Euh… ik als ontwikkelaar… euh doet de database werken tot aan de (00:27).

I: Ja.

R: En Gis, een ETL-expert. Dus die doet de datatransformatie voornamelijk, in een apart doel, informeren. Doen we verder niets mee. Dat kunnen we alleen aan. En de tester en die draait de RT en doet testen met de hand en die gebruikt de (00:43) eigenlijk…

I: En ook test automatisering euh…

R: Ja, met euh met Testics //ja//. Op dit moment doen we voornamelijk euh… database euh validatie en euh sturen we de, de applicaties weer vooraan die, die ETL uitvoert.

I: Ok.

R: Dus dat is eigenlijk hoe het nu in elkaar zit. In het Nieuwjaar gaan er drie deelprojecten st- verder //ja// starten we en wat we nu doen dat ronde we af en gaan we door naar andere projecten. En dan euh komen er een hele boel nieuwe dingen bij kijken. Euhm…we gaan- Python is de enige tool die we mogen gebruiken voor backend. Euh die zeggen Python euh…

I: Ja, dus vanuit de klant een euh…

R: Vanuit de klant, ja. Euh… dat gaan we doen. Er komt een dotnet plugin voor Autocat in. Dat is een ander project.

I: Ok.

R: En daarna gaan we wat we nu gebouwd hebben gaan we ombouwen euh van het tool naar ook een python, python applicatie. Euh… dat gaat als het goed is landen op een (01:37) landingen. Meerdere (01:39) zijn afhankelijk van welk project je wil.

I: Ja, precies //ja//. Want waar gaat het project? wat is precies de inhoudelijk project?

R: Inhoudelijk, er zijn vier organisaties binnen euh… de overheid //ja//. Vastgoed organisaties //ja//. Euh… die worden samengevoegd tot een organisatie //ja// het rijks vastgoedbedrijf RVB //ja//. Euh met een onderdeel van defensie, onderdeel van de publieke sector en dat wordt nu samengevoegd. En dan hebben ze allemaal losse systemen die min of meer hetzelfde doen. Alleen met hun eigen stukje data en die worden in een portaal in euh… een bewerkingsapplicatie, in Autocat bijvoorbeeld//euh euh// euh… en een brondatabase kern registratie. KRV daar werken we nu aan. En dat zijn echt de… het zijn nieuwe dragers van, die de nieuwe organisatie… daar zit alles in dat ze moeten weten. Die heeft de, de informatie komt erin, er wordt een unieke ID opgegeven, die bewaren en de rest is eigenlijk (02:30)//euh euh// en dat exporteren we naar een andere systeem. En die ID die wij eigenlijk genereren, opslaan die is uniek door het hele bedrijf en dat is, dat h- tegelijk ons systeem is eigenlijk de spin in heel de…

I: Maar jullie maken dus nu software om euh… dat te vervangen van die… personen?

R: We hebben, - we hebben, zeg maar nu met meerdere data bezig //ja//. Die doen we met die tool die we nu hebben, HTML-tool //ja//. Maken we daar nieuwe data bij//ja, precies// dat is de bron //ja, precies// en… euh dat- daar wordt alles op aangeknoopt, straks.

I: Ok. Maar die achterliggende databases blijven bestaan?

R: Op de duur gaan die eruit. Als// Want als// Als ze over een halfjaar zetten ze een deadline, een politieke deadline en dan worden de stekkers uit alle systemen getrokken en dan moet het nieuw applicatie zijn… die al die functies … die oude bieden, euh moeten daar een nieuw systeem inzitten.

I: Ja, ok. Dus dan de team is dan drie man groot. Een ontwikkelaar, tester en ETL-expert.

R: Ja.

I: Ok, euhm… en euh Python, je zei dat is een eis van de klant. Weet je ook waarom?

R: Euhm… onduidelijk waarom het is, maar er is een euh een lijst…een groene, een grijze en een rode lijst. //ja// grijs staat Java en Dotnet op. En…

I: Dat hebben ze liever niet?

R: Uitfaseren. Alles wat nieuw is moet Python en daar komt het op neer. En daar voor het web is het euh…

I: Interessante keus.

R: (03:53) en wat noem je, ja, wat Javascript frameworks. Maar geen engelen dus.

I: Maar dat is iets van wat hun eigen lijst. Wat ze zelf…

R: Ja.

I: Ok.

R: Euh binnen (04:04) ook nog bedacht of overwogen om de opdracht euh terug te geven omdat hier geen expertise voor Python //euh euh// is. Maar dat gaat nu al genoeg.

I: En jullie gaan die expertise gewoon opdoen voor dit project of?

R: Euh… ja, sowieso ben ik bekend met Python.

I: Ok, Ok.

R: Maar ze nemen bijvoorbeeld personen- moeten kwaiteits… //ja// review gebaard worden.

I: Ja, ja.

R: Dus, we gaan het gaan doen.

I: Maar dan volgend jaar wordt het project uitgebreid met meer ontwikkelaars?

R: Ja. Dus we worden- twee teams komen erbij //ja//. En wordt het (04:30) maar we zitten nu nog in de voorfase //ja//. En dat gaat pas in Januari, Februari door.

I: Ja, precies. Ok.

R: Dus euh.

I: Ok. Euhm en over euhm het- ik ga het dan hebben, gewoon specifiek hebben over iets als controleren of zoiets //ja//. Hoe controleren jullie of het geven van een (04:48) test omgeving ofzo //euh euh//. Euhm doen jullie dat? Doen jullie meerdere beelds per dag of deployments per dag?

R: Euh ook de check-in levert een CE en eigenlijk ook een (04:56) een potentiele //ja//. Ja, en aan het einde van de sprint als ze zeggen, je bent klaar dan voeren we ergens het nummertje in en dan euh versturen we het in (05:05).

I: Maar je maakt ook nu gebruikt van Docker al voor de deployments of dat niet?

R: Euh… nou we beginnen met Docker Images //nee//. We maken wel gebruik van Docker Images om euh alles klaar te zetten en te doen. Euhm… voor protector gebruiken we er een. Testics //ja// euh ga ik ook voor zo'n tools niet alleen op Jenkins daar gebruiken we Docker Images voor. Zo heb ik het ingericht.

I: Ja, en daar die applicatie zelf wordt dan niet gedeployed.

R: Neen, we hebben daar een euh server. Een FME-server heet het tool //ja// en daar kunnen we een staaltje wat wij geproduceerd hebben, wat er in de Git staat sturen wij daar naar toe. En kunnen we met een los euh reiscommando zullen we zeggen starten.

I: Maar waar draait die server dan?

R: Dat is gewoon de virtuele server hier ergens.

I: Ok. Dus een…

R: Dat is een, dat is een (05:52) server //ok// opgericht door, door Jelle.

I: Ok, en waarom is de- hebben jullie voor- of niet… waarom hebben jullie overwogen om dat op Docker te doen of niet?

R: Euh dat is allemaal voor mijn tijd.

I: Dat is allemaal ervoor.

R: Ik euh ik ben er later bijgekomen. Dan was die server er al.

I: Ok.

R: Maar die tool die wij gebruikt hebben daar is eigenlijk zogezegd, die mag niet meer gebruiken, punt. Dus nu mogen we- worden we gedoogd om het wel te gebruiken //ja// voor versie 1.0 //ja// maar wat we nu nieuw gaan bouwen moeten we- Is eigenlijk alles wat we nu gebouwd hebben moet opnieuw in Python gebouwd worden.

I: Ja, ja. Ok.

R: Dus is een beetje, ja. Politieke krachts (06:28) //ja//. Waardoor die beslissing gemaakt is om het terug te draaien. Dus en het idee is nu dat wij waarschijnlijk tijdelijk beheer als ik die zijn kan hosten. En dan de (06:40) die wij nu hebben die stuurt dan die //ja//die productieserver.

I: Juist, juist. Dus het resultaat zeg maar wat jullie nu aan het doen zijn wordt al in onder tijdelijk beheer gehost ook.

R: Ja, als het goed is wel.

I: Als het goed is wel.

R: Ja. En wat we produceren is niet meer dan in Nexus daar stoppen we dat weg. Is een zip met een database script die de bron- euh sorry, de doeldatabase//euh euhm//opzet. Euh en een ETL-bestand die, die ik naar de FME-server kan toesturen en kan uitvoeren met de juiste apparaat dus. En het idee is dan een aantal bronapparaten worden euh gepakt. Die worden gehusseld en die worden naar de doeldatabase geschreven//euh euhm//. En aan de hand daarvan wordt er een export gemaakt. In dit geval Cc en die gaat dan naar de klant. En dat is, zeg maar, de scoop van versie 1.O., dus een paar databases erin,//ja, ja// wisselen dan komen er wat Cc's uit die ik ga…

I: zeg maar, zeg maar het is resultaat is- zijn bestanden dat, dat is wat je oplevert.

R: Ja, ja //ok//. En versie 2.0 is dan een database met een heel applicatie landschap erop eigenlijk.

I: Juist, juist. Die database of die data import nu voor die tool? Dat zijn geen sta- op zichzelf staande databases?

R: Euh nou we krijgen een dump van een database //ok//. DBA's als echte (07:53) opgetuigd //ja//. En die lezen wij als een bron in//precies, ok//.

I: En dat is een stuk wat euh, zeg maar de DBA regelt//ja// regelt voor jullie.

R: Ja, en de volledig eindsituatie. Dus we worden, die dump wordt waarschijnlijk op SRP gezet. Het wordt dan al dan niet automatisch naar de database gezet die wij inlezen en dan ook niet gescript en triggered dat die ETL die plaatsvindt. Cc is eruit en dan terug naar de klant, maar dat gaat nog gebeuren.

I: Dus zeg maar van het stukje Docker platform en dat soort zaken gebruiken jullie met namen gewoon dingen als Jenkins, Sonner, (08:28).

R: Op dit moment wel //ja//. Ja, euhm we gaan voor het nieuwe traject gaan we wel Docker Images gebruiken //ja//. Zo de de situaties van de klant willen nabootsen. Die hebben een heel uitgebreide lijst van wat ze allemaal hebben //ok// en dat gaan we naar Docker container of Docker Images eigenlijk nabouwen en die plooien onze applicatie erin. En dan gaan testen.

I: Ja, dus dan ga je ook echt meer gebruik maken van dat euh meer frequentere deployal.

R: Ja, dat klopt. Dat is, ja dat wordt denk ik, geen idee hoe dat nu gaat, maar dat is wel het idee. We hebben ook met andere projecten dat gedaan. We gaan Docker Images, ja, Docker Image hebt voor een specifieke versie van euh van de applicatie.

I: Ja. Ok, duidelijk. Euhm… ja, dus omdat je nu dus niet gebruikt maakt van een heel deel delivery, overhalen in combinatie met Docker, maar… zijn- zijn er nu problemen daar waar je tegen aanloopt of…//euh..// in het verleden misschien bij andere projecten?

R: In het verleden heb ik dat wel, daar heb ik ook een lijstje van gemaakt//ja, precies. Dat euh…//. Euh waar wij laat pas achter kwamen dus voor euh voor het registratie passpoortsignalering project //ja// euh daar maakten we gebruik van Docker Images, hebben we (09:41) scripties uitgevoerd en dat was die plooiing van de applicatie in de image //ja//. Euh dat ging goed. Euh maar na, wat is het? 1-2 maanden was onze systeem al vol. Euh en dat was een van de problemen die we hadden. Dus…

I: Maar ik bedoelde dus als de harde schijf dus euh was even vol…

R: Dan krijg je honderd gigabyte //ja// volgens mij en dat was gewoon vol. We hadden, wat is het, Docker Image van twee gig ofzo //ja//. Moesten we naar, stonden we (10:06) in. Nou, dan had je dan vijf beeld per dag geproduceerd. Minimaal dan. //ja// Dus daar hadden we wat kunstschepen voor uitgehaald, maar je hoeft dat- daar misten we in. Het inzicht van wat is dit //ja//. En euhm… Ja, dan gewoon een Jenkins job hebben we daarvoor gebouwd die euh niet meer deed dan een euh Docker PS euh wat (10:29) en dan daar een selectie overheen die euh die de boel weer weggooiden en dan was die even goed. Missen we die (10:38) en zo de resten die daar euh//ja, ja// daarmee. Zodat we weer de schijf konden houden //ja//.

I: dus het probleem was concreet gemaakt, je maakt eigenlijk Images van steeds grofweg twintig gig in grote, toch, of?

R: Ja, twee gig.

I: Of twee- drie//ja, twee gig en als je dat dan vijf keer per//per uur doet, ja// dan gaat het heel erg// ja, ja//.

R: En dus…

I: Het probleem is dat je, dus dat is op zichzelf geen probleem, denk ik. Natuurlijk want de harde schijf loopt vol, dat is de consequentie daarvan, maar het probleem is dat het niet inzichtelijk is.

R: Niet inzichtelijk en eigenlijk wil je daar ook een oplossing van die, die kan, ja een (11:10) die bewaard de laatste vijf //ja//. Euhm maar goed, dat is een vraag van hoe snel de tester test. Als jij met twee of drie ontwikkelaars bezig bent gaat het best hard //ja//. Euhm…. Ja

ideaal gezien, wat ik zou willen is van bewaar deze en gooi deze weg. //ja// Dat je dit ergens kan markeren//ja, ja//. Dat het wat gemakkelijker kan. Euh die (11:34) kan het heel makkelijk, maar euhm…

I: Ja, want wat was het voor jullie makkelijk om Docker (11:39) meer is dat eigenlijk niet zo makkelijk.

R: Ja, ja. Ik kan nog altijd een Jenkins job gemaakt die de eerste- dit is allemaal beschikbaar en gooi dit weg. Het is gewoon van die AP's gebruik maken//euh euhm// alleen euhm voor een ontwikkelaar is dat wel te snappen, //euh euhm// maar voor testers of andere niet. En dat is een beetje het probleem als ze…als dat euh alle ontwikkelaars weg waren en dan was er net een probleem. Het was vol en dan stopt alles gewoon //ja//. Dat euh da's lastig. Dus een beetje een afweeg van hoe mooi maak je je pijplijn. Eigenlijk met euh hoe, hoe ver kan iedereen het snappen eigenlijk.

