Technical Marketing Competence Center Europe
IBM Deutschland Research & Development GmbH

An automated build environment under Linux on System z

Proof of Concept Report

IBM contact person: Marc Beyerle
System z Specialist
TMCC Europe
marc.beyerle@de.ibm.com

TMCC project number: 8055

Document version: 1.0

Document date: 11/17/2009

Table of Contents

1 Executive Summary

The customer, for whom this Proof of Concept was performed, is one of Europe's largest banking institutes. With over 25,000 employees worldwide, the bank offers all regular personal banking services, general insurance products, and the full range of corporate banking services. In its home country, the bank runs one of the largest branch office networks, and it also conducts business in various other European countries and the United States.

The customer is currently very active in the area of consolidating workload to Linux® on System z®. One application in this space is a build solution, based on both Open Source and IBM® software. It is currently running under a Microsoft® Windows® operating system. Before going into production with Linux on System z, the bank wanted to see a working prototype. Therefore, a Proof of Concept project was performed at TMCC.

After a live demonstration of the environment, the bank said they were *"... impressed how supportive and professional"* TMCC was able to implement exactly what they asked for. Key to the success of this Proof of Concept was not only to fully understand and meet the customer's expectations, but also to collaborate efficiently among the various IBMers involved. In addition, this project includes an interesting message: Our customers are actively looking for ways to move applications from Microsoft Windows to the IBM Mainframe.

2 Introduction

2.1 Objectives

The objectives of this Proof of Concept project were specifically to:

- Install the software stack required for the automated build environment (see section 4.2 on page 8 for a detailed list) under Linux on System z
- Prove that the software components work under Linux on System z

The latter had to be demonstrated by successfully executing a build process test case for the sample project that comes with Apache Maven. This sample project had to be under version control of IBM Rational® ClearCase® and the build results had to be displayed in the CruiseControl web interface.

2.2 Approach

The approach for this engagement was as follows (in chronological order):

- Installation of the base Linux on System z operating system
- Installation and configuration of the Open Source and IBM software packages
- Creation of the Apache Maven sample project and associated ClearCase artifacts
- Execution of the build process test case for the sample project

2.3 People involved

IBM United Kingdom	
Garry Geokdjian	System z Technical Consultant UK & Ireland STG Sales garry_geokdjian@uk.ibm.com

IBM Ireland	
Alan Murphy	Senior IT Architect Global Technology Services alan_murphy@ie.ibm.com

IBM Boeblingen	
Marc Beyerle	System z Specialist TMCC Europe marc.beyerle@de.ibm.com

IBM Boeblingen	
Volker Masen	eServer Software Management SCM Team MASEN@de.ibm.com
Elisabeth Puritscher	System Programmer TMCC Europe elisabeth.puritscher@de.ibm.com

2.4 Location and timeline

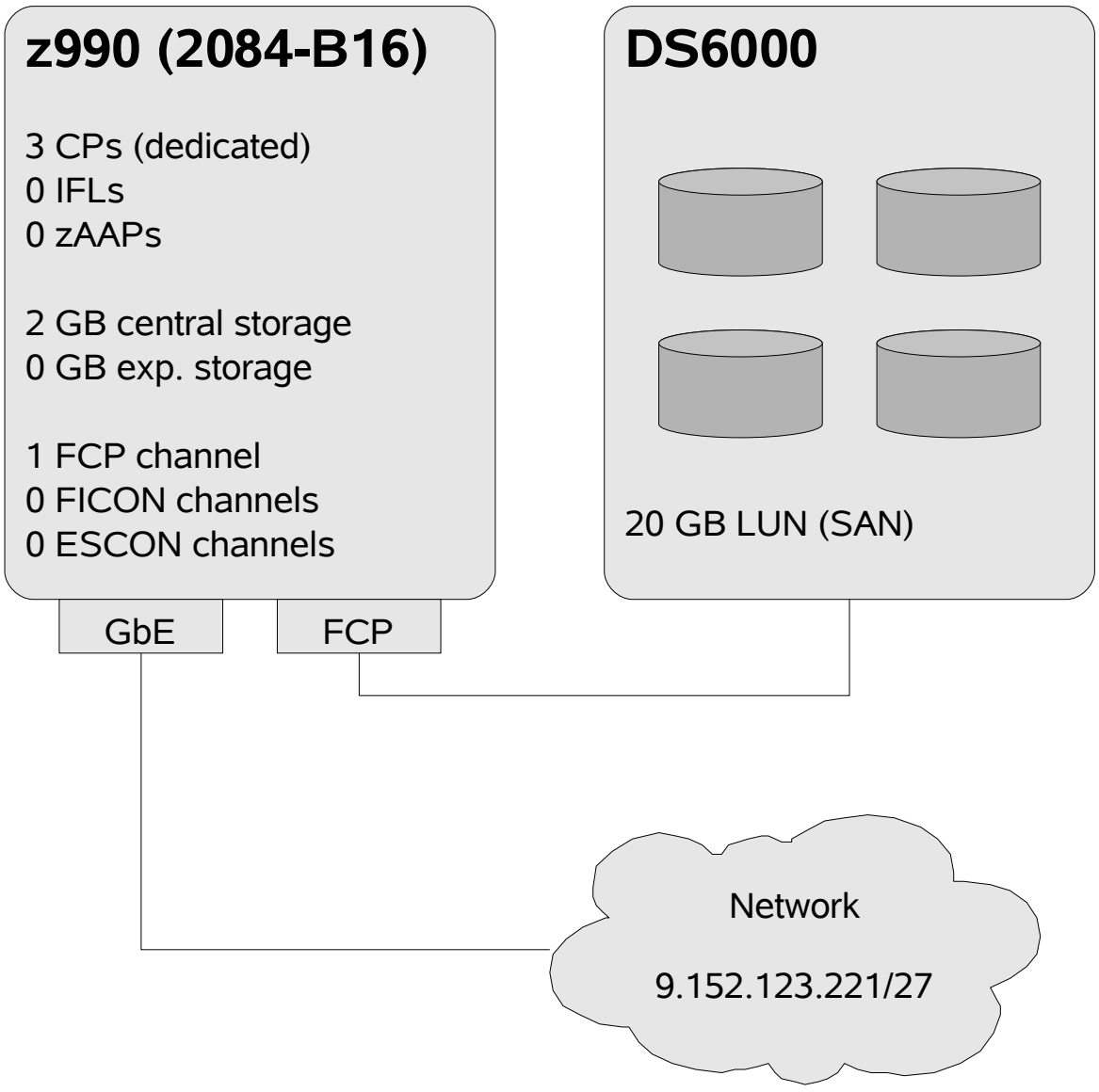
The Proof of Concept was performed in the IBM Laboratory in Boeblingen in June 2008. The results were presented remotely over the Internet using an IBM Lotus® Sametime® web conference. In July 2008, the installation was reproduced in the customer's System z environment.

