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.

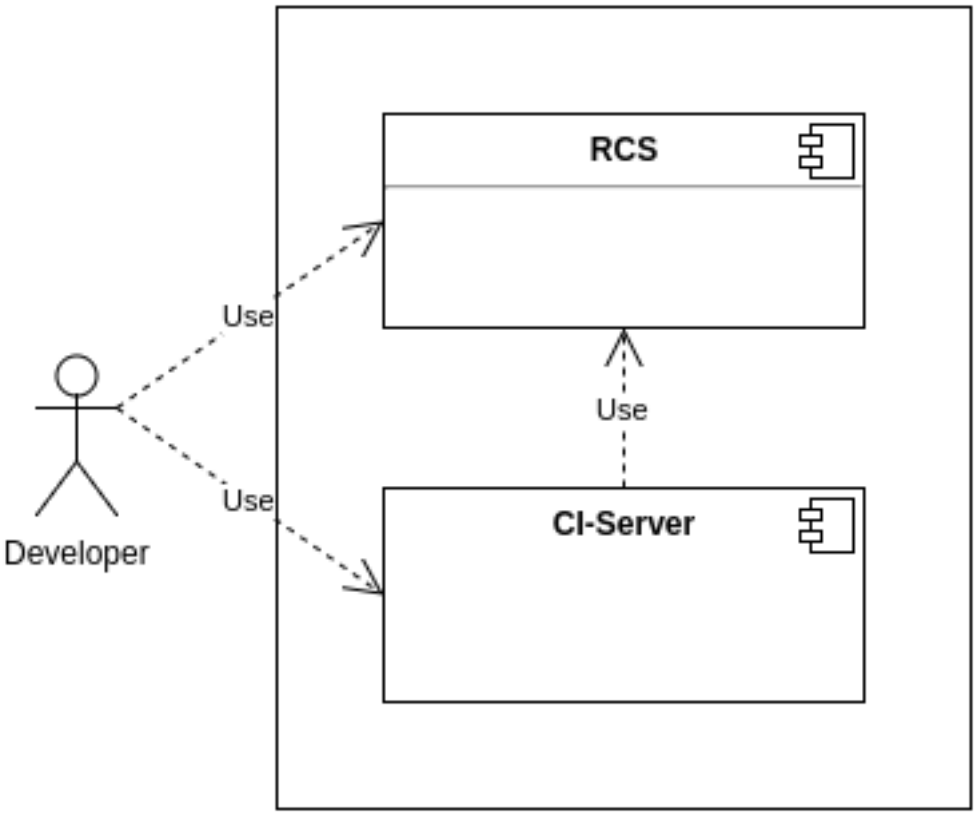

Figure 4.1: Overview Continuous Integration

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
6.

6.3.3 Recurring tasks

6.4 Configuring the CD-pipeline

6.4.1 Docker Dashboard

Every project team is equipped with a Docker Dashboard. The dashboard is a web application through which the team can manage running applications on the Docker infrastructure. The dashboard exposes a user interface and a programmable interface for automation purposes. Through the dashboard the team manages:

1. Deployment of CD-pipeline support services
2. Deployment of the application under development

Upon project start a vanilla dashboard is deployed. The team has the freedom to start any combination of Docker containers.

6.4.2 Configuration

Property	Value	Description
jira-reporter-user	reporter	
jira-reporter-password	****	

6.4.3 Nexus

6.4.3.1 Add nexus to Docker dashboard.

Go to the Docker Dashboard user interface, click 'Apps', 'New App'. Enter the following app definition:

```
name: nexus
version: 2.13.0-01
description: Sonatype Nexus repository manager.
```

```
#login with admin / admin123

#tags:infra

www:
 image: sonatype/nexus:2.13.0-01
 user: root
 volumes:
 - /sonatype-work
```

Click 'Save changes' and start the application.

6.4.3.2 Configure security

Go to the Nexus user interface and log in with

	Username	Password
Account	admin	admin123

Go to 'Security', 'Users'. Select user ***anonymous***. Give the user full control over all repositories. Click Add, select Repo: 'All Repositories (Full Control)'. Click 'OK' and 'Save'.

6.4.4 Gitlab

6.4.4.1 Add Gitlab to Docker dashboard

Go to the Docker Dashboard user interface, click 'Apps', 'New App'. Enter the following app definition:

```
name: gitlab
version: 8.6.1

#tags:infra

www:
  image: www.docker-registry.isd.tld:5000/gitlab-ce:8.6.1
  volumes:
    - /etc/gitlab
    - /var/log/gitlab
    - /var/opt/gitlab
```

Click ‘Save changes’ and start the application.

6.4.4.2 Create root account

Go to the Gitlab user interface. You will be asked to enter a new password for the root user. Enter the password twice and click ‘Change your password’.

6.4.4.3 Create usergroup and user(s)

Go to the Gitlab user interface. Log in with:

	Username	Password
Account	root	***see previous step***

Go to ‘Admin Area’, ‘Groups’, ‘New Group’. Enter:

Group path	Visibility Level
test-group	Private

Click ‘Create Group’.

Go to ‘Admin Area’, ‘Users’, ‘New User’. Enter:

Name	Username	Email

Click ‘Create user’.

Click ‘Edit’ and enter:

Password	Password confirmation
user@123!	user@123!

Click ‘Save Changes’.