I: Ja, maar zou je dan zeggen dus dat die euh Docker registrie euhm tool eigenlijk euh is ontoereikend dus…

R: Ja, de registrie is eigenlijk ontoereikend//ja, ja//. Die is gebruiksvriendelijk voor als je weet hoe het werkt //ja//maar daar houdt het op //ja//. Ja.

I: ja, dat is een veel gehoord probleem. Nou //ja//.

R: Maar hoe dat hetzelfde is van Grit. Grit is prima als je die euh als je het via (12:40) gebruikt voert het je commando lijstje uit. Maar als je echt Mers conflicten krijgt en je weet niet wat er gebeurt //ja//. Dan moet je toch eens naar een grafische tool euh //klopt, ja// en dat euh nou dat is het verschil tussen een expert en //ja// iemand die, die gedwongen wordt om ermee te werken.

I: En, en dit specifiek probleem hoe vaak denk je dat dat voorkwam dan? Ja, dus je zei om de honderd gig twee gig (13:05)//uiteindelijk… eind van die weken//. Ja, precies.

R: Om naar het einde toe dan zag je echt van, ok, nu piept en kraakt alles. Ja, ja dat ging nou veel sneller, maar goed dan was het project ook bijna afgelopen. Dus vandaar dat er ook geen tijd meer in besteed hadden. Het was meer gewoon pleisters plakken. Wat kunnen we weg, weggooien en dan kunnen we weer een weekje vooruit.

I: Ja, precies dus gewoon de (13:28) kunnen onderstrepen om het euh.

R: Ja te… en het project had ik erna had gedaan dat maakte niet gebruik van Docker Images. Dat was alleen (13:39) voor het applicatie zelf dus de (13:40)//euh euh// die opnieuw opleveren en de zip file voor de web, de front end, die werden gewoon in Nexus gedownload en dat was… dat was zo klein in vergelijking met een Docker Image dat je daar helemaal geen last van hebt.

I: Maar welke, welk project was dat precies?

R: Dat was slachtofferportaal.

I: Ah, slachtofferportaal //ja//. En hoe werkte de deployment daar dan? //euhm// gewoon op V(14:04) of nog?

R: uiteindelijk hebben we V(14:07) ik weet niet of je dat meegekregen had, maar we hadden, we waren weggegaan op euhm Docker Images //ja// gecopy paste van euh de paspoortsignalering project //ja// en we liepen wat tegen problemen aan dat we met Engine X dat we daar gewoon kant-en-klare images en dan deployde we hem en opeens was de service gestopt. En Engine X was gestopt in de container. //mhmm, ok// Nou, daar hadden we mhhm daarna hadden we besloten van wat zijn we nu aan het doen? De A- en de euh T- en de P-omgeving die zijn euhm allemaal als virtuele machines //ja//. Dus dan zitten we op O machines te //ja// te gebruiken en het werkte niet. Dus daar is het toen eigenlijk gedacht van, ok, we leveren (14:48) op, die

deployen ergens, dat is het //ok//. Dus geen Docker euh Images die we daar gebruikte. Euhm en dat was dus niet meer dan (14:58) die hun gegevens euh gedployed werden en op Engine X de omgeving, de front-end, en dat was het. Dat was zeg maar de (15:05) en dat konden wij wel helemaal van euh van D tot en met A kunnen we dat helemaal automatisch deployen//euh euhm// en P vanwege de (15:16) konden we daar niet bij, maar daar kon aan de hand van de Nexus (15:18) konden we zeggen dit moet gedployed worden. Toen ging het goed.

I: Ja, maar hoe werkte dat testen dan? Want je zei, je kon er automatisch testen gewoon deployen? //euh euhm// Euh dat deden we wel voor de testen neem ik aan.

R: Ja, alles, er werd euhm… kijken. Ja, er werd op de T of de- ja, de T werkte alles automatisch dus dat is eigenlijk gewoon een (15:35) beeld die erop gebouwd is. Dus als je een push deed dan springt het automatisch naar hun versie verder //ja//. En als dat goed bevonden was dan kon je in Jenksen een knopje drukken en dan ging die naar de aangeving. En dan kon de tester daar en de klant ook valideren //ja// wat het nou was.

I: Maar dat waren gewoon verschillende virtuele machines die voor de verschillende omgevingen, verschillende omgevingen deden //ja//, ja. Maar dat was een (16:04) project qua?

R: Qua doorlooptijd?

I: Ja, bemensing.

R: Euh twee backend developers, een front-end, en een tester. Denk ik dat het was, euh ja. Dus met z'n vieren, ja.

I: Maar bijvoorbeeld in test doorlooptijd was dat ook niet euh extreem hoog ofzo. Om alle testen uit te voeren.

R: Nee, nee. Dat was allemaal euh heel, heel kort. Het was voornamelijk handwerk. Euh het project was zo ingezet, we hadden een front-end. Het was niet meer dan een geraamte die aan de backend wat (16:30) inlas //ja// en dat toonden //ok//. En dat (16:36) fase aan de achterkant genereren dat wa- was het merendeel van het werk dus euh… dat is wat allemaal los van elkaar zit. Dus kon allemaal vrij gemakkelijk aan elkaar gevist worden. Dus euh…

I: Ok, euhm… ja, nog meer problemen zowat meer (16:55) wat, wat heb je ervan afgedaan?

R: Euh dat waren niet echt problemen. Euh ik ben wel, ik weet wat ik wil. Ik weet wat ik doe, ik heb met andere beeldsystemen//euh euhm//ook wel ervaringen gehad op vorige projecten. Euh ik red me wel. Ik kan (17:09) Windows descripting kan ik allemaal wel. Euh alleen wat er, wat me opvalt is dat elke keer dat je euhm opnieuw het moet uitvinden//euh euhm//. Euh op jullie blog euh IQ-team blog zie je weleens wat dingen van namelijk Docker, nou, dat vond ik helemaal anders. Euhm maar wat ik mis eigenlijk is meer een naslag werk en euhm wat voorbeelden //ja//. Want ik kan ook niet//wat documentatie// documentatie van kijk dit is hoe, dit is nu volgens mij door Jan-Piet en euh soort van (17:40) pijplijn bedacht met euh//euh euhm// (17:43) plugin. Als je daarvan- aan de hand daarvan zie je een voorbeeld die je doet met wat copy paste, dat zou wel heel helpen. Euhm..

I: Nou, wat is dan het achterliggende probleem waarmee je om…

R: Euh nou, stel ik kom nieuw binnen //ja// euh ik euh ik krijg een Jenkins portaal met waarschijnlijk een lege (18:05). Succes. Euh dan moet je als je echt nieuw bent mensen gaan bij elkaar gaan zoeken die je ook kunnen helpen. Euh en zeggen hoe zit ander werk in elkaar, zo kan je dit doen //mmhhm//. Maar hebt je nog iemand van het team die helemaal losgaat op basis van zijn eigen kennis. En wat misschien handig is- wat ik zou verwachten is bij die pijplijn is dat daar ook een soort van visie is. Kijk zo, hier willen we naartoe. Dit is het idee erachter. Als je het wilt kan je het ongeveer zo doen //ja// en aan de hand daarvan wat voorbeelden //ja//. Euhm nou, daar is het dat het uit elkaar kan lopen als euh verder in de tijd bent. Dus het moet dan wel onthouden worden ofzo iets //ja// gegeven worden. Euh maar dat is meer een

handigheidje. Het is niet een probleem, het is meer een handigheidje. Euhm voor de rest, ja, ik vind het wel een mooi systeem eigenlijk. Alleen euh- je moet zien, je moet weten wat je doet.

I: Ja, dat geldt voor veel dingen denk ik.

R: Dat geldt voor veel dingen, ja. Maar dat zie je wel dat mensen daar heel anders tegen aankijken. Zeker als ze geen ontwikkelaars zijn dan snappen ze wel wat er gebeurd //ja// er wordt vaak wel met de commentaar bijgeschreven, doe dit. Maar ja, het is toch lastig om te vatten.

I: Ja, als jij al eens voor een ontwikkelaar zorgt want euh ik weet niet hoe jij er dan tegenaan kijkt, maar het idee achter natuurlijk het hele euhm hoe het, hoe je het doet, zeg maar opgezet met Docker, het platform en alles. Dus, dat je zelf meer controle hebt over welke toolen je gebruikt in je ontwikkelstraat en hoe je je (19:32) genereert en toolen die je daarvoor nodig hebt. Je kunt (19:35) dingen starten op een relatieve simpele manier denk ik//ja, dat klopt//. Maar dat automatisch legt dat natuurlijk meer verantwoordelijkheid bij het ontwikkel team voor het beheer daarvan.

R: Dat klopt. Ja.

I: Euhm, ja. Hoe kijk jij daartegenaan want sommige zeggen van, ja ik ben ontwikkelaar. Ik wil me daar niet te veel mee bezig houden. Maar…

R: Euhm, ja. Ik, ik ontwikkel. Ik vind de vrijheid fijn//euh euhm// nog zeker, maar soms werken dingen niet en dan- als je dan zelf in kan grijpen is het heel fijn //euh euhm//. Maar het zou niet altijd moeten- hoeven. Euhm daar, als je daar een soort van standaardoplossing hebt, bijvoorbeeld, een simpele front-end applicatie, ja daar hoef je niet ingewikkeld voor te doen //nee//. Dat kan dan standaard zijn. Euhm maar als je bijvoorbeeld, wat we nu doen, nu hebben we iets zo afwijkend is, wat we euh want (20:25). Dus we hebben een aparte ETL-server waar we wat bestandjes naartoe sturen en dan weer terug. Ja, dat is niet een standaard//euh euhm// methodiek daarvan. Dus daar moet, daar moet dus iemand- daar moet handwerk op maat aan te pas komen. Maar als je een Javaapplicatie hebt die je in (20:38) gedeport wordt. Ja daar kan je wel wat voor verzinnen dat standaard is //ja//. Euhm maar goed ja, dat is een beetje afhankelijk van het project. Hoe ver je daarin moet gaan, denk ik.

I: Ja, nee. Uiteindelijk is het toch inderdaad om het project natuurlijk te helpen en euh in de eerste instantie met het geven van vrijheden om dingen te doen zoals zij goed den- zoals zij denken dat goed is en past bij hun project //ja//. Maar inderdaad een euh ja dat horen we natuurlijk wel vaker. Want sommige lijken gewoon heel erg op elkaar. Het zijn inderdaad vooral die Javaprojecten die naar hier gedeployed worden. Dus daar euh zijn we mee bezig //ok//. Sowieso euhm een van de dingen natuurlijk als een als een project hier begint krijg je eigenlijk alleen maar dat Docker deskboard//dat klopt, ja// dan is het van euh succes//succes, ja//. Volgens mij starten we meestal nog wel euh wat tooltjes Jenkens, zo heel veel dingen dat iedereen ook echt wel gebruikt. Maar dat is het dan wel. Dus die hele wiring daarvan moet je ook euh eigenlijk zelf doen op dit moment.

R: Klopt, ja. Wat je vaak ziet is dat er euh in alle stress de (21:38) van het project zijn maar net, alle tussendoor de euh zeg maar die, die hele infrastructuur opzet /ja//. Zeg maar dat jullie na dat jullie het hebben opgeleverd. Euhm maar dat het vaak ook houtje-touwtje is//euh euhm// en euh dat zag ik nu bijvoorbeeld, euh het nieuw deelproject van start. Ik heb nu zeg maar, ik kan dat wel doen en dan doe ik dit ook op eigenlijk op mijn eigen manier. Euhm, maar goed, dat doe ik- je ziet dat ik gewoon, dat ik mijn draai daaraan geef en Eduardo die doet dat net op een andere manier en Jan-Piet doet het ook op een andere manier. Je ziet wel dat dan- dat je daar verschillen gaat krijgen en euh met de (22:18) zoals het nu is. Met die, met die plugin dan zie je wel dat het wat standaard wordt. Dus dat je dan gewoon scriptjes hebt waar je…

I: Euh je bedoelt die euh die Docker euh of wat bedoel je? de Jenkins (22:26) eh? //ja// met een Jenkens bouw van//ja, ja dat klopt, ja//.

R: En dan worden gewoon wat schelscript en Git gedaan en die, die worden uitgevoerd //euhm// en dat maakt het dan weer wat standaard dus dan heb je zeg maar weer de infra gescheiden van de applicatie specifieke //ja// acties die uitgevoerd worden. Dus dat vind ik wel mooi //ja//.

I: Ja, want anders krijg je wat je zegt, inderdaad verwarring. Dezelfde soort oplossing voor verschillende projecten die net op een andere manier eigenlijk zijn uitgevoerd en //ja// alles werkt net even iets anders //klopt// afhankelijk van degene die dat doet. Klopt//ja, ja//

R: Daar wordt je toch meer gedwongen om binnen het standaard framework //ja// dus de… ik denk dat het wel goeie, goeie werk (23:05). Euh, ja volgens mij zijn het alle punten die- ja, wat ik anders vind is dat je documentatie van Docker, de API van de (23:16) desktop//euh euhm// en eventueel wat voorbeelden dat zou ik wenselijk vinden. Maar //ja// uiteindelijk is het allemaal wel uit te komen als je, als je een keer langsloopt of je kijkt bij andere projecten af. Je komt wel uit, maar het kan altijd ge…

I: Het komt daarop neer een meer gestandaardiseerd, initieel iets wat gestandaardiseerd is zodat je eigenlijk sneller kunt beginnen, misschien.

R: Ja, want euh…

I: En euh documentie.