3 Environment

The following sections describe the detailed hardware and software setup used during this project.

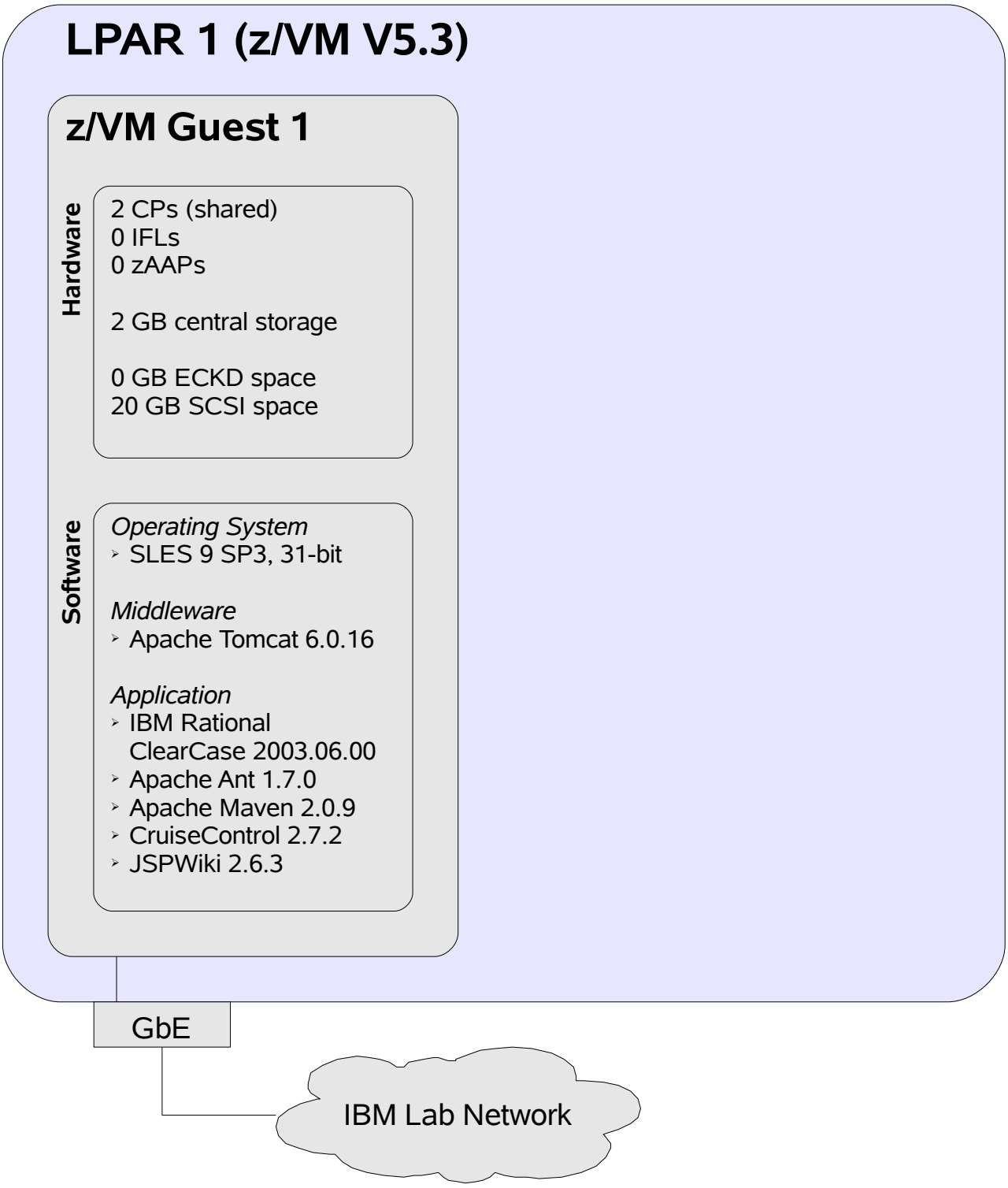
3.1 Hardware environment

The Proof of Concept was performed on an IBM eServer™ zSeries® 990, machine type 2084, model B16. See the illustration below for an overall hardware configuration overview.



