ACME Platform - Software Transfer Document

Tom Cruise and Brad Pitt

ACME Corp.

November 29, 2013

Abstract

This document describes the Transfer Phase activities for the ACME Platform.

Contents

1 Conventions **3**

2 Changelog **4**
2.1 Release date: 20130705 4
2.2 Release date: 20130704 4

3 Environment setup **5**
3.1 Ubuntu Linux Server 5
3.2 Java Development Kit 6
3.3 Simple Build Tool 6
3.3.1 Installation 6
3.3.2 Configuration 7
3.4 Apache Maven 7
3.4.1 Installation 7
3.4.2 Configuration 8
3.5 Git 8
3.5.1 Installation 8
3.5.2 Configuration 9
3.6 MySQL 9
3.6.1 Installation 9
3.6.2 Configuration 9
3.7 Node.js 9
3.7.1 Installation 9
3.8 Further notes 10

4 Building **11**
4.1 Space Rocket 11
4.1.1 Getting the source code 11
4.1.2 Checking out a given version 11
4.1.3 Build 11

5 Deployment **12**
5.1 Example deployment scenario 12
5.2 Space Rocket 12
5.2.1 Installation 12
5.2.2 Configuration 12
5.2.3 Running 12

6 Configuration **13**
6.1 Logging 13

7 Running **14**
7.1 No Hang Up 14
7.2 Console Window Managers 14

8 Appendix **15**
8.1 Acronyms used throughout the document 15

1 Conventions

Commands to be executed on the target system are represented using the convention in code listing 1.

Listing 1: Command line example as a regular user

```
$ id
uid=1000(acme) gid=1000(acme) groups=1000(acme),4(adm),27(sudo)
```

The `$` prompt symbol is the common convention for commands to be entered at the command line prompt by a **regular user** and is not supposed to be literally typed on the keyboard.

Listing 2: Command line example as the root user

```
# id
uid=0(root) gid=0(root) groups=0(root)
```

The `#` prompt symbol is the common convention for commands to be entered at the command line prompt by the **root user** and is also not supposed to be literally typed on the keyboard. An example is given in code listing 2.

On Ubuntu systems, it is usually possible to run commands as the root user from a regular user account by prefixing them with the `sudo` command[1]. An example is given in code listing 3.

Listing 3: Command line example as the root user using sudo

```
$ sudo id
uid=0(root) gid=0(root) groups=0(root)
```

[1]the password sudo asks for is the password of the regular user you used to log in

2 Changelog

2.1 Release date: 20130705

- Space-rocket v1.0.3.5
  - fix: plutonium containment code was leaking abstractions (commit: 2665b1fd)

2.2 Release date: 20130704

- Space-rocket v1.0.3.4 (**broken**)
  - fix: orbit handling code was freezing in ionosphere (commit: 7fedc183)
  - introduced `orbit-testing` profile, copied over all already used properties
  - prettified property files, adding more appropriate comments

3 Environment setup

This section describes how to setup the system and the tools that make up the environment where the ACME Platform (AP) software is hosted.

The components that we will be installing in this section are:

- Ubuntu Linux Server 13.04
- Java Development Kit (JDK) 1.7.0 update 21
- Simple Build Tool (SBT) 0.12.3
- Apache Maven 3.0.5
- Git 1.8.1.2
- MySQL 5.5
- Neo4j 1.9
- NodeJS 0.10.5

For each component we are indicating a minimum required version number, although we suggest grabbing the latest available versions at the time the installation is performed. If you are unsure about specific versions compatibility, just email us at `rocket-surgeons@acme.com`.

Ubuntu and the JDK are required to build and run the whole platform while SBT, Apache Maven and git are only required to build the software. Additionally, Node.js is required to run the targeting user interface software module.

In a hardened deployment setup, you might want to consider having a dedicated compiler machine with all of the above environment tools set up, then having different production machines with just the JDK installed to actually run the software.

3.1 Ubuntu Linux Server

A UNIX based Operating System (OS) is required. We recommend Ubuntu Linux Server 13.04 for 64 bit machines since all of the following examples have been tested on it, although they should work with little or no adaptation on any Debian based Linux distribution.

The command line examples should run on pretty much any UNIX variant, including Berkeley Software Distribution (BSD) based systems such as Mac OS X. Distribution specific package managers[2] need of course to be used in such case.

Ubuntu Linux Server might be downloaded from the official Ubuntu download site[3]. The file we used is called `ubuntu-13.04-server-amd64.iso`.