R: Ja, ja. Zie je het eigenlijk, misschien is het mooi als je een euh standaard (23:43) hebt. Wat eigenlijk als voorbeeld dient waar je van kan copy paste //ja//Even van let op, Dit is de standaard manier als je hiervan afwijkt dan kan het zijn dat je… dat er iets gebeurd wat je niet verwacht of je euh niet, niet iedereen kan je helpen omdat je afwijkt van de standaard dus dat is wel een beetje een afweging //ja//. Maar dat zou mooi zijn als dat is //ja//.

I: Even kijken want ik heb euh nog wat vragen. Misschien zijn ze niet allemaal relevant euhm… ja heb jij, heb jij gebruik gemaakt van de mogelijkheid om nu inderdaad gewoon euh willekeurig services te starten die jij nuttig vindt voor je, voor het ontwikkelproces, zeg maar. Dus naast Jenkins zou (24:25)//euh ja//. Zoals?

R: Euh (24:30) de klant die wil (24:36) krijgen//euh euhm// als euh als release //ja//. Dus dat, dat soort dingen gebruiken. Dus meer om het proces wat we meer, ja, te versnellen eigenlijk //ok//. Dat soort services moet je denken. Euhm…

I: Maar waren het echt een aantal of alleen…?

R: Ik zit echt te denken, ik heb al een aantal (24:50), maar ik heb ook wat meer ja af en toe heb je wat dingen nodig. Ja, je kan het lokaal draaien /euh euhm//. Maar soms moet het wat langer draaien //ja//. Dus dat soort dingen, dus meer (25:00) euhm makkelijk te maken. Euhm meer?

I: Dus dat was al euh gezegd, zeg maar. Omdat het mogelijk is.

R: Ja, dat verandert helemaal//ja, ja//. Dit is- het zijn maar- het, euh dat portaal (25:20) definitie in kan plakken, ja, is prima, dat werkt echt goed. Ja.

I: Want hoe zou je dat dan doen als je inderdaad dat iets zou hebben he. Dus jij zegt je hebt maar, je hebt eerder geschetst bij ander projecten. Je maakt gebruik van gewoon wat statische VM's//euh euhm// voor de verschillende omgevingen en stel je hebt zo'n (25:35) nodig. Hoe zou jij dat dan oplossen?

R: Ja, dan moet je bij euh bij euh hoe noem je dat? Het beheer bij//ja//Jelle en euh Erik euh vragen of ze dat kunnen doen. Euh maar dat is, ja, meer werk, doorlooptijd. Je kan het zelf doen en gewoon dat je het zelf kan doen dat is heel fijn. Dan heb je daar gewoon controle over. En als het niet meer nodig is dan ruim je het op //ja//. Ja, dus dat is heel mooi mechanisme. Euh en daarnaast kan je ook natuurlijk meer euh notes doen. Als je heel veel losse processen die gewoon eigenlijk, ja, synchroon kunnen draaien //ja//, ja, dat kan prima als je weet hoe dat

werkt in Jenkins kan je ook los eigenlijk en dan kun je binnen een paar minuten kun je echt een volledig geteste of volledig deployede versie hebben eigenlijk //ja//.

I: Ok. Euhm ja, dus voor de rest heb je als probleem genoemd dan die disk euh bij een project dat verloopt //ja//. Euhm maar verder geen dingen van stabiliteit, performance…

R: Nee, dat is heel prima. Performance is euh prima. Euh//beschikbaarheid? // beschikbaarheid ook prima. Bij… een jaar geleden hadden we eerst (26:45) offline was hadden we een of twee keer, maar dat is sinds de Zomer dat ik euh //ja// nergens last van heb gehad. Prima.

I: Alright. Euhm… oh ja, zou je jezelf kwalificeren als expert in gebruik van de tooling die je…

R: Wat is expert? //ja// expert dan weet je dat je niet alles weet //ja//. Dus ja, ja goed. Ik kan mijn weg erin vinden //ja//. Ik weet wat ik wil euh maar dat is dan ook meer gewoon op basis van ervaring //nou// dus ik kan dingen gedaan krijgen op een mooie manier geen idee, maar het werkt //ja//. Dus ja, wat is expert, ja.

I: Maar, een boven gemiddeld niveau//ja, ja// zou ik zeggen. Je weet hoe je, hoe je, hoe je de toolen zeg maar voor je kunt laten werken en hoe je//ja, ja// dat aan elkaar kunt knopen en dat soort zaken //ja, dat klopt, ja//. Euhm ja, even andere dingen en euh (27:36) wordt dat bijvoorbeeld ook Docker desktop euh dat je niet direct toegang hebt tot de VM's of de hosts waarop je applicatie of je Docker containers draaien. Is dat een probleem?

R: Euhm… even denken. Het project waar ik nu zit is dat geen probleem. Euh sommige images die wij gebruiken die hebben ook wel SSH aanstaan //ja// dus dan kan je hem aanzetten als //ja// je dat zou willen. Euh de vorig of het passpoortsignalering project hadden we wel wat files die we genereren, in de Docker images //euh euhm//. En daar gebruikten we de, in de eerste instantie de- jou SSH plugin //euh euhm// die daar bijkomt. Euhm hadden we in testics een keyword geschreven, die het op een rare manier euh overhaalde//euhm//. En later had jij volgens mij euh (28:25) browser plugin euh toegevoegd//euh euhm// en aan de hand daarvan downloaden de bestanden die erop zitten, maar daar, voor de rest hadden we niet echt SSH nodig. We hadden wel (28:35) geënabled nodig. Ja, daar konden we alles mee doen eigenlijk. Euh vrij goed gegaan.

I: Maar niet bijvoorbeeld dat euh een situatie waar je applicatie totaal niet opstart?

R: Daar ben ik twee keer voor naar het IQ-team gegaan //ja//. Maar dat was onze fout en dan zag je dat daarin dat je log er(28:50) een beetje in ontbrak //ja//. Je hebt normaal dat je, je Docker open kan doen als je het lokaal draait. Zie je dat er wat er aan de hand is, dat mis je. Euhm maar vaak… volgens mij slaat het ook, die login toegevoegd aan de deskport, soort van. Maar daar konden we ook wel wat mee, maar ik kan mij herinneren dat ik twee keer neer het IQ team was gelopen om te kijken van he, waarom start de, de image niet op //ok, ja//. Euhm maar goed.

I: Maar, staat er ook iets in wat je eerst lokaal test? //euh// of lokaal eerst eens gedraaid hebt?

R: Dus, later realiseerde we ook dat je directies die je natuurlijk, die je image gewoon kon toolen en dan euh kan je het zelf starten. Euh goed omdat jij dat, helemaal wist niets van aan //ja//. Maar goed, ja dat is, ja. Dat moet je een keer weten en dan euh dan is het geen probleem //nee//. Kan je alles mee //ja//. En dan kan je daar ook lokaal bij zelfs euh bij alles. Als je, als je hem poolt (29:40) //ja// wilt euh.

I: Ja, dus dan kan je ook lokaal eerst euh als er problemen zijn, zeg maar als het lokaal (29:45) soort van en dan euh…

R: Dan weet je wat er aan de hand is//ja// of ja. Ja, je kan het oplossen het probleem wat er, wat we mogelijk achten.

I: Maar zou je zeggen dat euhm- en stel je bent er inderdaad een dag niet. Alle, wat je net ook

al zei. Zou de andere dat ook kunnen of is dat iets wat jij echt specifiek doet, in je //euh// werk deed.

R: We waren al twee euh ik en Marcel (30:08) //ja// die dat soort problemen allemaal oplost eigenlijk //ja//. Die had daar kennis van.

I: Jullie waren de enige twee ontwikkelaars dan?

R: Ja, de backend ontwikkelaars//de backend ontwikkelaars//. Ja, dus die in de front-end werkt ja die, die weet wel hoe Docker werkt, maar die is, ja, die is er niet zo handig in. Je ziet ook het verschil daartussen in een ontwikkelaar die met Linux om kan gaan en iemand die dat niet kan. Dat zie je, daar is gewoon een heel duidelijk verschil in te zien. Euhm dat is al minder vanzelfsprekend //ja//. Euhm, maar goed, ja. Het project wat ik nu heb, het vastgoedproject, daar hebben ze wel problemen gehad als ik er niet was. Maar //ja// hebben we uiteindelijk allemaal zelf kunnen oplossen. En de login en in de scripts, ik heb ze heel duidelijk uitgelegd waar wat gebeurd en dan kunnen ze- konden ze vrij makkelijk terug passeren met wat (30:54) wat er aan de hand was //ok//. Alleen een aanpassing in de (30:58) dat is, dat gaan ze niet kunnen doen //nee//. Dat lijkt te ver van hun //ja//dus euh…

I: Denk je niet dat je ook euh bijvoorbeeld meer bij dat euh meer als, bijvoorbeeld support fungeert binnen je team. Misschien voor testers als er iets aan de hand is of…

R: Ja, ja. de rol dat ik nu heb is meer… ik heb kennis van databases, ik weet hoe ETL processen werken, maar ik ben geen expert in het tool die gebruikt wordt //ja//. Ik ben ook geen expert in het testen dus ik kan wel zeggen dat ik een soort van facilitator ben. De olie die in het team machine eigenlijk… //ja//. Ik zorg dat alles draait, ja.

I: En waarom vindt je dat?

R: Euhm… ja ik vind het leuk. Euh, maar je moet wel, wat ik wel geleerd heb is dat je het niet te mooi moet maken//euh euhm//. Ik heb dan maar wat scriptjes in het Git gezet en dan kunnen ze- we hebben nu een, die FRW-server, daar heb ik verschillende repost tools waar ze hun files uploaden en die je dan los kan testen. Want onze testers, die testen eerst wat en dan kwam de (31:55) of de CE-beeld langs en die overschreef zijn files waar die aan, aan het werken was. Dus dan had ik een aparte repost tool dat hij deze files apart dan kan uploaden en dan kan die dus die versie dan die heeft, kan die, die tegen zijn eigen data testen. Euh om mij (32:11) maar hij vergeet dat die daarin zit//euh euhm// en dat (32:15). Maar die zit dus in de verkeerde repost tool zit die te werken//euh euhm// dus hoe meer keuzes je bied aan gebruikers die minder technisch zijn, hoe euh… ja, dan daar moet je echt op letten van, als ze een fout hebben dan…wat doe je, wat doe je. Heel stappenplan, vragenlijst eigenlijk van wat ben je aan het doen. Dus dat is het, de keerzijde van, je kan het heel mooi maken, maar je moet het ook niet te mooi maken want dan werkt het tegen je eigenlijk. Dus euh…

I: Ja, en in principe is wat we wel bij andere projecten gehoord hebben is dat (32:44) van ik ben eigenlijk meer supporter aan het doen voor de testers, technisch support, zeg maar. Dingen die niet werken bij de test en dat het uiteindelijk allemaal technische problemen dan dat ik aan een nieuwe feature aan het ontwikkelen ben voor de applicatie. En dat heb je inderdaad, dat ken je niet zo.

R: Nee, neen. Ik ben wel in de eerste maand toen was er echt helemaal geen automatische test. Dus, alles was handmatig en wat ik gedaan heb is Testics werkend gekregen met een paar keywords die ik euh zelf toegevoegd heb om zelf zo een file, euh zo'n ETL-proces te starten op de server. Wat bestanden van die server te downloaden die gegenereerd worden en dan euh ja, de database creëren. Als je dit als invoer mag je dit dus uitvoeren. Dus dat heb ik opgezet voor ze en dat hebben ze zelf ingevuld //ok// en nu is het puur stukje bij beetje kwaliteit verhogen. Dus dat het robuuster is als er een error optreedt dat je een duidelijk melding krijgt van, dit is er aan de hand//euh euhm//. Maar dat is meer de, de in (33:40) geving die wij hebben. Die ik euh zeg maar euh in Testics zo aan elkaar knoop dat voor onze situatie de juiste vermeldingen

komen.

I: Juist, ok. Euhm even kijken dan zijn we er bijna euhm… ja wil jij iets over de gebruik van de kwaliteit rapportage of //ja// gebruik, gebruiken jullie dat zelf, zeg maar, binnen in het team om de kwaliteit in de gaten te houden en te verbeteren of?

R: Ja, Pim had het opgezet, Pim Marsman //ja//. En euh die is er nu niet meer maar nu, wij hebben in onze euh afsprakenlijst staan dat we elke week, niet maar een keer moeten kijken en doorlopen of dan gaan we normen verliezen, opgesteld. denken bij een release [Hoest] versie beheer voor euh release documentatie euhm test cases die worden uitgevoerd euhm dus ja, dat- we zien direct als de, de automatische test als dat niet goed gaat dan komt dat fdirect in terug. Dus daar, daar zijn we wel actief op.

I: En leveren jullie die rapportage ook op aan de klant?

R: Euhm//of niet?// neen dat [Hoest] dat deskport dat is puur voor onszelf om euh //ja//.

I: En de tests, met de tests of- dus de test rapportage is wel voor de klanten neem ik aan.

R: Ja, die worden meegestuurd vanuit (35:08) wordt dat gegenereerd en meegegeven //ja// in de release. Bij de vierde releasemanager komt dat euh komt dat naar voren //ja//.

I: Krijg je daar weleens feedback op vanuit de klant of…?

R: Euh de klant loopt nu een beetje achter de feiten aan//ja// euh maar de packets wordt wel bekeken, klopt het allemaal. We krijgen er feedback op maar ze doen er niets mee. Ze hebben het niet geïnstalleerd euh maar het heeft ook meer politieke achtergrond. Zij mogen die applicatie opeens niet meer installeren omdat het op de grijze lijst stond//euh euhm//. En nu is het op de rode lijst gekomen dus ze doen er nu niets mee. En wij hebben nu wel binnen Docker, dus euhm aparte omgeving opgetuigd waar ze hun actie plaats test kunnen doen. Dus, we hebben nu een klop in Jenkins en daar wordt voor, dan kun je in de downloadlijstje print selecteren en wordt die omgeving klaargezet //ok//. Dus dan kunnen ze los. Dat is altijd wel…

I: Maar dat, dan deployen die dan hier?