3.2 Virtualized environment in LPAR 1

For this Proof of Concept, a stand-alone Linux guest operating system using a single z/VM® virtual machine had to be set up. The following diagram illustrates the hardware and software environment for this z/VM Guest.



4 Implementation

This chapter contains one section for each phase of the hardware and software setup process. The approach taken in this chapter is to list only those items, which are unique to this project and not to mention common Proof of Concept tasks.

4.1 Hardware setup

The system resources required are illustrated in section 3.1 on page 6 and section 3.2 on page 7. Note that *Logical Partition 1* (LPAR 1) was exclusively connected to the IBM Boeblingen laboratory network. Although technically possible, there was no additional connection to the Internet set up due to IBM internal security restrictions.

4.2 Software setup overview

The following list summarizes the software, which was installed for this Proof of Concept:

- Novell® SUSE® Linux Enterprise Server 9 (s390) SP3, 31-bit
- IBM Rational ClearCase 2003.06.00, with User Space Patch 55 and MVFS Patch 56
- IBM 31-bit SDK for Linux on zSeries, Java™ 2 Technology Edition, V5, SR7
- Apache Tomcat 6.0.16
- Apache Ant 1.7.0
- Apache Maven 2.0.9
- CruiseControl 2.7.2
- JSPWiki 2.6.3

4.3 31-bit versus 64-bit

As indicated in the above list, this Proof of Concept project was carried out using the 31-bit version of *SUSE Linux Enterprise Server 9* (SLES9). However, setting up the environment under the 64-bit version of SLES9 works equally well. In fact, the customer is running it in 64-bit. The only differences are:

- When running under the 64-bit version of SLES9, you may either install the 31-bit or the 64-bit version of the IBM SDK for Linux on zSeries. At the customer's site, the 64-bit version was installed.
- The installation of Rational ClearCase under 31-bit SLES9 requires two changes in installation-related scripts. These changes are not required when installing ClearCase under 64-bit SLES9.

Note that it was not an objective to conduct a performance comparison between the 31-bit and 64-bit setup during this Proof of Concept project, and therefore, no such tests were performed.

4.4 Operating system setup

Since there were no specific software package requirements with respect to the Linux operating system, SLES9 was installed using the "Minimum system" software configuration option.

However, Rational ClearCase includes the *Multiversion File System* (MVFS) kernel module. In order to be able to compile it, the Linux kernel sources and the C Compiler of the *GNU Compiler Collection* (GCC) had to be installed on top of the minimal installation. Furthermore, a name server was configured, in order to facilitate host name resolution required by both Apache Maven and Rational ClearCase.

4.5 Middleware setup

Apache Tomcat is the only software package used in this project, which can be considered middleware. Therefore, this section contains only one sub-section.

4.5.1 Apache Tomcat

The first action in installing Tomcat is to set up a Java SDK. Note that all steps in this section have to be executed under the `root` User ID:

- Remove any existing Java SDK packages using YaST
- Download the IBM Java SDK for Linux on zSeries from the IBM developerWorks web site:

  `http://www-128.ibm.com/developerworks/java/jdk/linux/download.html`
- Install the IBM Java SDK:

  ```
  rpm -ivh ibm-java2-s390-sdk-5.0-7.0.s390.rpm
  ```
- Create a symbolic link from `/opt/ibm/java2-s390-50` to `/usr/lib/java`:

  ```
  ln -s /opt/ibm/java2-s390-50 /usr/lib/java
  ```

The last step above makes the `java` executable available on the command line. Note that you have to log out and in again in order for the changes to take effect. Next, Apache Tomcat can be installed:

- Download Tomcat from:

  `http://tomcat.apache.org/download-60.cgi`
- Extract the tar archive to `/opt`:

  ```
  cd /opt
  tar xvzf [/directory/to/]apache-tomcat-6.0.16.tar.gz
  ```

- Please note that in the above notation, `[/directory/to/]` denotes the file system path to the Tomcat installation archive.
- Create a symbolic link from `/opt/apache-tomcat-6.0.16` to `/opt/tomcat`:

```
ln -s /opt/apache-tomcat-6.0.16 /opt/tomcat
```

- Create a `tomcat` group:

```
/usr/sbin/groupadd -r tomcat
```

- Create a user `tomcat` with home directory `/opt/tomcat`:

```
/usr/sbin/useradd -c "Tomcat" -g tomcat -s /bin/sh -r
   -d /opt/tomcat tomcat
```

- Recursively change ownership of `/opt/apache-tomcat-6.0.16` to user `tomcat`:

```
chown -R tomcat. /opt/apache-tomcat-6.0.16
```

This completes the base Tomcat installation. However, there are still a few configuration steps required for both CruiseControl and JSPWiki (see the corresponding sections for details).