Go to ‘Admin Area’ > ‘Groups’ > ‘test-group’ Add user to group with role ‘Developer’.

Repeat for each user in the development team.

6.4.4.4 Create Jenkins user

Jenkins should be able to login to Gitlab in order to be able to checkout copies of the source code. Therefore a dedicated user should be created.

Go to the Gitlab user interface. Log in with:

	Username	Password
Account	root	***see previous step***

Go to 'Admin Area', 'Users', 'New User'. Enter:

Name	Username	Email
Jenkins	jenkins	noreply@jenkins.tld

Click 'Create User'.

Click 'Edit' and enter:

Password	Password confirmation
jenkins@123!	jenkins@123!

Click 'Save Changes'.

Go to 'Admin Area' > 'Groups' > 'test-group' Add user jenkins to group with role 'Master'.

6.4.4.5 Import Jenkins SSH public key

Go to the Gitlab user interface. Log in with:

	Username	Password
Account	jenkins	jenkins@123!

Go to 'Profile Settings', 'SSH Keys'. Enter the SSH public key:

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCrSFSIYRJjTbWqYuU6cGQ0aNae
wMQ/m0k3m/MA7mNb2LEGNb0CQAoJDwwuPiftfj9cP6UrTlFqKGRKuLlAD7qrf9kQ
OSzgfwDJ7lSYY8QqMP4U1CuL5IuMV/Zwg0npA9SnPpD8KcrxMQgKZ62F12xoR+vX
LMSgMnTwu7olZVQphdMcvu2H5ugV4kBNyyRfKSeDDatsYKnwVirhLBRMtdFTLqo2
```

```
wFe8dMM/2mZIiGl5KXg0gCXpD2VEFiVCINARGLsdh9nzn2gxLoagIbXzxWjGRo0t
u69GuS2YqNj7GX5QJMpTP4UAWPvymx1TiJqmWAatejdfhYeJWoTLA6dnxaFV
```

Click 'Add key'.

6.4.4.6 Create project repository

Go to the Gitlab user interface. Log in with:

	Username	Password
Account	root	***see previous step***

Go to 'Admin Area' > 'Projects' > 'New Project'. Enter:

Project Path	Visibility
/test-group/test-project	Private

Click 'Create project'.

Go to 'Admin Area' > 'Projects' > 'test-group/test-project' > 'Edit' > 'Protected Branches' In table: 'Already Protected', Select Developers can push for branch master.

6.4.5 Docker Registry

6.4.5.1 Add docker-registry to Docker dashboard

Go to the Docker Dashboard user interface, click 'Apps', 'New App'. Enter the following app definition:

```
name: docker-registry
version: 2.1.1

#tags:infra

www:
 image: distribution/registry:2.1.1
 mem_limit: 2048m
 environment:
  - REGISTRY_VERSION=0.1
```

```
  - REGISTRY_LOG_FIELDS_SERVICE=registry
  - REGISTRY_LOG_FIELDS_ENVIRONMENT=production
  - REGISTRY_STORAGE_CACHE_BLOBDESCRIPTOR=inmemory
  - REGISTRY_STORAGE_FILESYSTEM_ROOTDIRECTORY=/docker-registry/
  - 'REGISTRY_HTTP_ADDR=:5000'
  - REGISTRY_HTTP_HEADERS_X-CONTENT-TYPE-OPTIONS=[nosniff]
  - REGISTRY_HEALTH_STORAGEDRIVER_ENABLED=true
  - REGISTRY_HEALTH_STORAGEDRIVER_INTERVAL=10s
  - REGISTRY_HEALTH_STORAGEDRIVER_THRESHOLD=3
 volumes:
  - /docker-registry
```

Click 'Save changes' and start the application.

6.4.6 Sonar

6.4.6.1 Add sonar to Docker dashboard

Go to the Docker Dashboard user interface, click 'Apps', 'New App'. Enter the following app definition:

```
name: sonar
version: 4.5.7
description: Manage code quality

#tags: infra
#login with admin/admin

www:
  image: sonarqube:4.5.7
  environment:
    - SONARQUBE_JDBC_USERNAME=sonar
    - SONARQUBE_JDBC_PASSWORD=sonar
   - "SONARQUBE_JDBC_URL=jdbc:mysql://db/sonar?useUnicode=true&amp;characterEncoding=u
  volumes:
    - /opt/sonarqube/extensions/downloads
    - /opt/sonarqube/extensions/plugins
  links:
    - db
  enable_ssh: true

db:
  image: mysql:5.6
  environment:
```

```
    - MYSQL_ROOT_PASSWORD=root
    - MYSQL_DATABASE=sonar
    - MYSQL_USER=sonar
    - MYSQL_PASSWORD=sonar
  volumes:
    - /var/lib/mysql
```

Click 'Save changes' and start the application.

6.4.6.2 Install plugins

From a local shell execute:

```
cd /path/to/dir/with/sonar/plugins
scp sonar-checkstyle-plugin-2.4.jar \
    sonar-findbugs-plugin-3.3.jar   \
    sonar-java-plugin-3.14.jar      \
    sonar-pmd-plugin-2.5.jar        \
    sonar-web-plugin-2.4.jar        \
    www.sonar.<your-project>.tld

ssh www.sonar.<your-project>.tld
cd /opt/sonarqube/extensions/plugins
rm * && cp ~/* .
exit
```

Go to the Docker Dashboard user interface and restart the Sonar application.

6.4.6.3 Add quality profiles

Go to the Sonar user interface. Login with:

	Username	Password
Account	admin	admin

Go to 'Quality Profiles', Click 'Restore Profile'. Select 'Development Organization Java profile' to import and click 'Restore'. Click 'Restore Profile'. Select 'Development Organization Web profile' to import and click 'Restore'.

6.4.7 Reporting

6.4.7.1 Create Jira filter

Go to the Jira user interface. Login with:

	Username	Password
Account	jira-reporter-user	jira-reporter-password

Go to 'Issues', 'Search for issues'. If the basic search is shown instead of the advanced search, click Advanced. Enter the following query:

```
project = <Jira project name> AND type in (Story, "Logical Test Case", Systeemfunctie) ORD
```

Click 'Save as'. Filter name: . Click 'Submit'. Write down the filter-id of the filter (it's displayed in the URL, https://jira.development-organization.nl/jira/issues/?filter=).

6.4.7.2 Add reporting application to Docker Dashboard

Go to the Docker Dashboard user interface, click 'Apps', 'New App'. Enter the following app definition:

```
name: reporting
version: 2.2.2
description: Quality reporting

#tags: autorun

www:
  image: docker-registry.isd.tld:5000/birt-reports:2.1.65
  mem_limit: 2g
  environment:
    - REPORT_USER=reporter
    - REPORT_PASSWORD=reporter007
    - 'REPORT_URL=jdbc:postgresql://db:5432/birt'
    - REPORT_USER_RM=reporter
    - REPORT_PASSWORD_RM=reporter007
    - 'REPORT_URL_RM=jdbc:postgresql://db:5432/birt'
  links:
    - db

importer:
```

```
  image: docker-registry.isd.tld:5000/birt-jira-importer:2.4.1
  environment:
    - 'report_jdbc_url=jdbc:postgresql://db:5432/birt'
    - 'jira_filter=filter=<filter-id>'
  links:
    - db

trr:
 image: docker-registry.isd.tld:5000/birt-test-results-service:2.0.50
  environment:
    - 'report_jdbc_url=jdbc:postgresql://db:5432/birt'
  links:
    - db

db:
    image: docker-registry.isd.tld:5000/birt-database:2.0.37
    volumes:
        - /var/lib/postgresql/data
    environment:
        - POSTGRES_PASSWORD=my-secret-pw
    mem_limit: 2g

rm:
  image: docker-registry.isd.tld:5000/releasemanager:1.0.36
  environment:
    - DB_DRIVER=pdo_pgsql
    - DB_HOST=db
    - DB_PORT=5432
    - DB_USER=releasemanager
    - DB_PASSWORD=releasemanager007
    - DB_DATABASE=birt
  volumes:
    - /mnt/publish
  links:
    - db
```

Click 'Save changes' and start the application.

6.4.8 Selenium

6.4.8.1 Add Selenium to Docker dashboard

Go to the Docker Dashboard user interface, click 'Apps', 'New App'. Enter the following app definition:

```
name: selenium
version: official

#tags: autorun

server:
  image: selenium/standalone-firefox
```

Click 'Save changes' and start the application.

6.4.9 Quality-dashboard

TODO! detail how to setup the quality tracking and monitoring system
See http://wiki.isd.org/index.php/HandleidingKwaliteitssysteem > Opzet van een kwaliteitsdashboard

6.4.10 Jenkins

6.4.10.1 Add Jenkins to Docker dashboard

Go to the Docker Dashboard user interface, click 'Apps', 'New App'. Enter the following app definition:

```
name: jenkins
version: 1.651.2
description: An extendable open source CI server

#tags: autorun

www:
  image: jenkins:1.651.2
  volumes:
    - /var/jenkins_home
  environment:
    - "JAVA_OPTS=-Duser.timezone=Europe/Amsterdam"
  links:
    - jnlp
  enable_ssh: true

jnlp:
  image: tehranian/dind-jenkins-slave:latest
  environment:
   - "DOCKER_DAEMON_ARGS=-H unix:///var/run/docker.sock -H tcp://0.0.0.0:2375 --insecur
```

```
  - "JAVA_OPTS=-Duser.timezone=Europe/Amsterdam"
mem_limit: 2g
privileged: true
enable_ssh: true
```

Click 'Save changes' and start the application.

6.4.10.2 Install plugins

Go to the Jenkins user interface. Click 'Manage Jenkins', 'Manage Plugins', 'Available'. Tick the box next to the following plugins:

- Maven Release Plug-in Plug-in (0.14.0)
- Git plugin (3.0.0)
- SonarQube Plugin (2.4.4)
- OWASP Dependency-Check Plugin (1.4.3)
- Git client plugin (used by Git plugin)
- SCM API Plugin (used by Git plugin)
- Conditional BuildStep
- Run Condition Plugin
- Parameterized Trigger plugin
- Workspace Cleanup Plugin
- Build Pipeline Plugin

6.4.10.3 Configure system

Go to the Jenkins user interface. Click 'Manage Jenkins', 'Configure System'. Look for the section 'SonarQube servers'. Click 'Click Add SonarQube'. Enter:

Property	Value
Name	SonarQube
Server URL	http://www.sonar..tld:9000
Server version	5.1 or lower
Version of sonar-maven-plugin	3.0.1
Database URL	jdbc:mysql://db.sonar..tld:3306/sonar
Database login	sonar
Database password	sonar

Look for the section 'Git plugin, Global config'. Enter:

Property	Value

Property	Value
user.name	jenkins
user.email	noreply@jenkins.tld

Look for the section ‘Maven’. Click ‘Add Maven’. Enter:

Property	Value
Name	Maven 3.3.9
Version	3.3.9

Look for the section ‘E-mail Notification’. Enter:

Property	Value
SMT server	smtp.isd.org

Click ‘Save’.

6.4.10.4 Add Gitlab user

Go to the Jenkins user interface. Click ‘Credentials’, ‘Global’, ‘Add credentials’. Enter:

Property	Value
Kind	SSH username with private key
Scope	Global
Username	jenkins
Private key	Enter directly
Description	Gitlab user

Enter the key:

```
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAq0hUiGESYO21qmLlOnBkNGjWnsDEP5tJN5vzAO5jW9ixBjW9
AkAKCQ8MLj4n7X4/XD+lKO5RaihkSri5QA+6q3/ZEDks4H8Aye5UmGPEKjD+FNQr
i+SLjFf2cINJ6QPUpz6Q/CnK8TEICmethddsaEfr1yzEoDJO8Lu6JWVUKYXTHL7t
h+boFeJATcskXykngw2rbGCp8FYq4SwUTLXRUy6qNsBXvHTDP9pmSIhpeSl4NIAl
6Q9lRBYlQiDQERi7HYfZ859oMS6GoCG188VoxkaNLbuvRrktmKjY+xl+UCTKUz+F
```

```
AFj78psdU4iaplgGrXo3X4WHiVqEywOnZ8WhVQIDAQABAoIBAHSw9W5oe+eNpMut
TrBuq8YM+tLzT4BqIgqxw2+J+cUOLv6lE9z5lhyv1MOYcwlZLn+BmNyVIeBqHlHN
4d+kF7AJjO+BlHJp9DaemaGsrpNOB1ZXakeHcA8wSmRC/dKzWmiKtqolKu8BUZIN
Kmn55xBwl1tkU5OOYvkzXFFn5FvYg7NzvdgajfrywgU6GIm6miGWzkbOF5MRnXtb
hhx32sSu9H87Fu54DpMIWQzzpqJuPPLL8SxZKheuceYcV/tXT6IG5WnSl3KnNVYX
cSLRN4miN6dVlX9GIdwsAxdrqexm/OLw/J5Mgf2SaIKslMNoGciojtaQaLt/ofde
UWaOlWECgYEA3kg9Tv8/iIWIzIYvgC8D/ZyjyExlcX6jVSrbFCjZyYhDqxA2LPSa
SQwcHesLk4c4GS1in47Mo/otxdPYnmu2oOOb4hoICGLEJiWdiI7ljJ8CWtsjNGts
GhCsv3guTT1WnfOpWbsaKDodinEa2YNekxAjh/9EQZR4x6Y9EQ61C8kCgYEAxUOm
W8orY6M4YYeb1nA9izW1twjrK0jZ6u07n3ooAVJx5WNr69/ATZx/Lw4ou1RGZ3qb
q4/iol5DwenxcaETV5h5q/170pvnhkLoRIWd+Rd+oEu+GtYwVrLd/ybJZswJMYc/
KIRHrw3DpM0yVxrilENe3TQ8k2O3VJRsucmHly0CgYEA2Jq+m5dh2vB9MQhli1zF
X8LfWxUPGXzVPu4HFGsGZzvQ7QZcNIybOCmD0Ke13So8QVSXsXJe+j+VkRxyD1ZZ
YFlGsxq4zysnhyDKlULib5iXm9/FO5Sef/vVyrMbM4tdN4g0c8s+nwqatMio6GL6
qwZkCWd3pQxAchUNluylAfkCgYArkDIH6VDFs0D7QOBobecZfCYCIuUUbQU6/WMC
aA63pAZlGxy1PXeRbDMmKCFUpVra9Ve1fpQVOW4LP+fDKUhFOvX7xoH209lAbDwx
DbUCUm7zZWa5NH3+V4fxFha6LesF1hFbmELgZNDE70/jrptFFM+5WBTck9PjyNdt
/BSGjQKBgDS8bI/B6uMos882AG4eOcUBEVdaTaOIBqJEM0s5u8PPNBaXsx4kfH62
9vnfrX3tf8fj3UgrIsqEg/N2Pze2ktj8ikqz4cIJqX0fHHvEYC+FvqDcdit14Cv9
q0lAlP1AXSP4kry7SguwMTlewfcMUXxwTEIs0PXujqx8uTBLnUBY
-----END RSA PRIVATE KEY-----
```

Click ‘OK’.

6.4.10.5 Add SSH user to connect to dind-slave

Go to the Jenkins user interface. Click ‘Credentials’, ‘Global’, ‘Add credentials’. Enter:

Property	Value
Kind	SSH username with password
Scope	Global
Username	jenkins
Password	jenkins
Description	SSH user to connect to dind-node

Click ‘OK’

6.4.10.6 Add dind node

Go to the Jenkins user interface. Click ‘Manage Jenkins’, ‘Manage Nodes’, ‘New Node’. Enter:

Property	Value
Node name	docker-in-docker
Type	Dumb Slave
Number of executors	4
Remote root directory	/tmp
Labels	docker
Usage	Only build jobs with label restrictions matching this node
Launch method	Launch slave agents on Unix machines via SSH
Host	jnlp.jenkins..tld
Credentials	jenkins (SSH user to connect to dind-node)

Click ‘Save’

6.4.10.7 Create Jenkins job for Sonar

Go to the Jenkins user interface. Click ‘New item’.

Property	Value
Item name	sonar-app
Type	Maven project

Click ‘OK’.

Configure job sonar-app

Go to the Jenkins user interface. Click ‘sonar-app’, ‘Configure’.

	Property	Value
Discard old builds	Max # of builds to keep	5
Git	Repository URL	git@www.gitlab..tld:test-group/test-project.git
	Credentials	jenkins (gitlab user)
Build	Root POM	testapp/pom.xml
	Goals	clean install -DskipTests
Post-build Actions	***select***	SonarQube analysis with Maven

Click ‘Save’.

6.4.10.8 Create Jenkins job for OWASP dependency checker

Go to the Jenkins user interface. Click ‘New item’.

Property	Value
Item name	dependency-check-app
Type	Maven project

Click ‘OK’.

Configure job dependency-check-app

Go to the Jenkins user interface. Click ‘dependency-check-app’, ‘Configure’.

	Property	Value
Discard old builds	Max # of builds to keep	5
Git	Repository URL	git@www.gitlab..tld:test-group/test-project.git
	Credentials	jenkins (gitlab user)
Build	Root POM	testapp/pom.xml
	Goals	clean install -DskipTests
Post Steps	***select***	Run regardless of build resultaat