R: Ja, dat is het eigen deskport [Hoest].

I: Ok, dus daar doet de klant zijn acceptatie testen.

R: Ja, ja.

I: Ok, bijzonder.

R: Ja, dat euh zo werkt dat…

I: Euhm … Oh ja en iets over de test cases. Hoe worden die- vastleggen in (36:25)? //euh euhm// Daar weet je ook wel, daar weet je hoe dat werkt?

R: Hier worden de logische test gedefinieerd //ja// en euh sommige zijn handmatig en later is het wel het idee dat ze allemaal geautomatiseerd doen. Daar zij we nu wel vrij ver mee. Alleen we lopen tegen wat dingetjes aan wat euh… onze testers werken op Windows //ja// en dat is toch een heel ander gedrag dan op Linux omgeving. En daar denk ik aan (36:55) bijvoorbeeld//euhm euhm// dus daar dat ze wel tegen problemen aanlopen dus het is… ik probeer hem ook te pushen, ga naar Linux in het nieuwjaar…

I: Maar dat is met de tooling dan? Bedoel je Testics…

R: Ja, Testics. Ja, je moet dingen (37:05) vallen gedefinieerd moet in klein letters en de (37:10) alleen met hoofdletters //ja//. Leuk dat het op zijn machine draait. Ik heb allemaal groen, maar euh waarom gaat dat niet in de, //ja// in de beurtrapportage goed? Dus ik denk dat is wel,

dan moet je echt kennis weten van, wat is nou het verschil tussen Linux en Windows? //ja// worden daar wijzigingen op euh het idee is dat we wel euhm… oh, dat is ook een probleem waar ik tegenaan liep. Euh mailserver, als jij een mailtje wilt krijgen van euh ik ben kapot. Je wilt bijvoorbeeld euh, RT is omgevallen//euh euhm//. Euh niemand komt op het Jenkins desktop behalve ikzelf om te kijken of de RT nog steeds draait. En daar ook een mailtje voor doen. Ik weet hoe het moet. Ik heb het in andere projecten voor elkaar gekregen, maar op dit moment hebben we hem niet werkende gekregen //ok//. Even niet echt prioriteit, maar dat is ook een van de problemen waar ik tegenaan liep.

I: Nou, dat heeft dan gewoon te maken met, inderdaad die configuratie van dat soort standaard tooling. Dat zou gewoon//ja, ja// meer uniform geregeld kunnen worden. Want nu moet jij dat elke keer opnieuw doen eigenlijk.

R: Ja, dat klopt. Dus ja, je moet de SSB-server en (38:14) gaan halen en Jenkins moet je, ja goed. Dat werkt allemaal wat lastiger //ja//. Dus dat is euh het is handig als je dat af kan kijken.

I: Ja, ok. Euhm dan de laatste, de laatste punt eigenlijk euhm… hoe kijk je dan tegen de ondersteuning vanuit euh de organisatie? Dus, in dit geval vanuit IQ. Euh bij bijvoorbeeld euh het doel van je projecten in een soort van (38:40) situatie. Is er voldoende ondersteuning door de ondersteunende teams? Zoals die technische beheer IQ-team, TPA's…//euhm// misschien meer ondersteuning in het proces of?

R: Euhm ja, wat ik…de ondersteuning die ik heb als ik een probleem heb en ik kom naar het IQ-team of beheer//euh euhm//. Ik word altijd geholpen, prima. We komen er altijd uit. Euhm maar ja, kijk standaarddingen. Je stelt misschien meer voor, voor het IQ-team of uiteindelijk weer zelf wat je op je vragenlijstje hebt voor, ik heb dit probleem dan moet je dit doen. Maar het is meer om de teams waar je naartoe gaat dan te ontlasten //ja//. Goed, het is misschien meer vooruitlopend op de andere kant//euh euhm//. Euhm kijk als het vaak zo'n probleem is dan euh ja soms zijn mensen druk of bezig dan kun je ze niet bereiken. Dan kan je weer verder, maar goed. Als je je werk niet kan doen omdat je, omdat iets kapot is dan… of je weet het teneinde van, of je doet het niet goed, maar ik zeggen is het goed dat er daar een andere insteek in euhm…

I: Ja, de reden dat ik dat vraag is omdat euh wij hebben… ik heb natuurlijk zelf ook een aantal projecten gedaan bij IQ en toen tegen dezelfde problemen heb oftewel (39:48) deden we het beheer door//euh euhm// technisch beheer team of ja, gedeelde Jenkins en dat soort dingen, ze lopen tegen dat soort problemen aan dus wij hadden toen bedacht van, ja, wij moeten teams meer vrijheden geven. Want wij hadden eigenlijk, als ik naar mijzelf kijk als ontwikkelaar, wij weten prima hoe we de ontwikkelstraat willen inrichten en hoe we dat euh willen configureren dus dat euh die vrijheid zouden wij moeten hebben. Alleen wij hebben ons natuurlijk onvoldoende gerealiseerd dat, dat het misschien niet voor iedereen zo is en dat sommige teams of euh ontwikkelaars daar misschien meer ondersteuning bij nodig hebben //ja// en dat hebben we ons pas, denk ik, te laat gerealiseerd //ja//. Want sommige projecten zeggen we willen die vrijheid eigenlijk helemaal niet. Doe maar gewoon standaard. Dat klopt, ja, als dat bij jou past, ja, prima //ja// voor hun. Maar dat is ook denk ik afhankelijk van wat voor mensen je in je projecten hebt //ja//. Maar ik vind het wel altijd euh fijn om de mogelijkheid te hebben als het niet standaard werkt, niet goed is dan (40:39) en dan kan je toch doen wat je wilt. Euh ik heb al eens mee gemaakt bij een project niet binnen IQ, maar dat je dan net voor een deadline dat dingen niet werken en dan als je daar kan ingrijpen dan //ja// is dat wel fijn. Dat kan ook stress geven als je, als je die mogelijkheid hebt.

I: Ja, dus gewoon eigenlijk standaardiseren. Je zegt eigenlijk standaardiseren op de, ja hoe moet je dat noemen, op de generieken dingen, zeg maar. Dan is het nuttig, maar wel de vrijheid- De hele mogelijkheid laten om daar vrij// ja//.

R: Bijvoorbeeld wat je kan doen is die euh is nu in Jenkins (41:38)//euh euhm// daar zou je bij wijze van spreken ook voor een euh ik zou, ik heb een (41:15) applicatie met een mail (41:19) mechanisme erin. Dat je dat gewoon kan, kijk dit werkt. Als je het standaard wil kan je dit copy

pasten of euh overhalen. Gebruik deze Grit template, klaar. Voeg een applicatie toe en dan zou het moeten werken //ja//. Dat zou mooi zijn, maar dat is dan een start en dan kan je zelf laten beslissen euhm ik ga hiervan afwijken //ja// of juist niet, wat je wilt.

I: Maar de voordelen zitten dan vooral dus bij het starten//ja, ja//// van je project//dat euh, ja//. Je hebt het ook van verschillende eh, je hebt verschillende projecten bij IQ gedaan//euh//. ja, toch? /dat klopt//.

R: Ja, wat je ziet, wat ik zeker bij euh van passpoortsignalering naar euh slachtofferportaal ging. Het eerste wat we deden was twee weken lang copy pasten wat we daar bij die andere hadden en natuurlijk ook hier hebben. En, ja, toen kwam het idee van ja, waarom is het niet standaard (42:10) //ja// zoveel aan eigenlijk, kant-en-klaar en volgt een template, deploye //ja, ja//. Uiteindelijk komen we uit, maar als je dat euh in een jaar voor tien projecten doet kan je wel wat sparen//ja. Ok//. Dat euh…

I: Volgens mij waren dat euh dat waren mijn vragen zoal, ik denk het, ja. //prima// ok, je hebt zelf ook niets meer, geen problemen of //euh// dingen over hoe we, zeg maar…

R: Ik vind het prima werk (42:39) ben. Als je dit vergelijkt met andere bedrijven //ja//. Ik heb hier vorig jaar bij een bank gewerkt en daar gebruikten ze (42:45) dat soort dingen allemaal in elkaar. Euh dit is toch wel een heel stuk flexibeler. Euh je ziet…

I: In welke zin precies?

R: Euhm je ziet wat er gebeurd //ja// en daar hadden ze, bij die bank hadden ze dus, had je een (43:00) supportteam wat (43:04) niet beschikbaar was als er iets fout ging en dan moest je bellen als er wat fout ging en dan werd er gekeken en dan, ja het kopt. Ik zie precies hetzelfde wat jij ziet aan de hand van de melding die je zag, en dan werd er gekeken, ok, ik kan het oplossen of het werd doorgezet naar een tweede lijn, een derde lijn, ja. Euh en als je dan releases had, had je weleens de pech dat je urenlang bezig was.

I: Maar waarom kon je dat zelf niet oplossen als je daar direct toegang had tot..

R: Dat is een rechte, ja dat is een bank [Hoest] dat is euh allemaal afgeschermd. Euhm ja, dat is allemaal lastiger en je kan met omwegjes kan je wel dingen doen, maar je wordt daar constant gemonitord. Je hebt (43:40) daar met services als je daar met de hand inlogt dan zien ze dat en dan krijg je andere problemen //ok//. Dus het, euhm nou dat is een heel mooi applicatie landschap om te deployen //ja// en als ook geprobeerd te stabiliseren, maar het werkte toch niet helemaal lekker. En hier euh heb je de mogelijkheid om hetzelfde te doen, maar als je het afneemt van iemand anders is het prima landschap. Het loopt vooruit op het concurrentie eigenlijk. Dus…

I: Nou //prima//. Dat is euh goed om te horen.

R: Dus euh…

I: Alright, mooi.

Team 4

Property	Value
Date	19-12-2016 14:00
Duration	1:10
Present	***I:**** interviewer
	S: developer 1
	J: tester 1
Team members	4

Property	Value
Team size	Small
Project size	Small

Transcript

This paragraph contains the annotated transcript of the interview. One developer and one tester took part. Since the interview was in Dutch the transcript is also in Dutch. Annotations are in English.

S: developer J: tester

I: oke, ik heb een vragen en en die gaan we gewoon even af en dan kijken we hoe relevant het is voor dit project.

S: klinkt goed toch

I: Ja, het eerste is om inzicht te krijgen in wat voor soort project het is. Waar werken jullie aan, wat is zegmaar de schaal. Hoeveel applicaties? hoe ziet het project er uit. In termen van aantallen applicaties. Waar werken jullie aan?

s: Het is een beheerproject. Het is twee jaar geleden gestart binnen de softwareorganisatie. Daarvoor lag het onderhoud bij grote ontwikkelaar 1. En daarvoor bij nog een andere club, geloof ik. Uiteindelijk is het bij de softwareorganisatie gekomen. In principe voor doorontwikkeling en beheer.

I: Ja

S: Voor alle applicaties die de inspectiedienst 1 in gebruik heeft. Het zijn eigenlijk twee grote applicaties. Waarvan er eentje een ouderwetse winforms applicatie is. Die andere is een webapplicatie voor de boeteafhandeling. Wat eigenlijk ook wat verouderde code is, maar wel web. En er hangen nog wat randapplicaties omheen. Pak en beet is het zegmaar een half miljoen regels code, in die orde van grote.

I: En het is allemaal dot net?

s: Allemaal dot net, ja. Het is allemaal C#, wel verschillende technieken daarbinnen dan. En SQL server als database.

I: en als je kijkt naar, ik weet niet of dat bestaat hoor, een standaad dot net project. Wijkt jullie project daarvan af? Of volgt het gewoon de geldende richtlijnen voor dot net projecten?

S: <> Nou, het is niet de beste code die we in beheer hebben. Historisch gezien ook. Ehm, dat is denk ik niet zo het probleem. Het is vooral het probleem dat het zoveel verschillende applicaties zijn die bij verschillende partijen geleeft hebben. Er zijn gewoon heel veel verschillende smaakjes gebruikt. Een manier om een database te benaderen dat is zes keer opnieuw uitgedacht. Dat maakt het lastig. Voor de rest is het vrij standaard dot net, geen exotische dingen.

I: Ok

S: Het is ook niet zo'n hele spannende applicatie eigenlijk.

I: Nee, je zegt het is dan doorontwikkeling en tijdelijk beheer? Of? Wat zij je precies daarvoor?

S: Het is gewoon, <>, ja beheer. Er komen wensen binnen…

I: Wie doet de hosting?

S: hostingpartij 1

I: Ok.

J: Die ken je?

I: Nee die ken ik niet, maar dat is…

I: hostingpartij 1 is volgens mij, ik weet 't niet zeker hoor, een van de grootste overheidspartijen.

J: Het is in elk geval geen interne afdeling zegmaar. Wij doen zelf niet de hosting.

I: Hoe gaan jullie om met zaken als continuous delivery? Doen jullie daar aan? En dan misschien niet tot aan productie maar misschien tot aan een test of acceptatieomgeving.

S: Wij hebben bij de softwareorganisatie intern een omgeving. Die soort van representatief is met de productieomgeving van de project klant. Bij hostingpartij 1 dan. Daar kunnen wij, naja, het is nog niet helemaal continuous delivery. Met 1 druk op knop kunnen wij het hele landschap, zo'n tien applicaties, met een druk op de knop uitrollen. Naar elke gewenste versie die wij willen.

J: Dan tevens de ART's er achteraan laten draaien. Dus in zoverre zou je het wel continuous integration kunnen noemen. Denk ik.

S: Ja, maar er is altijd een handmatige actie nodig. We hebben geen nightly build ofzo.

J: Nee, en ook niet dat automatisch als jij nieuwe code incheckt dat dan het hele riedeltje gaat lopen.