4.6 Application setup

This section contains a sub-section for each of the application-level software packages included in this Proof of Concept.

4.6.1 Rational ClearCase

There are many different ways to set up Rational ClearCase, especially for environments with high amounts of users performing work in parallel. Please refer to the following IBM Redbooks® publication for more information on this topic:

- U. Wahli et al.: *Software Configuration Management: A Clear Case for IBM Rational ClearCase and ClearQuest UCM*. IBM Document No. SG24-6399-00, 2004.

For this particular project, ClearCase was installed as straightforward and lightweight as possible. Before the actual installation process can be started, the so-called ClearCase *Release Area* has to be created and upgraded to the required patch level. Again, all steps in this section have to be executed under the `root` User ID:

- Create the Release Area:

```
mkdir /CC_release_area
```

- Extract the ClearCase installation archive into the Release Area:

```
cd /CC_release_area
```

```
tar xvzf <name_of_the_install_archive>
```

- Download the ClearCase patches listed in 4.2 on page 8 from the following web site:

  `http://www-1.ibm.com/support/docview.wss?rs=984&uid=swg21136950`

- Follow the instructions in the `README` files of the patches step by step. First, install patch clearcase_p2003.06.00-55, then install patch clearcase_p2003.06.00-56.

During the patch process for the Release Area, the `site_prep` command has to be invoked. For this Proof of Concept project, `site_prep` was executed without any additional arguments. The command issues a number of questions, for which the answers depend on whether you want to integrate the to-be-installed ClearCase into another (existing) ClearCase environment. One of the main purposes of integrating the new ClearCase installation into an existing environment is to make reuse of existing ClearCase servers, such as registry and license servers.

For this particular project, ClearCase had not to be integrated into another ClearCase environment. Therefore, the questions were answered in the following way:

- ClearCase License Server Host: host name of the server, onto which ClearCase is installed (at TMCC, `zlx8055` was used)
- ClearCase Registry Server Host: same as above
- ClearCase Registry Region: `tmcc`

All other questions were answered with the default values provided by the `site_prep` command.

As already mentioned in 4.3 on page 8, the following additional steps are necessary for 31-bit SLES9 (not required for 64-bit):

- Open the file `/CC_release_area/2003.06.00/linux_390/clearcase/install/Kernel.pl` with an editor (for example vi)
- Locate line 371 and change it from:

  ```
  if ($os_dist_ver !~ m@[78]\.*@) {
  ```

  to:

  ```
  if ($os_dist_ver !~ m@[789]\.*@) {
  ```

- Save the file and exit the editor

If you are using vi, do the following in order to go to line 371: hit the ESC key, type a colon, enter 371 and hit the return key.

Now, the actual install process can be started:

- `cd /CC_release_area/2003.06.00/linux_390/clearcase/install`
- `./install_release`

A sequence of questions will be asked before the copying of the files commences. Here are the answers used for this Proof of Concept project:

- Method of installation: `1` (Local Install)
- Model of installation: `2` (Full-copy)
- Specify directory in which Rational products are to be installed:

  `/opt/rational`
- Pathname to the network-wide release directory:

  `/CC_release_area/2003.06.00/linux_390`
- Select `9` (ClearCase Full Function Installation)
- Select `f` (Finish selection)

After this, the questions of the `site_prep` command are repeated. They should be answered using the default values provided by the `install_release` script.

The installation process will now copy the required files. It finishes with some warning messages, which indicate that the MVFS kernel module could not be compiled. ClearCase ships with only a few pre-compiled modules for older kernel versions, so these warning messages can be safely ignored. However, since ClearCase is not fully functional without this module, you have to compile it manually:

- Change to the directory containing the source files:

  ```
  cd /var/adm/rational/clearcase/mvfs/vnode_src
  ```
- Follow the instructions in the `README.txt` file

Before you execute the *"Build and install"* step listed in the above file, make sure that you execute the following steps for 31-bit SLES9 (not required for 64-bit):

- Open the `vnode_param.mk.config` file with an editor
- Change the first line in this file from:

  ```
  RATL_EXTRAFLAGS := -DSLES8
  ```

  to:

  ```
  RATL_EXTRAFLAGS := -DSLES9
  ```
- Save the file and exit the editor

Now the path to the ClearCase executables has to be added to the system-wide `PATH` environment variable. In SLES, this is typically done by creating an additional configuration file in the `/etc/profile.d` directory. See section A.1 on page 25 for the name and contents of this file. After creating the file, log out and in again in order for the changes to take effect.

Since the ClearCase installation for this Proof of Concept is not integrated into an existing ClearCase environment, a few additional steps have to be completed in order to enable the required server functionality. Note that the following steps are not required if you integrate ClearCase into

another ClearCase environment:

- Add your IBM Rational ClearCase license key to the license database:
  - Open the file `/var/adm/atria/license.db` with an editor
  - Paste your license key at the end of this file
  - Save the file and exit the editor
- Create an encrypted ClearCase *Versioned Object Base* (VOB) tag registry password:

```
cd /opt/rational/clearcase/linux_390/etc/
./rgy_passwd
```

- Create a Linux user, who will host a VOB in his home directory:

```
useradd -m marc
```

- Switch to this User ID:

```
su - marc
```

- Create a VOB:

```
mkdir /home/marc/VOBs
cleartool mkvob -tag /vobs/mypoc -public -comment "My PoC"
  /home/marc/VOBs/mypoc.vbs
```

Next, create a so-called *View* for the above VOB. The following steps have to be completed, regardless of whether you integrate ClearCase into another ClearCase environment or not:

- Create a View:

```
mkdir /home/marc/VIEWs
cleartool mkview -tag marc_mypoc_main -snapshot
  /home/marc/VIEWs/marc_mypoc_main.vws
```

- Create a so-called *Config Spec*:

```
cd /home/marc/VIEWs/marc_mypoc_main.vws
cleartool edcs
```

- In the editor that opens, enter the following:

```
element * CHECKEDOUT
element * /main/LATEST
load /vobs/mypoc
```

- Save the file and exit the editor. This activates the new Config Spec.