		Invoke OWASP Dependency-Check analysis
	click	Advanced
	select	Generate optional HTML reports
Post-build Actions	***select***	Publish OWASP Dependency-Check analysis results

Click ‘Save’.

6.4.10.9 Create job build-app

Go to the Jenkins user interface. Click ‘New item’.

Property	Value
Item name	build-app
Type	Maven project

Click ‘OK’.

Configure job build-app Go to the Jenkins user interface. Click ‘build-app’, ‘Configure’.

	Property	Value
Discard old builds	Max # of builds to keep	5
Git	Repository URL	git@www.gitlab..tld:test-group/test-project.git
	Credentials	jenkins (gitlab user)

	Property	Value
Additional Behaviours		Check out to specific local branch
	Branch name	master
Build Triggers	***select***	Poll SCM
	Schedule	H/5 * * * *
Build environment	***select***	Installed maven version
Build	Root POM	testapp/pom.xml
	Goals	clean install

Click ‘Save’.

6.4.10.10 Create job build-image

Go to the Jenkins user interface. Click ‘New item’.

Property	Value
Item name	build-image
Type	Freestyle project

Click ‘OK’.

Configure job build-image Go to the Jenkins user interface. Click ‘build-image’, ‘Configure’.

	Property	Value
Discard old builds	Max # of builds to keep	5
Restrict	Label Expression	docker
Git	Repository URL	git@www.gitlab..tld:test-group/test-project.git
	Credentials	jenkins (gitlab user)
Build	***select***	Execute shell
	Command	cd docker && ./build.sh

Click ‘OK’.

6.4.10.11 Create job sonar-art

Go to the Jenkins user interface. Click ‘New item’.

Property	Value
Item name	sonar-art

Property	Value
Type	Maven project

Click ‘OK’.

Configure job sonar-art Go to the Jenkins user interface. Click ‘sonar-art’, ‘Configure’.

	Property	Value
Discard old builds	Max # of builds to keep	5
Git	Repository URL	git@www.gitlab..tld:test-group/test-project.git
	Credentials	jenkins (gitlab user)
Build	Root POM	testART/pom.xml
	Goals	clean install -DskipTests
Post-build Actions	***select***	SonarQube analysis with Maven

Click ‘Save’.

6.4.10.12 Create job build-art

Go to the Jenkins user interface. Click ‘New item’.

Property	Value
Item name	build-art
Type	Maven project

Click ‘OK’.

Configure job build-art Go to the Jenkins user interface. Click ‘build-art’, ‘Configure’.

	Property	Value
Discard old builds	Max # of builds to keep	5
Git	Repository URL	git@www.gitlab..tld:test-group/test-project.git
	Credentials	jenkins (gitlab user)
Additional Behaviours		Check out to specific local branch
	Branch name	master
Build environment	Release Ggoals	-Dresume=false release:prepare release:perform -D
	DryRun goals	-Dresume=false -DdryRun=true release:prepare -D
Build	Root POM	testART/pom.xml
	Goals	clean install -DskipTests

Click ‘Save’.

6.4.10.13 Create job run-art

Go to the Jenkins user interface. Click ‘New item’.

Property	Value
Item name	run-art
Type	Maven project

Click ‘OK’.

Configure job run-art Go to the Jenkins user interface. Click ‘run-art’, ‘Configure’.

	Property	Value
Discard old builds	Max # of builds to keep	5
Parameters	***select***	This build is parametrized
	select	Add String Parameter
	Name	browserType
	Default Value	FIREFOX
	select	Add String Parameter
	Name	seleniumServerUrl
	Default Value	http://server.selenium..tld:4444/wd/hub
	select	Add String Parameter
	Name	applicationServerUrl
	Default Value	http://www.testapp..tld:8080
Git	Repository URL	git@www.gitlab..tld:test-group/test-project.git
	Credentials	jenkins (gitlab user)
Build	Root POM	testART/pom.xml
	Goals	-DbrowserType=$browserType -DseleniumServerUr

Add ‘Execute shell’ Post Step. Select ‘Run only if build succeeds’. Enter command:

```
export \
  URL="http://trr.reporting.<your-project>.tld:4567/upload" \
  APP_NAME="Testapp" \
  APP_VERSION="SNAPSHOT" \
  TEST_DESCRIPTION="ART Testapp" \
  TEST_USER="Jenkins" \
  TEST_VERSION="Master" \
  TEST_TARGET="$applicationServerUrl" \
```

```
  TEST_PLATFORM="$browserType" \
  TEST_RUN="ART" \
  DIR="testART/target/surefire-reports/junitreports"

# Parallele upload van resultaat
echo "Sending reports in ${DIR}"
echo "${APP_NAME}"
for file in $DIR/\*.xml; do
  [ -f $file ] || continue
  echo $file
done | xargs -I{} --max-procs 0 bash -c '
  curl ${URL} \
    -s \
    -F "junit=@{}" \
    -F "application_name=${APP_NAME}" \
    -F "application_version=${APP_VERSION}" \
    -F "testrun_description=${TEST_DESCRIPTION}" \
    -F "testrun_user=${TEST_USER}" \
    -F "testrun_version=${TEST_VERSION}" \
    -F "test_target=${TEST_TARGET}" \
    -F "test_platform=${TEST_PLATFORM}" \
    -F "testrun=${TEST_RUN}" \
```

6.4.10.14 Create job load-ltcs

Go to the Jenkins user interface. Click 'New item'.

Property	Value
Item name	load-ltcs
Type	Freestyle project

Click 'OK'.

Configure job load-ltcs Go to the Jenkins user interface. Click 'load-ltcs', 'Configure'.

	Property	Value
Discard old builds	Max # of builds to keep	5
Bruild Trigers	***select***	periodically
	Schedule	H 6-20 * * 1-5

Add 'Execute shell' Build Step. Enter command:

```
#!/bin/bash -ex

# JIRA importer aanroepen
data=$(curl -s http://importer.reporting.<your-project>.tld:4567/import)

if [ "$data" == 'Import completed' ]
then
  exit 0
else
  exit 1
fi
```

6.4.11 Application under development

6.4.11.1 Add application to Docker Dashboard

Go to the Docker Dashboard user interface, click 'Apps', 'New App'. Enter the following app definition:

```
name: testapp
version: latest

www:
  image: www.docker-registry.<your-project>.tld:5000/testapp:latest
  enable_ssh: true
```

Click 'Save changes' and start the application.

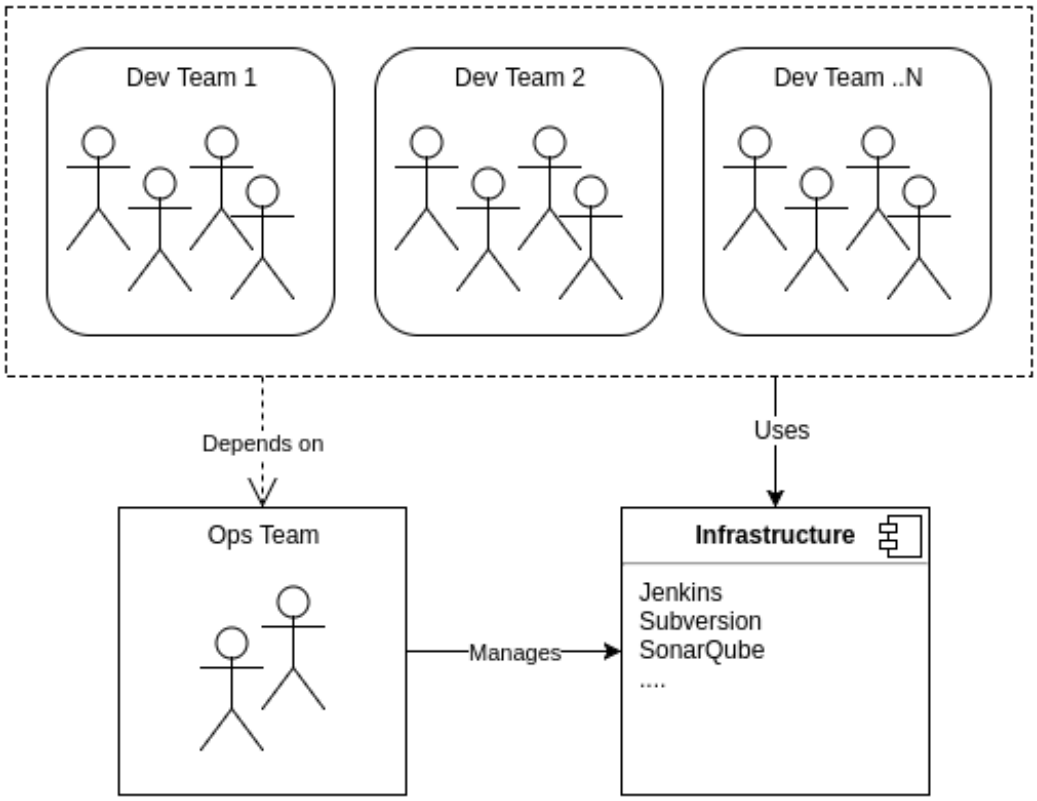

Figure 6.1: Relationship between Ops and Development teams in a shared environment

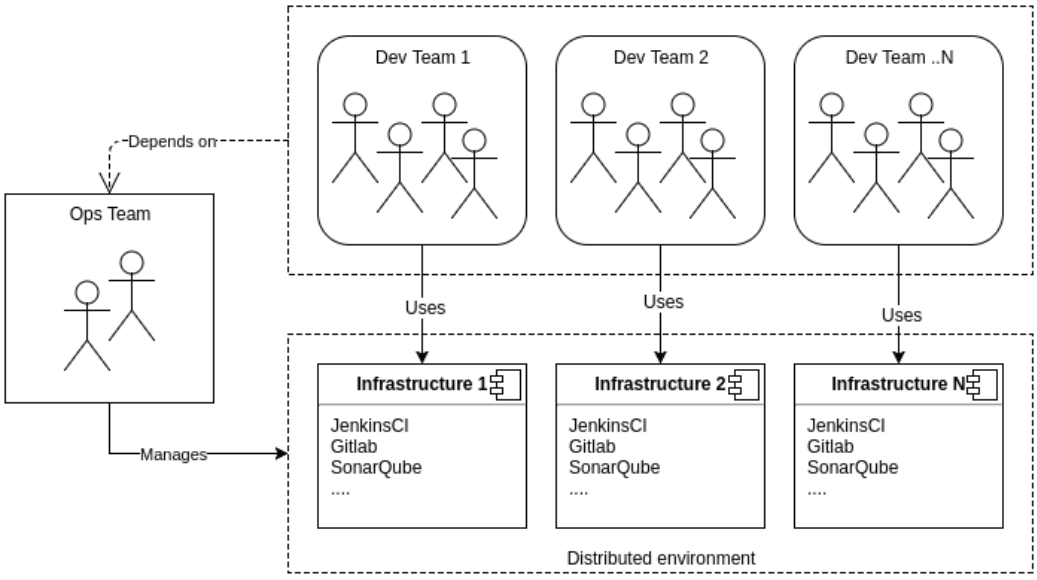

Figure 6.2: Relationship between Ops and Development teams in a distributed environment