S: Ik moet zeggen van nou, ik wil hier een nieuwe versie van maken. Dan krijgt het een nieuw nummertje enzo.

J: Ik als tester moet dan zeggen, nu deze versie deployen. En dan…

I: Kun je zelf een nieuwe versie deployen?

J: Ja.

I: Dus het is gewoon echt inchecken, dan wordt het automatisch gebouwd, neem ik aan.

S: >

I: En dan is het dus met een druk op de knop deployen en het uitvoeren van de testen.

J: Ik druk op de knop execute, en dan start de deploy.

S: Naar de klant kant zegmaar, dat moeten we een paar handmatige stapjes voor doen. Maar hetzelfde ding wat wij automatisch deployen dat leveren we op dan. Met de configuratie al voorgemaakt zegmaar, voor hun omgevingen.

I: En wat is dat dan voor . Wat lever je precies op?

S: In principe alle binaries gewoon. Applicaties. Maar daarbinnen zit een mapje met alle configuraties voor de verschillende omgevingen. Die staan dan helemaal goed.

J: Hun machine namen..

S: Dus zij moeten het even kopieren, overschrijven en dan werkt het. Zij hoeven daar niets meer aan te configureren als het goed is.

I: Ok

J: Maar het deployen daadwerkelijk deployen of neerzetten van die applicaties op hun omgeving is wel een handmatige actie aan hun kant. Op dit moment nog.

S: Ja, dus stel die tien a twintig applicties moeten zij wel een voor een copy pasten, ja. En databases restoren en scripts uitvoeren moeten zij allemaal als losste stappen doen.

J: En voor een van de volgende sprints, , story op de planning staan in de nabije toekomst om ook dat stapje bij hun geautomatiseerd te kunnen gaan doen.

I: Ja, want levert dat soms problemen op? Of gaat het eigenlijk altijd wel goed?

s: Sinds wij die configuratie al helemaal klaarmaken voor hun, gaat het meestal wel goed.

I: Meestal.

s: Meestal. Nog niet 100%.

J: Het is nu voornamelijk een issue dat het ze gewoon nog veel tijd kost om zo'n installatie te doen. Ik denk dat ze er zo een ochtend mee bezig zijn.

S: Ja, misschien wel ja.

J: een dagdeel ofzo zegmaar.

S: In principe werkt het bij ons, hun omgeving is bijna identiek. Dus die scripts die wij gebruiken om het uit te rollen, dat kunnen zij volgens mij ok gewoon gebruiken. Binnen nu en volgende sprint denk ik, in principe.

J: In principe staat 'ie voor de volgende sprint. In de nabije toekomst gaan wij die scripts gereed maken zodat zij ze ook kunnen gebruiken.

I: Ja, en hiervoor. Toen jullie nog niet de configuratie volgens hun systemen opleverde, waren er toen meer problemen?

S: Ja, voor ons was het ook veel meer werk. Elke configuratiewijziging moesten wij documenteren dan. En die moesten zij dan weer gaan toepassen. Documentatie lezen, van oh dan moet ik zeker dit doen. Dat ging vaak fout. Nu documenteren wij het niet meer, wij passen het gewoon toe. We vragen aan hun wat de waarde bij hun moet zijn en passen het toe in onze codebase. En klaar. zij hoeven er verder niet meer over na te denken.

I: En wat voor configuratiewijzigingen zijn dat dan?

S: Nieuwe server endpoint ofzo, database endpoints. Dingen die er bij komen, koppelingen met andere databases. Dat soort dingen. Of instellingen voor nieuwe rechten.

08:15

I: Ja ik heb verder geen idee hoe je een dot net applicatie deployed. In dit geval zijn het gewoon executables die je

S: Ja in principe alles staat al ingericht natuurlijk, dus voor de bestaande applicaties kun je gewoon de binaries vervangen. Voor de website moet je voor IIS de webserver wel even aangeven van ik wil een website hebben op dit adres en hier staan de binaries zegmaar. En deze authenticatiemethode zijn nodig. In principe is dat eenmalig nodig. En dan kun je daana gewoon die bestanden vervangen als je dat wil.

I: Want als je een nieuwe lege windows server zou krijgen, wat moet er dan nog geinstalleerd worden? Want je zij ik kan hier ook intern met een druk op de knop deployen, maar zou dat ook kunnen naar een verse windows server installatie?

s: Onze scripts wel, nouja vers. Als er maar IIS op staat.

I: Oke, dat is de enige afhankelijkheid? en de scripts doen de rest?

S: Ja, de scripts doen de rest die configureren ook IIS bij ons.

I: En wat voor scripts zijn dat dan? gewoon bash?

S: Nee powershell.

I: Powershell, ja natuurlijk. Allright , ja met hoeveel ontwikkelaars zijn jullie in het team?

s: Drie.

I: Een tester of twee testers?

J: Een tester, drie (ontwikkelaars) en een tester. Ja

S: en in principe informatieanalyse (fluisterend: voor zover we dat doen). Dat ligt bij de klant.

I: Oke, ja de vragen zijn wel erg toegespitst die van onze infrastructuur gebruikmaken. Want ja, we weten dat daar wel problemen mee zijn. dus een van de vragen is welke problemen kom je tegen in eht dagelijhks werk. maar ik weet niet of dat bij jullie aan de hand is. Is de tooling eigenlijk goed op elkaar afgestemd?

S: Van het iq-team bedoel je?

I: Nee, wat jullie gebruiken aan tooling zegmaar. Om de ontwikkelstraat in te richten, dagelijks je builds te kunnen doen, automatische builds te kunnen doen.

S: Ja dat werkt opzich wel.

I: Wat gebruiken jullie daarvoor?

S: TFS.

I: Gewoon TFS?

S: En maatwerk.

J: Dus TFS en dan zegmaar al die powershells die daadwerkelijk de deployment doen. Die zijn door de voorganger ontwikkelaar in elkaar gezet. Dus ja dat werkt goed.

S: Ja TFS die roept een paar van die powershells aan na zo'n build. En we hebben ook een eigen websiteje gemaakt waar J, en wij ook, die deploys mee doen.

J: Die roept ook powershells aan.

S: Ja die roept uiteindelijk ook weer powershells aan. Scant gewoon mappen naar bepaalde versies die daar staan. En dan kun je die gewoon kiezen en deployen.

I: Over hoeveel powershell code hebben we het dan? Paar honderd regels of een paar duizend?

s: Ik weet het niet precies. Vijfhonderd ofzo, ik gok maar wat hoor. Zoiets. Misschien iets meer. Maja, zoiets zegmaar. Tot duizend.

I: En het testen, de automatische testen. Waar zijn die in geschreven?

J: , ik maak gebruik van TestX protractor en TestX.

I: Nou dat is toch nog een dingetje dan

S: Nou..

J: gedeeltelijk.

S: Als je het echt wilt weten.

I: Ja ik wil het echt weten.

S: Het was niet onze keus.

I: Ok.

S: Maar het is echt gepushed.

I: Vanuit? Door de ontwikkelorganisatie?

S: Ja.

J: Ja. Toen wij begonnen, of toen ik begon in iedergeval was echt nog het standpunt van de organisatie van het moet TestX zijn want dat is nou eenmaal onze standaard.

I: Ok.

J: Inmiddels staan ze meer op het standpunt van laten we per project kijken wat werkt. Maar wij hebben best wel veel maatwerk, eigen specifieke keywords moeten ontwikkelen in Testx protractor om al onze testjes te kunnen uitvoeren. En dat heeft voor een deel gewoon te maken met dat die applicaties verouderd zijn. En nooit ontwikkeld zijn om geautomatiseerd te kunnen testen.

S: Ja het werkt met rare popups en iframes en allemaal dat soort dingetjes. Het werkt gewooon niet lekker.

J: Nee. en anderzijds denk ik ook dater nooit een goedo nderzoekje gedaan is naar is dit nou de tool om deze applicatie mee te testen. We zijn nu bezig met een stukje nieuwbouw. Dat is eigenlijk om de bestaande applicaties als Angular webapplicatie na te bouwen. Gebruiken we ook weer TestX en dan zie je dat dat veel beter aansluit. Vooral omdat protractor specifiek bedoeld is om Angular webapplicatiss mee te testen. En dan zie je dat zo'n tool wel geschikt is. Terwijl dat voor die ouwe meuk van ons gewoon niet geschikt is. En dat dat eigenlijk ook heel veel capaciteit bij onze developers gekost heeft om die keywords zo te maken dat ze met de quirks van onze applicaties kunnen omgaan.

I: Kan jij zelf keywords maken? Of dat niet?

J: Nee. Nee. nee.

I: Ok. En gebruik je dan de Excel integratie?

J: Ja, ik weet wel dat, hoe het het ook alweer. Yaml. Dat er een andere methode was.

I: Ja.

J: En toen dachten wij van nou met de hele berg aan excellen die we al hebben liggen zijn we waarschijnlijk twee jaar bezig om dat om te bouwen en om te zetten. En in gedeeltelijke oude applicaties die waarschijnlijk ooit wel uitgefaseerd gaan worden. Dat App1 hopen we niet heel veel testen meer voor te hoeven schrijven.

S: Ja, App1 is weer een andere tak van sport. Is winforms. Daar is TestX niet zo goed mee.

I: Nee, winforms is gewoon een native windows applicatie?

S: Ja, zegmaar native windows applicatie op zelfs hele oude techniek. Maar dat kan TestX niet zo goed. Of protractor.

I: Nee, dat is webbased, dus ja.

S: Daar heeft iemand, die ken je denk ik wel, B.

I: Ja ja.

S: Ja G (iq-team lid) heeft daar een beginnetje mee gemaakt. toen is B door gaan ontwikkelen. Beetje hetzelfde idee als TestX. Is ook rondom Excel gebouwd. Maar die roept dan uiteindelijk geen protractor aan maar een soort native windows UI engine die ja die test dan zegmaar de applicatie. Ja dat … Daar kunnen we het nog wel een weekje over hebben denk ik.

J:

S: Maar in principe is dat in beheer bij ons.

I: Maar is dat dan…

S: Trouwens, we gebruiken een ding van jullie. XLS2Test.

J: Ja die staat voor winart wel aan geloof ik.

S: Ja.

J: Voor het Testx is die uiteindelijk weer uit gegaan. Maar voor winart is het allemaal aan blijven staan.

S: Het is een module die zet die Excel sheets om.

I: Ja, naar Testx

15:29 s: objecten. Ja. Dus dat word nog wel gebruikt voor onze windows applicatie. Want die excel sheets kan ik ….. doen en dan krijg ik daar een object voor terug. anders moet dat native ook weer gecodeerd worden.

I: En jullie draaien dat zelf ergens. Ja precies. dat is volgen mij alweer oud.

s: ja het is oud.

J: het zal mij niet verbazen als wij nog de enige gebruiker daarvan zijn.

I: ja dat zou kunnen ja.

s: iemand heeft het een keertje uitgezet bij jullie. Ja ik weet niet of het jullie team is. Maar toen werkte het niet meer bij ons.

I: Oh, het draaide eerst centraal?

S: nee, het draaid denk ik in ons Docker

I: Oh ok, handig om te weten. Maar opzich, bevalt TestX wel. Het moet alleen wel aansluiten bij de applicatie?

J: Wat mij betreft voor nu voor zo'n moderne applicatie daar sluit de tool gewoon goed aan op de programmatuur en dan heb ik weinig klagen.

s: Ja, als je het aan mij vraagt ben ik niet zo'n fan van Excel. Om dit soort dingen in Excel te doen.

J: Het is misschien nog weer een volgende stap. ja en of Yaml daar de oplossing voor is dat denk ik eigenlijk niet.

s: Ik vind ook niet dat wij onze 400 testen moeten gaan herschrijven.

I: zou je daar misschien hulp bij nodig hebben. Niet dat wij dat met de hand gaan doen, maar meer in de zin van die testen automatisch migreren naar een ander formaat. de testen blijven hetzelfde, het is meer het formaat.

s: Ja klopt.

I: Bij Yaml zijn de keywords en alles blijft hetzelfde. Het is alleen een ander formaat.

J: Ja ik vraag me alleen af wat de winst is als we dat nu zouden doen. In de zin van we hebben met heel veel effort uiteindelijk voor onze oude webapplicatie behoorlijk stabiel gekregen. Er wordt niet heel veel meer aan doorontwikkeld.

S: Nee, dat is waar.

J: Het is nu meer een beetje van… Het heeft heel veel effort gekost en het heeft nu een soort van volwassenheidsniveau bereikt. Die testen draaien nu al zo lang dat heel veel meer testen erbij schrijven willen we ook niet. Dan ben je zo een dag ART's aan het stampen. Het is nu iets van vier uur als we voor alle applicaties de geautomatiseerde testen aftrappen. Dus we zitten ook wel redelijk aan de limieten dat je niet meer testen erbij wil. En, dus ja wat win je dan zegmaar door heel veel effort te steken in het omzetten naar Yaml.

I: Nee, en je kunt ook mixen en matchen. Je kun Excel en Yaml naast elkaar gebruiken.

J: Het grote voordeel wat ik begreep van N is dat je geen merge conflicten meer hebt.

I: Ja.

J: en omdat wij toch eigenlijk voor die testen niet meer dan drie man tegelijk aan dezelfde test aan het ontwikkelen zijn.

S: Naja, we coordineren dat heel goed. Toch? Als jij ART's aan het schrijven bent dan ga ik daar niet in rommelen.

J: Ja, ook dat, ja.

I: Ja, oke. Maar er is dus extra coordinatie binnen het team nodig.

J: Maar we hebben ook wel periodes gehad toen we echt nog veel testen bij maakte. Dat het niet raar was als er drie man tegelijkertijd aan het ARTen waren. Maar dat hebben we ook al best een lange tijd niet meer. Toch?

S: Nee, nee.

I: Ok.

J: En ik denk dat het voor WinART zal het sowieso niet kunnen om om te gaan naar Yaml.