- Load the actual contents of the View into the View's physical directory:

  ```
  cleartool update
  ```

The View is now in a state, where new elements such as directories and files may be added. Since the sample project that comes with Apache Maven will be put under version control, you will have to make yourself familiar with the process of registering directories and files with ClearCase. The basic flow for adding new elements is the same for both directories and files:

- Change to the directory, in which the new element will be created:

  ```
  cd /home/marc/VIEWs/marc_mypoc_main.vws/vobs/mypoc
  ```

- Check out the current directory:

  ```
  cleartool checkout .
  ```

- Create the element:
  - For directories: `mkdir <directory_name>`
  - For files: `vi <file_name>`
- Invoke the ClearCase `mkelem` command on the new element:

  ```
  cleartool mkelem <name_of_the_element>
  ```

- Check in the new element:

  ```
  cleartool checkin <name_of_the_element>
  ```

When you are done with the above steps, do not forget to check in the current directory, in which you created the new element.

4.6.2 Apache Ant

Perform the following steps in order to install Apache Ant (under the `root` User ID):

- Download Ant from:

  http://ant.apache.org/bindownload.cgi

- Extract the tar archive to `/opt`:

  ```
  cd /opt
  ```

  ```
  tar xvzf [/directory/to/]apache-ant-1.7.0-bin.tar.gz
  ```

- Please note that in the above notation, `[/directory/to/]` denotes the file system path to the Ant installation archive.
- Create a symbolic link from `/opt/apache-ant-1.7.0` to `/opt/ant`:

  ```
  ln -s /opt/apache-ant-1.7.0 /opt/ant
  ```

Identical to Rational ClearCase, the path to the Ant executables has to be added to the system-wide `PATH` environment variable. See section A.2 on page 25 for the name and contents of the corresponding configuration file. After creating the file, log out and in again in order for the changes to take effect.

4.6.3 Apache Maven

Perform the following steps in order to install Apache Maven (under the `root` User ID):

- Download Maven from:

  `http://maven.apache.org/download.html`

- Extract the tar archive to `/opt`:

  ```
  cd /opt
  tar xvzf [/directory/to/]apache-maven-2.0.9-bin.tar.gz
  ```

- Please note that in the above notation, `[/directory/to/]` denotes the file system path to the Maven installation archive.

- Create a symbolic link from `/opt/apache-maven-2.0.9` to `/opt/maven`:

  ```
  ln -s /opt/apache-maven-2.0.9 /opt/maven
  ```

Identical to the last software packages, the path to the Maven executables has to be added to the system-wide `PATH` environment variable. See section A.3 on page 26 for the name and contents of the corresponding configuration file. After creating the file, log out and in again in order for the changes to take effect.

Maven requires Internet access, since it downloads numerous plugins when executed for the first time. At the customer site, it was necessary to configure an HTTP proxy for Maven. This is achieved by creating a Maven *User Configuration*:

- Switch to the User ID, who will execute the build process:

  ```
  su - marc
  ```

- Create a Maven directory in the user's home directory:

  ```
  mkdir .m2
  ```

- Create a Maven user configuration with the file name and contents listed in A.4 on page 26

4.6.4 CruiseControl

Perform the following steps in order to install CruiseControl (under the `root` User ID):

- Download CruiseControl from:

  `http://cruisecontrol.sourceforge.net/download.html`

- Extract the ZIP archive to `/opt`:

  ```
  cd /opt

  unzip -d . [/directory/to/]cruisecontrol-src-2.7.2.zip
  ```

- Please note that in the above notation, `[/directory/to/]` denotes the file system path to the CruiseControl installation archive.

- Create a symbolic link to `/opt/cruisecontrol`:

  ```
  ln -s /opt/cruisecontrol-2.7.2 /opt/cruisecontrol
  ```

- Compile the main CruiseControl components:

  ```
  cd /opt/cruisecontrol/main

  sh build.sh
  ```

- Compile the reporting component:

  ```
  cd /opt/cruisecontrol/reporting/jsp

  sh build.sh
  ```

During the compilation process of the reporting component, you have to enter a few values for reporting-related Java properties. Here are the values used for this Proof of Concept project:

- `user.log.dir = /work/cruise/logs`
- `user.build.status.file = status.txt`
- `cruise.build.artifacts.dir = /work/cruise/artifacts`

After compiling the reporting component, install it under Tomcat:

- Switch to the `tomcat` User ID:

  ```
  su - tomcat
  ```

- Copy the Web Archive to the corresponding Tomcat directory:

  ```
  cp /opt/cruisecontrol/reporting/jsp/dist/cruisecontrol.war
    /opt/tomcat/webapps
  ```

Identical to the last software packages, the path to the CruiseControl executables has to be added to the system-wide `PATH` environment variable. See section A.5 on page 27 for the name and contents of the corresponding configuration file. After creating the file, log out and in again in order for the changes to take effect.