For instructions on how to install and configure Ubuntu Linux Server, please refer to the official online Ubuntu documentation[4].

In our examples, we will refer to a regular user called `acme` and assume all work to be done in her environment and home directory. You can of course use a different user, as long as all work is done in that user's environment. To add a new regular user and set her password, please follow the example in code listing 4.

Listing 4: Adding a regular user with the `acme` login

```
# useradd --groups sudo --create-home acme
# passwd acme
```

We assume all binary package downloads to be placed in the `$HOME/dist` directory and the presence of a `$HOME/local` directory to install the downloaded packages. To create these directories, please follow the example in code listing 5.

[2]such as `yum`, `rpm`, `portage`, `brew`, etc.

[3]`http://www.ubuntu.com/download/server`

[4]`https://help.ubuntu.com/`

Listing 5: Creating the `dist` and `local` directories

```
$ mkdir ~/dist
$ mkdir ~/local
```

3.2 Java Development Kit

The JDK must be downloaded from Oracle's web site. The Linux x64 tarball[5] is the one we will be using[6]. To unpack the tarball and verify its contents, please follow the example in code listing 6.

Listing 6: Unpacking the tarball to the `local` directory and checking its contents

```
$ tar xz -C ~/local -f ~/dist/jdk-7u21-linux-x64.tar.gz
$ ls ~/local
jdk1.7.0_21
$ ls ~/local/jdk1.7.0_21
bin         lib          src.zip
COPYRIGHT   LICENSE      THIRDPARTYLICENSEREADME-JAVAFX.txt
db          man          THIRDPARTYLICENSEREADME.txt
include     README.html
jre         release
```

To make the java tools available to the command line prompt, the bottom of your `$HOME/.profile` file should look as in code listing 7.

Listing 7: Code to add to the bottom of the `$HOME/.profile` file

```
JAVA_HOME="$HOME/local/jdk1.7.0_21"
PATH="$JAVA_HOME/bin:$PATH"
```

Having modified the `$HOME/.profile` file, run the commands in code listing 8 to make the changes available to the current prompt and to verify that the java bytecode interpreter works.

Listing 8: Sourcing the `$HOME/.profile` file and verifying the java interpreter

```
$ source ~/.profile
$ java -version
java version "1.7.0_21"
Java(TM) SE Runtime Environment (build 1.7.0_21-b11)
Java HotSpot(TM) 64-Bit Server VM (build 23.21-b01, mixed mode)
```

3.3 Simple Build Tool

3.3.1 Installation

The SBT tarball should be downloaded from the official download page[7]. At the time of writing, the current version is 0.12.3.

To unpack the distribution tarball and check its contents, please run the commands in code listing 9.

Listing 9: Unpacking the tarball to the `local` directory and checking its contents

```
$ tar xz -C ~/local -f ~/dist/sbt.tgz
$ ls ~/local
jdk1.7.0_21  sbt
$ ls ~/local/sbt
bin  jansi-license.txt
```

Modify the `$HOME/.profile` file so that its bottom now looks like the example in code listing 10.

[5]a `tar.gz` file.
[6]at the time of writing, the current tarball is a file called `jdk-7u21-linux-x64.tar.gz`.
[7]`http://www.scala-sbt.org/download.html`

Listing 10: Code to add to the bottom of the `.profile` file

```
JAVA_HOME="$HOME/local/jdk1.7.0_21"
SBT_HOME="$HOME/local/sbt"
PATH="$SBT_HOME/bin:$JAVA_HOME/bin:$PATH"
```

Having modified the `$HOME/.profile` file, run the commands in code listing 11 to make the changes available to the current prompt and to verify that SBT works:

Listing 11: Sourcing the `.profile` file and verifying SBT works

```
$ source ~/.profile
$ sbt --version
sbt launcher version 0.12.3
```

3.3.2 Configuration

Make sure your shell environment defines an `SBT_OPTS` variable as shown in the example in code listing 12.

Listing 12: Verifying SBT options

```
$ echo $SBT_OPTS
-XX:+CMSClassUnloadingEnabled -XX:MaxPermSize=256M
```

This can be achieved by adding the command as in code listing 13 to your `$HOME/.profile`.

Listing 13: Setting SBT options in the `.profile` file

```
SBT_OPTS='-XX:+CMSClassUnloadingEnabled -XX:MaxPermSize=256M'
```

Authentication needs to be configured in order to allow the SBT resolver to use the nexus repository manager. An `$HOME/.ivy2/.auth-acme` configuration file must be created with contents such as in code listing 14. You should of course overwrite the `user` and `password` fields with sensible values.