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.

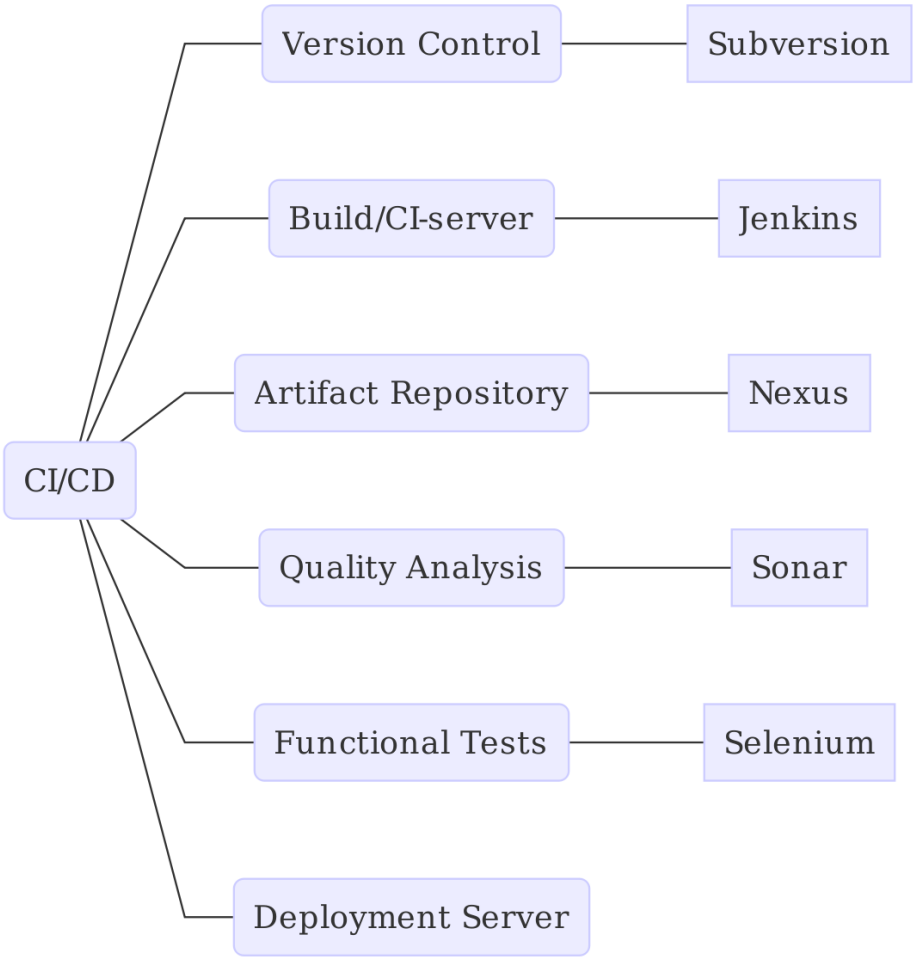

Figure 7.1: CI/CD Schematic Overview

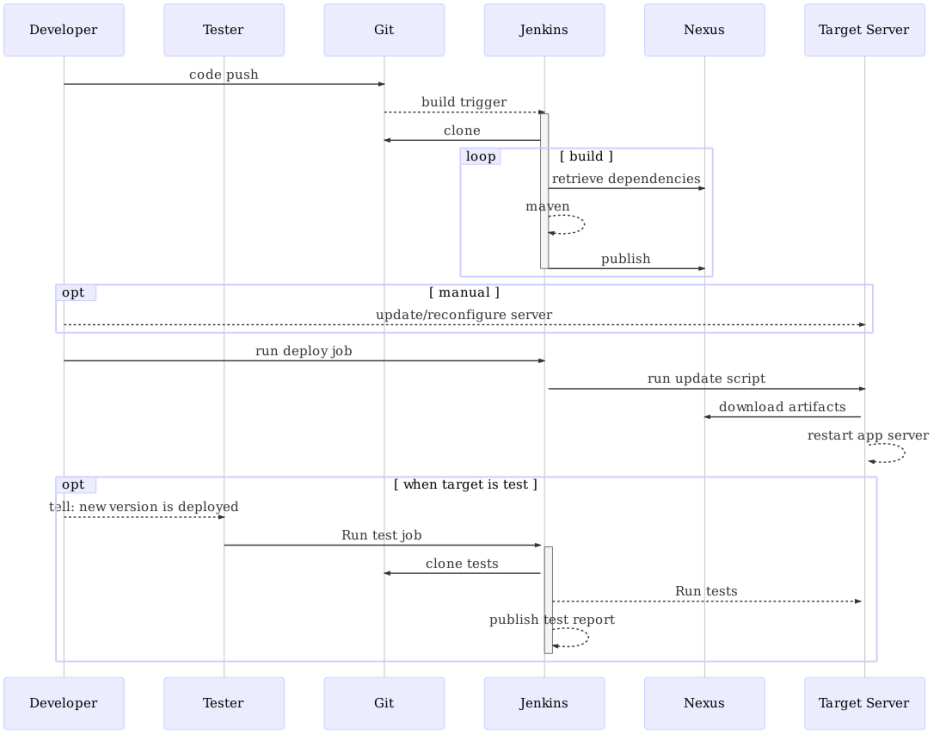

Figure 7.2: Basic CI

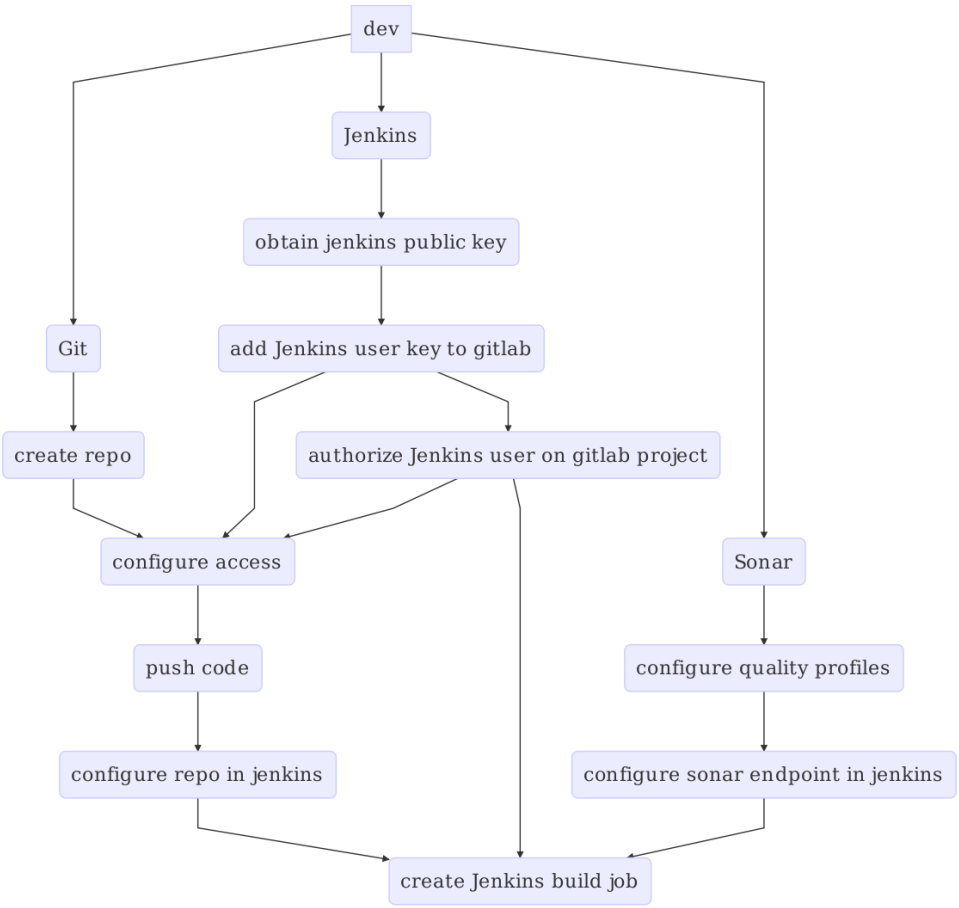

Figure 7.3: Basic CI setup

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.

25:40

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 passpoortsig-

nalering 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.