Different to the other software packages, the executable bit for the main CruiseControl executable has to be set. As the `root` user, do the following:

- `chmod 755 /opt/cruisecontrol/main/bin/cruisecontrol.sh`

4.6.5 JSPWiki

Although not directly related to the build environment, JSPWiki was installed in order to demonstrate proper operation under Linux on System z. No specific adjustments were made, therefore you can simply follow the installation guide at:

- http://doc.jspwiki.org/2.4/wiki/InstallingJSPWikiStepByStep

In section *"Configuring the main policy file"*, the author mentions the setting of the `CATALINA_OPTS` environment variable. This is best done by creating an extra configuration file in the `/opt/tomcat/bin` directory. For the name and contents of this file, see section A.6 on page 27. Note that this is a general best practice, and not specific to Linux on System z.

5 Test case

In order to prove that the software components work under Linux on System z, a test case had to be agreed upon for this project. It was decided that the test case should consist of a two-step build process for the sample project that comes with Apache Maven. This chapter describes the setup and execution of this test case.

5.1 Definition

The test case consists of two steps:

- Perform an initial build of the Maven sample project. This is done by invoking the CruiseControl executable with an empty build output directory.
- Perform a build after one arbitrary file in the source directory has been changed. This is done by performing a checkout on the file, a modification of the file, and finally a checkin of the file. Note that CruiseControl picks up the changes automatically.

The test case is considered successful, if the two operations described above complete without errors and the build results are displayed in the CruiseControl web interface.

5.2 Approach

The approach for setting up the test case is as follows:

- Create a ClearCase VOB and an associated View for the source files. This has already been done (see page 13 for details).
- Create the Maven sample project in the View's directory
- Configure CruiseControl
- Perform the initial build
- Perform a build after a source file change

The following sub-sections describe the above steps in more detail.

5.2.1 Creating the Maven sample project

Perform the following steps in order to create the Maven sample project:

- Switch to the User ID, who will execute the build process:

  ```
  su - marc
  ```

- Change the current directory to the View:

  ```
  cd /home/marc/VIEWs/marc_mypoc_main.vws/vobs/mypoc
  ```

- Create the Maven sample project:

```
mvn archetype:create -DgroupId=com.ibm.tmcc.poc
   -DartifactId=MavenPoC
```

The last command will create a directory called `MavenPoC` in the current directory. It contains the so-called *Project Object Model* (POM) file called `pom.xml` and the directory for the source files `src`. Now you have to follow the process described on page 14, in order to register all directories and files under (and including) `MavenPoc` to ClearCase.

5.2.2 Configuring CruiseControl

You can read more about this task in the *"Getting Started with the Source Distribution"* guide, which can be found here:

- http://cruisecontrol.sourceforge.net/gettingstartedsourcedist.html

Perform the following steps (under the `root` User ID) in order to set up the so-called *Working Area*, a set of directories used by CruiseControl:

- Create the tree structure:

```
mkdir -p /work/cruise
mkdir /work/cruise/artifacts
mkdir /work/cruise/checkout
mkdir /work/cruise/logs
```

- Recursively change ownership of `/work/cruise` to the User ID, who will execute the build process:

```
chown -R marc. /work/cruise
```

Now you have to configure the *Build Loop*, which is (citing the above guide) *"... a Java application that provides the core build scheduling functionality of CruiseControl"*. Part of the Build Loop execution process is to get the latest version of the source files from the repository. There are two ways to accomplish this when using CruiseControl together with the combination of Maven and ClearCase:

- Maven fetches the source file updates from ClearCase as part of the POM file
- An Ant script fetches the source file updates

For this particular Proof of Concept, the second solution was implemented. There are two reasons for this: First, the Ant approach provides a more fine granular control over the source file fetch process. Second, the Ant approach is well documented, as opposed to using ClearCase from within Maven. See section A.7 on page 28 for the name and contents of the Ant script file, which is used as a so-called *Delegating Build Script*.

Now the actual Build Loop can be configured. The guide mentioned above provides a detailed description of the individual sections of this file, so this information is omitted in this report. Refer

to section A.8 on page 29 for the name and contents of the Build Loop configuration file. You will notice that the `<schedule>` element contains a `<composite>` element, which is different to the sample configuration provided in the guide. This is due to the source file update method described in the last paragraph. The first XML child element of the `<composite>` element calls the Ant script, and the second child element starts the Maven build process. Another difference is the `<merge>` element underneath the `<log>` element. This is required to include the output of the JUnit tests into the CruiseControl web interface.

Before the Build Loop can be started, a few file system adjustments have to be made:

- Switch to the User ID, who will execute the build process:

  ```
  su - marc
  ```

- Create a symbolic link to the ClearCase View:

  ```
  cd /work/cruise/checkout
  ln -s /home/marc/VIEWs/marc_mypoc_main.vws/vobs/mypoc/MavenPoC .
  ```

- Create a directory for the build log files:

  ```
  cd /work/cruise/logs
  mkdir MavenPoC
  ```

- Create a caching directory for the CruiseControl web application:

  ```
  cd /work/cruise/logs/MavenPoC
  mkdir _cache
  chmod 777 _cache
  ```

The last command will write-enable the caching directory for all users. This is necessary, since the Tomcat User ID has to be able to write to it.