Listing 14: Setting SBT authentication options in the `$HOME/.ivy2/.auth-acme` file

```
realm=Sonatype Nexus Repository Manager
host=nexus.acme.com
user=username
password=secret
```

3.4 Apache Maven

3.4.1 Installation

The Apache Maven tarball should be downloaded from the official download page[8]. At the time of writing, the current version is 3.0.5.

To unpack the distribution tarball and verify its contents, please run the commands in code listing 15.

Listing 15: Unpacking the tarball to the `local` directory and checking its contents

```
$ tar xz -C ~/local -f ~/dist/apache-maven-3.0.5-bin.tar.gz
$ ls ~/local
apache-maven-3.0.5  jdk1.7.0_21  sbt
$ ls ~/local/apache-maven-3.0.5
bin  boot  conf  lib  LICENSE.txt  NOTICE.txt  README.txt
```

[8]`http://maven.apache.org/download.cgi`

Modify the `$HOME/.profile` file so that its bottom looks like the example in code listing 16.

Listing 16: Code to add to the bottom of the `.profile` file

```
JAVA_HOME="$HOME/local/jdk1.7.0_21"
SBT_HOME="$HOME/local/sbt"
MAVEN_HOME="$HOME/local/apache-maven-3.0.5"
PATH="$MAVEN_HOME/bin:$SBT_HOME/bin:$JAVA_HOME/bin:$PATH"
```

Having modified the `.profile` file, run the commands in code listing 17 to make the changes available to the current prompt and to verify that maven works.

Listing 17: Sourcing the `.profile` file and verifying maven works

```
$ source ~/.profile
$ mvn --version
Apache Maven 3.0.5 (r01de14724cdef164cd33c7c8c2fe155faf9602da;
Maven home: /home/acme/local/apache-maven-3.0.5
Java version: 1.7.0_21, vendor: Oracle Corporation
Java home: /home/acme/local/jdk1.7.0_21/jre
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "3.8.0-19-generic", arch: "amd64"
```

3.4.2 Configuration

A `$HOME/.m2/settings.xml` file must be created with contents such as in code listing 18. The `username` and `password` fields in the `server` blocks must of course be edited with your user's specific correct values.

Listing 18: An example `$HOME/.m2/settings.xml` file

```
<?xml version="1.0" encoding="UTF-8"?>
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
    http://maven.apache.org/xsd/settings-1.0.0.xsd">
    <activeProfiles>
        <activeProfile>acme-nexus</activeProfile>
    </activeProfiles>
    <profiles>
        <profile>
            <id>acme-nexus</id>
            <repositories>
                <repository>
                    <id>acme-nexus-mirror</id>
                    <url>http://nexus.acme.com/content/groups/public/</url>
                </repository>
            </repositories>
        </profile>
    </profiles>
    <mirrors>
        <mirror>
            <id>acme-nexus-mirror</id>
            <name>ACME Nexus Mirror</name>
            <url>http://nexus.acme.com/content/groups/public/</url>
            <mirrorOf>external:*</mirrorOf>
        </mirror>
    </mirrors>
    <servers>
        <server>
            <id>acme-nexus-mirror</id>
            <username>username</username>
            <password>password</password>
        </server>
    </servers>
</settings>
```

3.5 Git

3.5.1 Installation

To install git and verify it's been correctly installed, please run the commands in code listing 19.

Listing 19: Installing git and verifying it works

```
# apt-get install git
$ git --version
git version 1.8.1.2
```

3.5.2 Configuration

Create an `$HOME/.netrc` file with contents as in the example in code listing 20. You must of course enter your user's specific correct values in the `login` and `password` fields.

Listing 20: An example `$HOME/.netrc` file

```
machine scm.acme.com
login username
password secret
```

3.6 MySQL

3.6.1 Installation

To install MySQL server and the command line client, please run the commands in code listing 21.

Listing 21: Installing MySQL

```
# apt-get install mysql-server
```

During the installation process, you will be asked for a root password: this is the password that will be used for the MySQL admin user and has nothing to do with the OS root password.

To connect to MySQL server with the root user and to verify that the installation was successful, run the commands in code listing 22. You should be asked for the root password, then see the MySQL prompt.

Listing 22: Connecting to MySQL server as the root user

```
$ mysql -uroot -p mysql
```

3.6.2 Configuration

The default configuration should be okay. Just make sure the `bind-address` directive[9] specifies a correct ip address value[10].