S: Nou, maar dan moeten we dat allemaal zelf gaan onderhouden.

I: Ja, ik weet niet hoe dat precies opgezet is destijds door B.

I: , je zei net dat de doorlooptijd vier uur is, als je alle testen voor alle applicaties draait.

J: Ja, ja.

I: Doen jullie dat regelmatig?

S: Ja, officieel. Eigenlijk willen we nog steeds een nightly run doen. , maar onze kwaliteitsmanager heeft volgens mij de max staan op drie dagen. Dus als we dan drie dagen niet gedraaid hebben

dan gaat het systeem piepen. Dus ja, ik weet niet of we dat doen elke drie dagen?

J: Nee, de praktijk is een paar keer per sprint.

S: Een keer per week minimaal.

J: Ja zoiets. Behalve dan op het moment dat er specifiek voor een applicatie user stories opgelost zijn dat ik voor die applicatie los dan de hele ART draai.

S: Maar niet alles.

J: Niet alles van het hele landschap. Nee.

I: Ok. . Ja hoe staat het met jullie met de beschikbaarheid van de hele ontwikkelomgeving?

s: die is 95% up ofzo. Dat is gewoon goed.

I: Dat is ook iets dat gewoon hier wordt gehost door ITB?

s: Ja.

J: Ja is dat nog ITB officieel? Volgens mij is dat door het weggaan van L ook meer bij het team komen te liggen.

20:31

S: Ja, de infra. Dus de servers, dat ligt bij ITB. Alleen de software die daar op staat in principe, buiten een kale windows. Dat is wordt ook door ons onderhouden.

J: Het was L die dat stukje eerst deed.

S: Bij zijn vertrek is dat binnen het team komen te liggen.

I: En hoe ervaren jullie dat? Hoe gaat dat?

S: Mja, omdat het draait hebben we er meestal niet zoveel problemen mee. Maar. Mocht er iets bijkomen. Een Exchange server ofzo.

J: .

S: Dan, . Waar wij geen ervaring mee hebben. We willen gewoon iemand binnen de ontwikkelorganisatie hebben die dat voor ons doet. Dat is gewoon niet onze expertise. Die expertise is op dit moment ook niet binnen de ontwikkelorganisatie. Het loopt nog niet zo snel, loopt niet zo soepel.

J: Nee maar volgens mij was het punt ook een beetje dat er te weinig Windows omgevingen in beheer zijn om er echt dedicated iemand op te zetten. Voor L was gewoon te weinig werk om… En ja, dat snap ik wel. Je kunt ook niet voor vier uur in de week iemand neerzetten. Misschien nog wel minder. Dus het loont waarschijnlijk niet om daar iemand op te zetten.

I: Nee

J: Maar als er dan eens wat moet gebeuren dan is dat wel een uitdaging.

S: Natuurlijk kunnen we het wel oplossen binnen het team hoor. De kennis is er wel, maar..

J: Ten koste van de velocity.

S: Ja ten koste van de velocity van de ontwikkeling in de sprint.

I: En dat is ook handmatig werk ook? Het onderhouden van die omgevingen?

S: Ja.

J: Ja het zijn gewoon statische omgevingen heh.

I: Over hoeveel virtuele machines hebben we het dan?

S: volgens mij hebben we er vier denk ik, per omgeving. Vier hebben we er nodig om het hele landschap te hosten.

I: Ja

S: En daar hebben we vijf varianten van. En die vijf setjes zegmaar. Vijf keer vier servers. . Op die vijf setjes kunnen we twee instanties draaien. Gewoon op dezelfde server maar er naast. Die zijn wel allemaal statisch.

I: Oke

S: We hebben zegmaar een dev omgeving. Die telt vier servers. Daar ontwikkelen we tegen. Daar kunnen we twee verschillende versies op zetten. En dat hebben we dan vijf keer. Eentje voor de ART zegmaar.

J: Eentje voor test.

S: Eentje om productiebugs te testen.

I: Dus als je deployed zeg maar, met je deploy knop, dan moet je ook kiezen waar?

S: Ja. Maar eentje is nu dedicated in gebruik door performancetesten.

J: Ja.

S: En eentje voor de art-omgeving.

I: Moet je daar wel eens over afstemmen binnen het team? Van, ik ga nu dit deployen daar.

S: Is opzich wel handig. Want als je aan het testen bent en iemand drukt op die knop dan kun je weer overnieuw beginnen.

J:

I: Gebeurt dat ook?

J: Ja, af en toe.

S: Af en toe doen we dat expres.

J:

S: Om J even scherp te houden.

I: Maar het geeft geen grote problemen bij jullie?

J: Nee, ja weet je we zitten allemaal bij elkaar op de kamer. Dus meestal is het van 'joh ik ga even naar test deployen, is dat een probleem?' Nou en als er dan niemand geageerd dan druk ik op de knop.

I: ok, ja dan heb je natuurlijk ook nog VM's voor je buildomgeving, voor TFS?

S: Die, ja. Die hebben wij ook. Maar dat ligt niet bij het team. Dat ligt wel bij ITB ook. Ik kan er niet bij ook. Die andere VM's daar ben ik gewoon administrator op. Maar die TFS server daar kunnen wij niet bij. Als ik daar iets op wil dan schiet ik tickets in. Die komen meestal bij

E terecht denk ik. Een nieuwe gebruiker toevoegen ofzo.

I: Kun je daar nog veel aan configureren? Of is dat gewoon wat het is?

S: TFS?

I: Ja, of kun je daar met plugins werken of, ja?

S: Neuh, gebruiken we allemaal niet.

I: Het is gewoon een standaard installatie?

S: Wij gebruiken TFS heel weinig eigenlijk. Alle mogelijkheden.

I: Je gebruikt het alleen als code repository? Dus git?

S: Ja er hangt Git ander inderdaad als repository. En we gebruiken hem als buildserver ja.

I: En voor de rest alle custom automation dat zijn powershell scripts?

S: Ja, TFS kun je ook helemaal inrichten dat dat net als Jira al je tickets bevat en zo.

J: Ja en zou je TFS niet in principe ook als je continuous integration oplossing kunnen gebruiken? Dat het als een soort scheduler voor verschillende jobs kan werken?

S: Jaja, dat sowieso ja.

I: Jullie gebruiken het toch ook als continuous integration oplossing? Voor het bouwen zegmaar?

S: Ja elke check-in die gaat 'ie bouwen zegmaar. En SonarQube.

I: Dat gebruiken jullie ook?

S: Ja SonarQube wel.

J: Daar zouden we in theorie ook nog stapjes kunnen maken. Dat je zegt iedere keer als je incheckt dan doet 'ie ook SonarQube.

S: Ja dat doet 'ie ook.

J: Oh dat doet 'ie ook? Ik dacht dat jullie elke keer apart die build moesten starten.

S: Nee, alleen voor App1 v6. Dat is zo ingericht, ik weet niet waarom. Ik denk omdat die build heel lang duurt ofzo.

J: Maar voor andere applicaties als jullie een build doen gaat ook automatisch Sonarqube af?

S: Ja.

J: Dat is wel netjes.

I: Doen jullie ook iets met die informatie die sonar geeft? Of komt het voornamelijk terug in de kwaliteitsrapportage en zie je het op die manier?

S: Nou, , we doen er wel iets mee hoor. Maar voor die nieuwe applicatie proberen we hem goed te houden zegmaar. die oude applicaties dat is gedoe. Er zijn gewoon zoveel violations. We proberen het wel een beetje aan pakken. Het is moeilijk onderhouden.

J: Het kwaliteitsdashboard staat min of meer zo afgesteld dat de baseline zoals we de applicatie binnen hebben gekregen dat is de geaccepteerde technische schuld. En als we aan die applicatie sleutelen dan mag het niet slechter worden.

S: Ja.

J: En als wij het kunnen uitleggen aan onze kwaliteitsmanager waarom het toch verslechterd is, of als er bijzondere omstandigheden zijn dan wordt er zo mee omgegaan van het is legacy, we weten het.

S: Maar die rules die in Sonar zitten, dat profiel, dat wordt ook regelmatig uitgebreid. En ja, dan komen er weer een paar duizend violations bij bij ons. En ja, dan moeten we de norm weer ijken en dan gaan we weer verder. Maar eigenlijk precies de kwaliteit meten, historisch en trends dat kun je eigenlijk niet meer zien.

I: Nee, snap ik. Maar kunnen jullie dan inzichtelijk krijgen wanneer je een wijziging doet of dat extra violations tot gevolg heeft.

S: Ja, dat wel. Maar ook beperkt hoor moet ik zeggen.

I: Ja, als het eenmaal zo'n grote hoop is, dan komt er een klein beetje er bovenop bij.

S: Ja, maar dat zie je wel. Dat kun je wel zien.

I: Het heeft voor jullie niet perse prioriteit?

S: Nee, eigenlijk niet.

J: Maar echt verslechteren doen we het ook niet.

S: Nee, dat niet. Maar als je kijkt naar die hele hoop, je zou dat eigenlijk gewoon terug willen brengen naar nul.

J: Ja we zouden eigenlijk gewoon stories op de backlog moeten hebben met betrekking tot technical dept. Maar dat doen we niet.

S: Ja, we hebben wel per story een taak. Kijk even naar de kwaliteit. De dingen die je geraakt hebt, kun je daar nog iets aan verbeteren? Maar het is allemaal minimaal wat mij betreft.

J: We hebben geen overkoepelend plan inderdaad om over een half jaar of een jaar op de nul te komen.

S: Ook een beetje met het idee in het achterhoofd dat we alle applicaties willen herschrijven. De huidige software is of gaat richting end of life eigenlijk. Dus de vraag is of je daar nog moet investeren. Wij hebben gekozen om het opnieuw te doen.

J: Waar we waarschijnlijk met dit tempo over een jaar of acht mee klaar zijn. Maar.

S: Sneller dan Sonarqube op nul krijgen denk ik.

J: Ja, zou het?

S: Ik denk het wel.

I: Goed, oja. Leveren jullie de kwaliteitsrapportage ook op aan de klant? of kijkt die daar helemaal niet naar?

S: Ze kijken er niet naar.

I: Maar hij wordt wel opgeleverd?

S: Nee, hij wordt niet opgeleverd maar ze kunnen er gewoon bij.

I: Maar, waarom?

S: Nou ik heb het wel eens gevraagd volgens mij maar.

J: We zouden dat wel onderdeel van de review kunnen maken. Even twee minuten met elkaar naar de kwaliteitsrapportage kijken ofzo. Ik weet niet of dat iets toevoegd aan het proces maar.

S: Ik weet niet of het iets toevoegd.

30:26

I: En de testrapportage wordt ook opgeleverd aan de klant?

J: Nou de testrapportage is eigenlijk ook niet meer dan de ART resultaten zoals die op het kwaliteitsdashboard staan. Ik schrijf niet los daarvan nog zegmaar een document of iets.

I: Nee, maar die worden ook opgeleverd en daar wordt naar gekeken?

J: Nee, daar geldt hetzelfde voor. die staan gewoon op het kwaliteitsdashboard. en dat is van buitenaf benaderdbaar voor hun. Dus als ze willen kunnen ze die openen.

S: Ja maar daar staat alleen maar een getalletje. Zoveel testen geslaagd, gefaald.

J: Ja, maar kunnen ze daar niet in doorklikken naar BIRT?

S: Nou, dat werkt niet extern denk ik.

J: Oooh, dat zou kunnen. Nou dan is het antwoord nee.

I: Oke. En , wat vind je eigenlijk van de manier hoe de testcases en dat soort zaken worden vastgelegd in Jira?

J: Ja, ik vind een paar dingen. Af en toe doen wij redelijk kleine stories die eigenljik cosmetisch van aard zijn en dan vind ik het gewoon veel overhead om daar logical testcases voor te schrijven want of ik dan de testcase lees of de acceptatiecriteria in de story lees dat is bijna een op een hetzelfde omdat het zulke simpele stories zijn. Dus dan hebben wij inmiddels ook de vrijheid dat we mogen zeggen bij zo'n story om te zeggen van we verwachten nu nul logical testcases. Want anders zijn we echt aan werkverschaffing aan het doen.

S: Voorbeeld. Een label aanpassen ofzo.

J: Ja. Of verplaats een invoerveld. Naja, noem eens wat dom .

J: Ennn.. Voor de overige stories vind ik het opzich goed om logical testcases te schrijven. Heb ik alleen nog wel eens wat moeite met het format zegmaar. Ik ben gewend dat ik of een beslistabelletje of zo'n matrixje of een graaf maak waarin je je logische paden tegen elkaar afzet. En zoiets kan ik niet kwijt in Jira. Dus nu heb ik dat ik alsnog voor mijzelf even uitteken wat de testgevallen moeten zijn. Dan denk ik van nou, nu heb ik voldoende informatie op m'n kladblok om die testen uit te kunnen voeren. Maar, oja, dan moet ik ze voor de administratie ook nog in Jira zetten. En op het moment dat die logical testcases daadwerkelijk geautomatiseeerd moeten worden. Dan denk ik van nou, dat is goed want dan hangt het samen met je kwaliteitsrapportage. Prima. Maar anders zou ik zeggen, wat voegt het toe.Je kunt die dingen ook op will not be executed zetten.

I: Ja

J: Wat betekend dat ze maar een keer uitgevoerd worden. Nou onze PO's kijken nooit naar die dingen. Ja jullie als ontwikkelaars doen vaak nog een review op de LTC's.

S: Vaak, altijd.

J: Altijd. Ja en daar wil het nog wel eens handig voor zijn dat een ontwikkelaar naar die

testgevallen kijkt en dan ofwel tegen mij zegt van joh, je mist dit of dat. Ofwel dat je als ontwikkelaar ook kunt denken van, wacht, ik heb deze testsituatie over het hoofd gezien. Dat heb ik nog niet gebouwd. Dus, maar. Dat is wel handig. Maar ik vind dat het ook nog best vaak verzand in meer wat punten en comma's rechtzetten zegmaar in een administratief dingetje dan dat het echt de testen verbetered. Of dat het de kwaliteit van het ontwikkelproces verbeterd.