5.2.3 Perform the initial build

In order to perform the initial build, complete the following steps:

- If not already done, switch to the User ID, who will execute the build process:

  ```
  su - marc
  ```

- Change to the directory containing the Working Area:

  ```
  cd /work/cruise
  ```

- Start the Build Loop:

  ```
  cruisecontrol.sh
  ```

Here are some tips, for the case when the Build Loop cannot be started or returns with a `"build failed"` error message:

- If the `cruisecontrol.sh` script cannot be found on the command line, you probably forgot to configure the profile extension for CruiseControl (see section 4.6.4 on page 15). In addition, make sure that the executable bit for the `cruisecontrol.sh` script is set.
- If you get an error similar to

  ```
  cleartool: Error: Not a vob object: ".".
  ```

  you probably forgot to register the sample project's directories and files to ClearCase (see page 14 for details on how to do this).
- If you get an error similar to

  ```
  cleartool: Error: Unable to access "...": No such file or directory.
  ```

  check the quotes around the View path name in the Delegating Build Script.

5.2.4 Perform a build after a source file change

In order to trigger an automatic build, you have to change a source file, which is under the control of ClearCase. For the next steps, open another shell on your server:

- If not already done, switch to the User ID, who will execute the build process:

  ```
  su - marc
  ```
- Change to the directory of the Maven sample project:

  ```
  cd /home/marc/VIEWs/marc_mypoc_main.vws/vobs/mypoc/MavenPoC
  cd src/main/java/com/ibm/tmcc/poc/
  ```
- Check out the Java source file contained in this directory:

  ```
  cleartool checkout App.java
  ```
- Open the file with an editor
- Locate line 11 and change it from:

  ```
  System.out.println( "Hello World!" );
  ```

  to something similar to:

  ```
  System.out.println( "Hello Linux on System z World!" );
  ```
- Save the file and exit the editor
- Check in the Java source file:

  ```
  cleartool checkin App.java
  ```

If everything is configured correctly, CruiseControl will pick up the changes automatically and trigger a new build process.

5.3 Results

In order to check whether the builds were executed successfully, you can check both the output messages of CruiseControl on the command line and the web interface, which can be found here:

- `http://<your_host_name>:8080/cruisecontrol`

If you click on the `MavenPoc` link, you will see all kinds of statistics regarding the build operations. For this Proof of Concept project, both builds were executed successfully.

6 Conclusion and outlook

Having successfully completed this Proof of Concept project together with IBM, the customer has made its next step towards a new data center built upon the strengths of Linux on System z.

Utilizing the capabilities of the underlying virtualization technology, the bank can now start to benefit from the centralized server management that comes with z/VM. Other benefits are the simplified networking structures based on System z HiperSockets™ and the ability to provision new servers in a matter of minutes. Finally, the possibility of taking the former Microsoft Windows system out of service adds to the "green" aspect of this project. Consolidating workload on the IBM Mainframe means less overall power consumption and less floor space required. The more physical servers are consolidated, the greener the solution will be in the end.

With its unique combination of System z hardware, software, and technical consulting skills, TMCC was able to contribute its share to the success of this project. In addition, the knowledge transfer to both customer specialists and the local IBM team will help to ensure that the client is well positioned for future projects in the Linux on System z space.

If you want more information regarding this project or if you need technical assistance in an IBM System z related Proof of Concept, do not hesitate to contact the author Marc Beyerle (marc.beyerle@de.ibm.com). Alternatively, you may also contact the Technical Marketing Competence Center Europe at tmcc@de.ibm.com

A Additional material

All shell scripts and configuration files in this chapter were written for and tested under *SUSE Linux Enterprise Server 9* (SLES9) SP3, s390 (31-bit) and s390x (64-bit). Nevertheless, please read the following disclaimer for the scripts and configuration files provided in this appendix:

Disclaimer of warranties

The shell scripts and configuration files in this document are created by IBM. The shell scripts and configuration files are provided to you solely for internal usage only. The shell scripts and configuration files are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the shell scripts and configuration files, even if they have been advised of the possibility of such damages.

A.1 Profile extension for ClearCase

File name:

`/etc/profile.d/clearcase.sh`

File contents:

```
#
# ------------------------------------------------------------------
# |   CLEARCASE ENVIRONMENT SETTINGS                               |
# |                                                                |
# |   Author: Marc Beyerle (marc.beyerle@de.ibm.com), 06/17/2008   |
# |   Copyright IBM Corp. 2008                                     |
# ------------------------------------------------------------------
#
export PATH=${PATH}:/opt/rational/clearcase/linux_390/bin
```

A.2 Profile extension for Ant

File name:

`/etc/profile.d/ant.sh`

File contents:

```
#
# ------------------------------------------------------------------
# |   ANT ENVIRONMENT SETTINGS                                     |
# |                                                                |
# |   Author: Marc Beyerle (marc.beyerle@de.ibm.com), 06/18/2008   |
# |   Copyright IBM Corp. 2008                                     |
# ------------------------------------------------------------------
#
export ANT_HOME=/opt/ant
export PATH=${PATH}:${ANT_HOME}/bin
```