S: Ja, eens.

J: Toch.

S: Ja. Ik moet zeggen dat ik het opzich best wel veel werk vind om zo'n LTC in Jira op te voeren. Je moet hem handmatig koppelen aan de story. Je moet, ja..

J: Je moet er met het reviewen van de story ook nog aan denken om de status om te zetten naar reviewed.En hij moet op resolved staan, en.

S: Je merkt echt.. Je kunt niet al die dingen achter elkaar uittypen. Je moet dan weer een nieuw issue maken, dingen koppelen. Het overzicht raak je ook best wel snel kwijt volgens mij.

J: Ja klopt.

S: Ik ben er wel aan gewend. Ik heb meestal twee schermpjes open staan. Links de story met de LTC's en rechts ga ik dan nieuwe dingen maken.

J: Ja, doe ik ook, ja.

S: Maar toch is het eh….

J: Maar dat vind ik met reviewen ook. Je moet een LTC openen om te kunnen reviewen. Maar je mist dan de samenhang met andere LTC's terwijl in principe zou je dan willen zien van oke als de ene LTC dit afdekt, hoe hangt dat dan samen met wat de rest afdekt weetje wel. Dat overzicht mis je ook

34:42

S: Ja.

J: Wat is de reden dat je dat vraagt eigenlijk.

I: Nou omdat dat samenhangt met , bij de meeste projecten in elk geval, hoe de hele workflow is ingericht in zo'n continuous delivery straat. Want het begint natuurlijk met de wijziging die je wilt doen, je user story. En van daaruit begin je vaak te werken. En inderdaad, daar hangen die LTC's mee samen, de implementatie van de test case, de kwaliteitsrapportage. Het hangt allemaal met elkaar samen en , ja als het ene heel veel extra werk of onnodige handelingen dan kan dat zie je bij sommige teams dat dat doorsijpelt in de rest van het werk.

S: Ja, wij gaan daar best wel pragmatisch mee om. Soms heb je eigenlijk vijf LTC's nodig om verschillende scenario's te doen, maar die proppen wij vaak in 1 dan.

J: Ja zeker als we toch al van.. Bij ons is er een mechanisme dat de PO per user story aangeeft of die hem belangrijk genoeg vindt om hem op te nemen in de ART.

S: Ja of handmatig, we hebben ook nog een deel handmatig testen.

J: Als die PO al zegt van joh, hij moet goed getest worden want hij moet goed opgeleverd worden, maar ach hij hoeft niet de regressietest in. Dan kun je iets makkelijker met je LTC's omgaan.

S: Ja dat doen wij.

J: Ja ik vind dat opzich ook wel terecht hoor.

S: Ja, nee ik ook. Want anders zou ik het niet doen.

I: Je zei dat jullie ook handmatig testen. Wat is de reden daarvan? Is dat heel moeilijk te automatiseren?

S: Ja. We hebben een aantal testen… We hebben een outlook plugin ontwikkeld. Ja, dat is gewoon moeilijk testen. We hebben het wel ooit op de backlog gezet om dat te maken. We hebben een proefje gedaan met de tooling die we ook gebruiken om de windows applicaties te testen. In theorie moet dat werken. Maar de klant vond dat teveel werk. En die plugin gaat er op termijn ook weer uit, dus. Dat gaan we ook niet meer doen denk ik. Zolang die in gebruik is gaan we dat elke keer handmatig testen, dat deel.

I: Maar valt dat dan mee, het werk dat je daarmee hebt? Is dat ooit een probleem of niet?

S: Ja, het is misschien een uurtje werk ofzo.

J: Als ik hem helemaal doe ben ik een uurtje of twee bezig.

I: Oke.

S: En dan hebben we nog twee testjes denk ik, voor een service. Windows service die periodiek zegmaar draait. Ja vanuit de ART moeten we dat ding kunnen stoppen en starten zegmaar. Ja daar hebben we maatwerk voor nodig denk ik, want dat kan nu niet. Je moet eigenlijk op die Windows machine inloggen remote en dan de service stoppen en starten zegmaar.

38:12

S: Dat pastte toen niet in het automatiseringsproces dus dat doen we ook met de hand. En we hebben een hele set aan…

I: Dat kan niet gewoon via powershell? Via dat scriptje?

S: Jaja, dat kan wel.

I: Maar je kunt niet op Windows remote inloggen en iets van powershell uitvoeren?

S: Ja, dat kan je ook remote uitvoeren. Maar dat moet wel gemaakt worden. Laat ik het zo zeggen.

J: Dan moet je de aansturing vanuit je tooling ook wel weer regelen. En dat zou dan weer custom werk zijn waarvan de klant zei, nou, laat maar zitten.

S: Ja.

J: En gezien hoe relatief weinig tijd, want het is maar een keer in de drie weken een uurtje voor een tester of ontwikkelaar om zo'n testje uit te voeren versus de ervaring die we toen hadden dat het best wel veel tijd kostte om dat in de tooling op te nemen, ja is toen besloten van het is het niet waard.

I: Maar die handmatige testen is nooit een struikelblok? Aan het einde van de sprint, onder tijdsdruk?

S: Ja„ ja.. we doen het liever niet maar. Het moet. We hebben geen andere oplossing er voor op korte termijn. We hebben nog een setje die is semi-automatisch. Een setje restful services die we testen. Ja ook een beetje een lang verhaal denk ik maar dat doen we met soap-ui. een test suite hebben we daarin gemaakt. Moet je af en toe wat handmatige dingetjes in doen.

J: Ja het punt is, daar zouden we Win-ART voor moeten gebruiken..

S: nou, of TestX.

J: We willen natuurlijk…. We moeten eerst de uitgangssituatie goedzetten en controles doen.

S: Ja dat kan ook in SQL heh.

J: Misschien nog niet eens zo'n slecht idee trouwens.

S: Nee, nee. Maar dan moet je..

J: Ook dan moet je er weer effort in steken.

S: Het idee was ooit van uh, services worden niet geautomatiseerd heeft iemand ooit gezegd. Automatisch getest. Ja de frontend die de service consumeert die moet je testen, maar de frontend is een mobiele applicatie die niet in beheer meer ligt bij de ontwikkelorganisatie. Dus ja die gaan wij nooit automatiseren.

J: Het idee was we wachten met het automatiseren totdat die frontend ook bij ons in beheer komt, want dan gaan we die automatiseren.

I: Maar dat is dus nog een andere applicatie die bij een andere partij in beheer is?

S: Ja.

I: En daar werken jullie dan wel nouw mee samen?

S: Jawel.

J: Wij bouwen de backend en zij bouwen de frontend. En uiteindelijk moet dat op elkaar klikken.

S: Meestal gaat het wel goed.

J: Totdat ze in productie gaan. en dan uh.

S: Ja dan gaat het weer stuk, ik weet niet maar. Ja. 't is best wel complexe infra denk ik. Zit weer het Good platform tussen. Blackberry, is een of andere beveiliging. Lijkt een beetje op het Blackberry netwerk zeg maar. De applicaties draaien dan weer in een geisoleerde container. Met een VPN verbinding met het bedrijfsnetwerk. Maar die schil die er omheen zit veroorzaakt ook veel problemen.

41:36

I: Hebben jullie daar last van? Of zij juist?

S: In principe hebben wij daar niet direct last van nee. Maar vaak dan in de ontwikkeling zie die Good schil er niet omheen en werkt alles. Dus ook met onze backend. Dan gaat het in productie met die Good schil er omhheen en de configuratie daar en dan werkt het niet meer.

I: Ja, oke. Hebben jullie die mobile app ook hier om dingen mee te proberen?

s: Nee eigenlijk niet.

I: Hoe gaan jullie dan om met wijzigingen aan die backend service? Dat communiceren jullie op een of andere manier naar dat team?

S: Ja, in principe ligt het een beetje bij ons. Er komt een wens van de klant, ik wil dit met die app doen. Wij kijken wat voor data daarvoor nodig is. Dan bedenken wij de interface en sturen we dan naar die andere partij. Die mogen daar wat van vinden. Meestal vinden ze er niks van.

J: Het zijn opzich eenvoudige services. Het is niet alsof er zulke berichten over de lijn gaan.

I: De communicatie met de klant loopt wel via jullie? dus jullie bouwen dan die backend service en dan communiceren jullie met het team dat de app bouwt..

S: In principe niet, dat staat helemaal los dat doet de klant zelf. Ik heb technisch wel contact met ze maar functioneel en hoe die app eruit ziet daar doen wij niet zoveel mee. In principe niets.

J: Het usability aspect wat natuurlijk aan zo'n app zit enzo dat is tussen die partij en de klant.

43:04

I: Hebben zij dan hun eigen setup van die tooling? Of van de applicaties om die app te kunnen testen.

S: In principe testen zij niet met onze omgeving. zij hebben zelf een soort dummy service ingericht denk ik op basis van de interface beschrijving.

I: Maar komt het dan echt pas voor het eerst samen in productie?

S: Nee, nou, niet helemaal volgens mij. Wij lopen meestal een stuk voor met ontwikkeling van de backend. Dus die staat dan al in de testomgeving bij de klant.

I: Ja.

S: en op die omgeving kunnen ze dat wel testen.

I: Oh ok. En daar wordt ook al gewerkt met dat beveilingsplatform, of dat dan weer niet?

S: Nee, dat niet. Dat komt pas in productie naarboven en daar klapt het nog wel eens.

J: Dat zou een goed idee zijn als ze dat al in de testomgeving doen.

S: Ja dat weet ik maar dat was 'gedoe'.

I: In dit geval is dit dus niet voor jullie een probleem en ook niet voor hun, maar meer voor de klant alleen?

S: Uiteindelijk is het voor ons allemaal een probleem.

J: Je krijgt vanuit productie dan vragen van kunnen jullie eens meekijken met wat er nu is gebeurd.

S: Ja.

J: En dan mis je inderdaad een geintegreerde omgeving. Die bouwer zegt dan van joh kunnen jullie je soap-ui projectje even sturen want dan weten wij precies hoe zo'n request eruit moet zien. En dan zeggen wij, stuur het berichtje wat er bij jullie uitkomt maar eerst naar ons misschien kunnen wij dat tegen onze software afvuren. En dat, ja, dan is het jammer dat je niet ergens een geintegreerde omgeving hebt waar we met z'n allen in kunnen kijken.

S: Ja productie.

I: Of een soort pre-productie. Je weet ook niet wat de reden is dat ze dat niet gewoon in pre-productie of acceptatieomgeving hebben ingericht met dat beveiligsingsplatform erbij. Zijn dat licentiekosten waarschijnlijk?

S: Ja, ze hebben daar een omgeving voor, maar die is niet ingericht. Wat de reden daar precies van is weet ik niet. Tijd, kosten, kennis. Iets.

I: Oke, duidelijk. Andere teams die hebben de mogelijkheid.. Jullie weten denk ik wel wat wij doen toch met het Docker platform? Of niet?

S: Mja, ik weet wat Docker is.

I: De andere teams hebben nu.. Kijk vroeger was het zo dat de contunous integration omgevingen werden beheerd door het interne beheer team. Dan had je 1 Jenkins, of iedereen had eigenlijk 1 Jenkins, 1 build server en daar werkte iedereen op. Of sommige teams hadden een eigen instantie. Maar daar kon je zelf niets op installeren, geen plugins of configuratie aanpassen naar de wensen van het team. Dat kan nu wel. Daarbij kunnen ze via het Docker Dashboard, die hebben jullie waarschijnlijk ook toch?

s: Ja, hebben wij ook.

I: Eigenlijk elke dienst starten die je nodig vindt ter ondersteuning van je ontwikkelstraat. Wat je als team maar wil. Hebben jullie behoefte aan zo'n dienst? En dan met name in de Microsoft wereld. Docker komt natuurlijk ook naar Windows dus.

S: Wij hebben daar nog niet echt behoefte aan denk ik. Misschien zou het het werk iets kunnen vergemakkelijken hoor. Maar het is nu allemaal ingericht.

I: De standaard tooling die er is, werkt voldoende voor jullie project en team?

S: Ja het werkt goed. Met alle scriptjes die wij eromheen gebouwd hebben zegmaar, naar onze eigen behoefte. Werkt dat opzich wel.

I: Maar bijvoobeeld ad-hoc omgevingen kunnen…

J: Ja, maar omdat het zo specifiek is, microsoft omgeving, denk ik dat de kans groot is dat je een beetje verzand in dezelfde situatie die we met de test tooling gehad hebben. Dat je dan een soort van standaardoplossing vanuit het IQ-team ondersteund kan worden. Maar dat dan in de praktijk gaat blijken dat die toch gezien onze omgeving niet helemaal voldoet en dat je dan best wel weer heel veel effort moet gaan steken in zoiets werkend te krijgen. en, ja dat is voor ons gezien alle investeringen die de klant heeft gedaan in zegmaar wat we nu draaiend hebben denk ik niet dat ze daar echt open voor zouden staan.

s: Wat is Docker voor Windows exact.

I: Dat is een hele goeie, daar moet ik ook nog beter in gaan duiken. Wat het nu is is dat Docker voor Windows ondersteund nu alleen Linux applicaties. Want Docker draait dan in de hypervisor. Je hebt dan eigenlijk Linux naast windows.

48:04

S: Je draait Docker op hyperv?

I: Ja.

S: Maar kun je daar binnen ook Windows?

I: Nog niet dus. Neen nog niet. Dat komt wel.

S: Maar ik kan dan in theorie een SQL instantie draaien?

I: Ja. Dus dat is wel iets waar bij aankomend jaar ook echt naar willen gaan kijken. Het is nu nog niet productiewaardig. Maar dat zal wel komen.

S: Ik denk niet dat wij dat snel binnen onze projecten ook zullen gaan gebruiken hoor. Waarschijnlijk worden alleen de nieuwste versies ondersteund.

I: Ja, ik heb geen idee.

S: Nou waarschijnlijk. Dat zie ik ook bij dot net core nu. Dot net vijf bestaat niet meer, dat heet nu dot net core.

I: Ja.

S: Werkt allemaal iets anders, lightweight, blabla. Dat draait, dat werkt in Docker volgens mij. Het is multiplatform gemaakt. Draait op Linux, dus ook in Docker volgens mij.

I: Dus dat is dan open source of niet.

S: Dot net core is ook open source ja. Maar die technieken gebruiken wij niet. Wij zijn gebonden aan de oude technieken. En ik denk die oude technieken waaraan wij gebonden zitten gaan nooit draaien in Docker. Ik denk niet dat ze oude spullen compatible gaan maken.

J: Maar dat ligt dus niet zozeer aan Docker, maar die dot net vier gaan ze nooit.

S: Dot net vier gaat denk ik nooit, nee. Waarom, nee, dat is veel te veel werk.

I: Ik heb geen idee. Dot net core kun je op Linux draaien. Dat kan dan nu met Docker omdat Windows Docker Linux ondersteund via HyperV. Maar echte Windows applicaties kun je nog niet met Docker draaien. Maar dat, komt er als het goed is aan. Dan zou je in theorie elke Windows applicatie in een container moeten kunnen draaien. Dat zou dan ook voor oudere applicaties gelden.

S: Maar het is allemaal…

I: Geen idee wanneer dat zo ver is. Maar bijvoorbeeld iets als omgevingen on-demand. Want jullie hebben nu een x aantal statische omgevingen.

S: Ja.

I: Zouden jullie daar iets aan hebben? In de zin van, bijvoorbeeld, parallel testen draaien om te doorlooptijd te verkorten.

S: Ja opzich wel. Dat was ook ooit het doel, om dynamische omgevingen in te richten. , ja L was daar mee bezig.

J: Ja? Omdat ook echt voor elkaar te krijgen?

S: Dat was de oorspronkelijke wens. Ik weet niet precies waarom, maar het lukte niet of… Toen hebben we die statische omgevingen gehad. En ja opzich werkt dat voor ons. We hebben tien van die dingen, we werken met vier man. We hebben in theorie twee omgevingen per persoon tot onze beschikking. Het kost denk ik wat om die servers in de lucht te houden, maar dat zien wij niet. Opzich is het mooier om het dynamisch te kunnen doen maar dat gaat niet zo makkelijk voor ons.

I: Het is nu niet direct een bottleneck voor jullie?

S: Nee, het was een bottleneck omdat we maar vijf omgevingen hadden. En meer virtuele omgevingen kon niet, qua kosten denk ik. Toen hebben we die dubbele instantie erbij gedaan. Dat we twee instanties per vm kunnen draaien. En nu zijn we verre weg van… er is altijd wel iets vrij.

51:29

I: Hoe kijken jullie aan tegen de ondersteuning vanuit de organisatie voor jullie project? Misschien in technische zin, of procesmatige zin? Missen jullie daar iets? Wordt er teveel gestuurd? Deze vraag zit er met name in omdat bij andere teams Docker min of meer naar binnen is geschoven. Zo van, we gaan nu Docker doen, terwijl daar vanuit de organisatie misschien wel te weinig ondersteuning voor was.

J: Dat hebben wij ervaren met de test tooling zeg maar. Dat is eigenlijk gepushed van je moet dit gebruiken. Ja, maar het werkt niet. Nouja.

I: Dat is gepushed door de technical lead van het project?

J: Ja.

S: Docker wilden ze eigenlijk ook pushen.

I: Maar hoe dan?

S: Ja, dat kan niet, dus. Er waren toen ook al verhalen over Docker voor Windows. Maar vooralsnog is het er nog niet. Maar over ondersteuning vanuit de ontwikkelorganisatie.

I: Ja, met name voor dit soort zaken dan.

S: Ja ik vind niet dat er echt ondersteuning is voor ons.

I: Maar mis je dat? Je zei net dat we alles wel in het team kunnen oplossen. Dat is natuurlijk ook heel goed.

S: Ja, zo is dat een beetje gegroeid. We moeten het wel zelf oplossen. We hebben ons eigen ding bedacht en dat draait nu. Maar ja. Voorbeeld, we willen die Exchange server hebben.

I: Ja.

S: Ja, dat moeten we zelf oplossen. Het wordt niet ondersteund door de ontwikkelorganisatie.

I: Maar vind je het nu meer een nadeel dan een voordeel?

S: Ja, dat vind ik een nadeel ja.

I: Ja, maar in andere gevallen zou je ook kunnen denken van het is een voordeel. Je hebt zelf meer controle over je omgevingen.

J: Ja, ik vind opzich het voordeel dat je een stukje flexibiliteit hebt. Vanuit het team weten we met elkaar best wat de juiste tooling of de juiste way to go is. Dat er ruimte voor is dat je daar je eigen keuzes in kunt maken vind ik wel goed. Ik geloof namelijk niet in de omgekeerde gedachte die eerst bij de ontwikkelorganisatie heerste. Dat was van we hebben een blauwdruk en ieder project moet maar in die blauwdruk passen. Ja dat gaat niet. Als je een dot net project hebt of een Java project dat kun je niet allemaal in dezelfde blauwdruk stoppen. Dus dat die flexibiliteit er inmiddels is dat vind ik wel goed. Alleen dat staat los van een stukje technische of infrastructurele ondersteuning die misschien wel wat meer geboden kan worden.

S: Ja nou het gaat ook niet goed nu merk je.

I: Wat bedoel je dan?

S: Ja, Exchange is iets dat wij buiten het team willen houden. We hebben aan de ontwikkelorganisatie gevraagd van los het op, wij kunnen dit niet, wij willen dit niet. Het is uiteindelijk toch binnen het team gekomen. We zijn nu drie weken verder maar we zijn eigenlijk nog geen stap verder.

J: En uiteindelijk gaan we het toch weer zelf doen.

S: Ja we zijn het al zelf aan het doen. Maar er wordt ook veel heen en weer gebounced. Server verkeerd ingericht. Dat kunnen wij niet zelf. Dus er moet weer een nieuwe VM kloon komen.

J: Ik denk dat dat vaak ook het nadeel is van deze vorm van ondersteuning zegmaar. Dat je eigenlijk net te ver van de projecten af zit om alle ins- en outs te weten. In hoeverre kun je dan echt adequate ondersteuning bieden. Ze kunnen prima dat servertje neerzetten, maar wat was er nou waarom het voor ons alsnog niet ging werken? Ja dan denk je, kunnen zij (technisch beheer) dat weten. Ja misschien als ze iets proactiever zijn en even wat meer vragen aan ons.

S: Maar dat die eerste run verkeerd was?

J: Ja.

S: Ja. Ook ontbrekende kennis denk ik.

J: Ik vind dat ook wel lastig hoor. Wat dat betreft hou ik m'n hart vast als straks ook die performance en security testen, wat ook nu meer een dienst moet gaan worden, in plaats van een of twee handige harries die hier over de werkvloer lopen. En dat model werkt opzich. R heeft voor ons performancetesten gedaan. Ja zodra hij er even niet uitkomt dan loopt hij gewoon naar ons, of wij naar hem.

I: Ja.

J: En uiteindelijk los je het dan op doordat je gewoon bij elkaar zit. En nu gaat dat ook weer losgetrokken worden van dat wordt een dienst en op afstand en.

S: Eigenlijk wel een beetje hetzelfde idee als met IQteam en IT beheer. We gaan bijna nooit met elkaar samen zitten om iets op te lossen. We moeten een ticket aanmaken.

J: En dan gaat iemand naar dat ticket kijken die waarschijnlijk onze omgeving nog nooit gezien heeft, weet je wel. Hoe kun je dan goede ondersteuning bieden. En dan zeggen wij ja die gasten die weten het ook allemaal niet.

S: En zij roepen dat over ons.. Er zit een muur tussen eigenlijk.

I: Ja daarom ben ik ook dit aan het doen. Er zitten twee kanten aan, het is voor m'n scriptie maar het helpt wel. We hebben als best veel nieuwe inzichten gekregen.

S: Als wij ondersteuning willen hebben van IT beheer moeten wij exact zeggen wat we willen hebben. Wij willen een VM server dit hebben, met software X er op. We moeten hem zo geconfigureerd hebben.

J: Je moet het eigenlijk weten tot op het niveau waarop je het eigenlijk net zo goed zelf kan doen.

S: Als wij dus een foutje maken in onze aanvraag dan moeten we dus weer helemaal opnieuw beginnen.

J: Terwijl we willen gewoon zeggen van dit is ons probleem…

S: Lost het op..

I: Maar zou je dan toch niet liever zelf de mogelijkheid willen hebben om ergens met een paar kliks een vm aan te maken.

J: Ja ik geloof meer, misschien spreek ik voor m'n beurt, maar ik persoonlijk geloof meer in een soort van inleen model. Op het moment dat wij die expertise nodig hebben, voor deze drie stories, trekken wij iemand van zo'n expert team intern bij ons in het team.

S: Ik denk dat het veel efficienter is als wij iemand tijdelijk kunnen inhuren, of wat dan ook.

J: Voor het IQ-team ook hoor. Stel dat we ooit op het punt komen dat we met Docker willen gaan werken. Ik zou zeggen zet maar gewoon dedicated een IQ-man in ons team neer tot het moment dat het werkt zeg maar. Maar op afstand van hier heb je een server en dan tegen het team zeggen van nou ons taakje is gedaan, ga er maar wee werken. En dan zeggen wij na drie klikken ja maar dit werkt voor ons niet. Ja dan mikken wij dat weer terug over de schutting, weet je wel.

I: Nee, nee. Dat is natuurlijk ook niet de agile gedachte. Maar zo hebben wij dat in het begin ook wel gedaan met de projecten die Docker gingen doen. Dat wij daar een aantal uur per week

met hen samen gingen zitten om te kijken hoe we dat konden toepassen binnen hun project.

J: Ik weet niet wat jullie ervaringen daar mee zijn, maar het lijkt mij wel een model wat zou moeten kunnen werken. En als het eenmaal werkt dan kun je op afstand ondersteuning bieden.

I: Nou kijk de gedachten is natuurlijk dat, vroeger kon je als team zelf helemaal niets doen en moest je alles via technisch beheer regelen. En dan had je precies de problemen die je beschrijft. We zijn aan het kijken van je moet naar een juiste mix. Want je zei ook van ja, als team weet je goed welke services je nodig hebt in je ontwikkelstraat en hoe je bepaalde dingen wilt inrichten. Wat voor jou en voor dat project lekker werkt. Dus die vrijheid moet er ook zeker zijn, denk ik. We zijn aan het zoeken naar een juiste balans.

S: Ja, maar wij zijn wel iets anders begonnen met L. erbij. In principe was L voor ons diegene die voor ons de externe problemen oplostte. Ja die zal ook wel met technisch beheer gecommuniceerd hebben maar. Daar hadden we eigenlijk geen problemen mee. Voor ons was dat wel dienstverlening.

J: Ja, hij zat wel in dat team (technisch beheer) maar hij was een soort van dedicated…

S: Nu hij er niet meer is, gaat het, ja, anders.

J: Maar hij kon ons helpen doordat hij ook kennis had van onze omgeving.

S: Ja, hij wist ook van het project inderdaad.

J: En dan werkt dat inderdaad. Hij heeft kennis van de omgevingen en van het project. Dan heb je aan twee woorden genoeg om ons te helpen.

S: Als hij er nu nog was en we zouden zeggen van we hebben een Exchange server nodig, fix het maar. Hij zou waarschijnlijk ook nog met de klant gaan bellen om na te vragen waarom het nodig is en op welke manier.

I: Oke, duidelijk. Maar stel je zou iets in de cloud kunnen starten, dat zou niet een oplossing zijn? Dat je zou kunnen zeggen van we maken een image van Exchange.

S: Ja, dat zou perfect zijn. Als wij klik Exchange kunnen doen en wij kunnen daarmee koppelen. Zou ideaal zijn. Ook wel SQL server. Ja dat hebben we nou allemaal draaien, maar voor nieuwe dingen.

I: Misschien voor nieuwe dot net projecten die ooit gaan starten.

S: Als we in theorie ooit nog met sharepoint moeten gaan babbelen ofzo.. Ik weet dat de klant ook sharepoint heeft. Ja, klik sharepointomgeving. zou ideaal zijn.

J: Maar ja, dan zouden er dus dat soort Dockerized omgevingen moeten zijn. Out of the box die..

I: Ja, Docker maakt het misschien makkelijker. Maar je kunt ook eerst met, weet ik veel, automatisering op VM niveau werken.

S: Ja dat is ons om het even.

I: Maar dit is toch een goede aanleiding voor ons om naar Docker voor windows te gaan kijken. Dit is natuurlijk te laat voor jullie nu.

S: Nou dat weet ik niet. In principe kan dit project nog jaren en jaren draaien.

J: Ja maar dan moeten we niet zo'n traject krijgen als met de ART tooling gehad hebben. Zo van, hier heb je een halve oplossing.

S: Nee.

1:04:20

J: Uiteindelijk heeft dat niet heel veel goodwill bij onze klant oplgeverd ook. Want het wordt dan verkocht aan de klant van we hebben een werkende en kant en klare oplossing voor testautomatisering.

I: Ja.

J: Maar per saldo zien zij dat ze het hele jaar 25-30%, misschien wel meer, van hun sprint van hun velocity kwijt zijn aan het in de lucht brengen en houden van die tooling. Ja dat geeft toch een beetje scheve gezichten.

I: Ja.

J: Dus ik denk wel dat dat belangrijk is als we ooit besluiten om over te gaan op iets als Docker dat we hele goede afspraken met elkaar maken.

I: Oke, dan waren dit mijn vragen.