A.3 Profile extension for Maven

File name:

```
/etc/profile.d/maven.sh
```

File contents:

```
#
# ------------------------------------------------------------------
# |   MAVEN ENVIRONMENT SETTINGS                                   |
# |                                                                |
# |   Author: Marc Beyerle (marc.beyerle@de.ibm.com), 06/18/2008   |
# |   Copyright IBM Corp. 2008                                     |
# ------------------------------------------------------------------
#
export M2_HOME=/opt/maven
export PATH=${PATH}:${M2_HOME}/bin
```

A.4 Maven proxy configuration

File name:

```
${HOME}/.m2/settings.xml
```

File contents:

```
<settings>
  <proxies>
   <proxy>
      <active>true</active>
      <protocol>http</protocol>
      <host>proxy.somewhere.com</host>
      <port>8080</port>
      <username>proxyuser</username>
      <password>somepassword</password>
    </proxy>
  </proxies>
</settings>
```

Note: This configuration file was taken from the Maven documentation on the Internet, which can be found here:

- http://maven.apache.org/guides/mini/guide-proxies.html

A.5 Profile extension for CruiseControl

File name:

```
/etc/profile.d/cruisecontrol.sh
```

File contents:

```
#
# ---------------------------------------------------------------------
# |   CRUISECONTROL ENVIRONMENT SETTINGS                              |
# |                                                                   |
# |   Author: Marc Beyerle (marc.beyerle@de.ibm.com), 06/17/2008      |
# |   Copyright IBM Corp. 2008                                        |
# ---------------------------------------------------------------------
#
export PATH=${PATH}:/opt/cruisecontrol/main/bin
```

A.6 Tomcat environment configuration for JSPWiki

File name:

```
/opt/tomcat/bin/setenv.sh
```

File contents:

```
#
# ---------------------------------------------------------------------
# |   JSPWIKI ENVIRONMENT SETTINGS                                    |
# |                                                                   |
# |   Author: Marc Beyerle (marc.beyerle@de.ibm.com), 06/19/2008      |
# |   Copyright IBM Corp. 2008                                        |
# ---------------------------------------------------------------------
#
export CATALINA_OPTS="-Djava.security.policy=/etc/wikis/jspwiki.policy"
```

A.7 Delegating Build Script for CruiseControl

File name:

```
/work/cruise/build-MavenPoC.xml
```

File contents:

```
<!-- Delegating build script, used by cruisecontrol to build MavenPoC.
     Note that the basedir is set to the checked out project -->
<project name="build-MavenPoC"
        default="build"
        basedir="checkout/MavenPoC">
    <target name="build">
        <!-- Get the latest from ClearCase -->
        <ccupdate viewpath="/home/marc/VIEWs/marc_mypoc_main.vws/
            vobs/mypoc/MavenPoC/" graphical="false"
            log="/work/cruise/logs/MavenPoC/clearcase.log"
            overwrite="true" rename="false"/>
    </target>
</project>
```

Important note: In the above listing, the full path name behind the `viewpath` attribute has been split, in order to make it readable. However, when creating your own copy of this file, do not split it. Otherwise, the build process will fail during the source file update operation.

Additional note: This configuration file was adapted from the CruiseControl documentation on the Internet, which can be found here:

- http://cruisecontrol.sourceforge.net/gettingstartedsourcedist.html

A.8 Build Loop configuration for CruiseControl

File name:

```
/work/cruise/build-MavenPoC.xml
```

File contents:

```
<cruisecontrol>
  <project name="MavenPoC" buildafterfailed="true">
    <listeners>
      <currentbuildstatuslistener
          file="logs/MavenPoC/status.txt"/>
    </listeners>

    <!-- Bootstrappers are run every time the build runs,
         *before* the modification checks -->
    <bootstrappers>
    </bootstrappers>

    <!-- Defines where cruise looks for changes, to decide
         whether to run the build -->
    <modificationset quietperiod="10">
      <clearcase branch="main" viewpath="checkout/MavenPoC"/>
    </modificationset>

    <!-- Configures the actual build loop, how often and which
         build file/target -->
    <schedule interval="60">
      <composite>
        <ant anthome="/opt/ant"
             buildfile="build-MavenPoC.xml"
             target="build"
             uselogger="true"
             usedebug="false"/>
        <maven2 mvnhome="/opt/maven"
                pomfile="checkout/MavenPoC/pom.xml"
                goal="clean compile test"/>
      </composite>
    </schedule>

    <!-- directory to write build logs to -->
    <log logdir="logs/MavenPoC">
       <merge dir="checkout/MavenPoC/target/surefire-reports"/>
    </log>

    <!-- Publishers are run *after* a build completes -->
    <publishers>
    </publishers>
  </project>
</cruisecontrol>
```

Note: This configuration file was adapted from the CruiseControl documentation on the Internet, which can be found here:

- http://cruisecontrol.sourceforge.net/gettingstartedsourcedist.html

IBM Deutschland Research & Development GmbH
Technical Marketing Competence Center Europe
Department 3300
Schoenaicher Str. 220
71032 Boeblingen
Germany