3.7 Node.js

3.7.1 Installation

The Node.js tarball should be downloaded from the official download page[11]. At the time of writing, the current version is 0.10.5.

To unpack the distribution tarball and verify its contents, please run the commands in code listing 23.

Listing 23: Unpacking the tarball to the `local` directory and checking its contents

```
$ tar xz -C ~/local -f ~/dist/node-v0.10.5-linux-x64.tar.gz
$ ls ~/local/node-v0.10.5-linux-x64
bin  ChangeLog  lib  LICENSE  README.md  share
```

Modify the `$HOME/.profile` file so that its bottom looks like the example in code listing 24.

[9]located in `/etc/mysql/my.cnf`

[10]the default of `127.0.0.1` is okay if you run the MySQL server and the space rocket in the same machine, otherwise you will need to specify an ip address that the space rocket can use to remotely connect to the MySQL server

[11]`http://nodejs.org/download/`

Listing 24: Code to add to the bottom of the `.profile` file

```
JAVA_HOME="$HOME/local/jdk1.7.0_21"
SBT_HOME="$HOME/local/sbt"
SBT_OPTS='-XX:+CMSClassUnloadingEnabled -XX:MaxPermSize=256M'
MAVEN_HOME="$HOME/local/apache-maven-3.0.5"
NODE_HOME="$HOME/local/node-v0.10.5-linux-x64"
PATH="$NODE_HOME/bin:$MAVEN_HOME/bin:$SBT_HOME/bin:$JAVA_HOME/bin:$PATH"
```

Having modified the `.profile` file, run the commands in code listing 25 to make the changes available to the current prompt and to verify that node[12] and npm[13] work.

Listing 25: Sourcing the `.profile` file and verifying that node and npm work

```
$ source ~/.profile
$ node --version
v0.10.5
$ npm --version
1.2.18
```

3.8 Further notes

The `.profile` file gets automatically sourced at every new login so there's no need to manually source it from now on.

[12]`node` is the javascript runtime
[13]`npm` is the node package manager

4 Building

Software modules need to be built in the same order as they appear in this document.

In this section we assume a `$HOME/build` directory has been created and all build work, including checking out the source code, will be performed in this directory.

We also assume that if the build tool execution exits with a successful exit code[14] and a `target` directory was created in the process, then the build was successful and all artifacts were created correctly.

4.1 Space Rocket

4.1.1 Getting the source code

Please run the commands in code listing 26.

Listing 26: Cloning the `space-rocket` module

```
$ cd $HOME/build
$ git clone http://scm.acme.com/git/space-rocket
```

4.1.2 Checking out a given version

Software versions are represented in the git repository as tags. Please follow the example in code listing 27 to list the available tags and checkout a given version.

Listing 27: Listing version tags and checking out one in the `space-rocket` module

```
$ cd $HOME/build/space-rocket
$ git tag
v1.0.2
v1.0.3
$ git checkout v1.0.3
Note: checking out 'v1.0.3'.
```

4.1.3 Build

To build the software module, please run the commands in code listing 28.

Listing 28: Building the `space-rocket` software module

```
$ cd $HOME/build/space-rocket
$ mvn clean install
```

[14] Apache Maven is very explicit with regard to its exit code, printing the final build status at the end of its processing work, i.e. it will print an explicit *success* string

5 Deployment

5.1 Example deployment scenario

This document assumes all software to be installed on the same machine. In the real world, however...
To Be Written (TBW)

5.2 Space Rocket

5.2.1 Installation

Make sure you've previously followed the build steps in 4.1.3.

Assuming you've previously built version 1.0.3, to package the spacerocket into a tarball for copying over to a remote machine, please run the commands in code listing 29.

Listing 29: Preparing the space rocket's distribution tarball

```
$ cd ~/build/space-rocket/target
$ tar cvjf ~/dist/space-rocket-1.0.3.tar.bz2 space-rocket-1.0.3
```

At this point you must copy the distribution tarball to the target machine in the `$HOME/dist` directory.

Once on the target machine, to unpack the tarball and to verify its contents, please run the commands in code listing 30.

Listing 30: Unpacking the space rocket's distribution tarball

```
$ tar xj -C ~/local -f ~/dist/space-rocket-1.0.3.tar.bz2
$ ls ~/local/space-rocket-1.0.3
bin  config  deploy  lib
```

5.2.2 Configuration

To configure logging for the space rocket, please refer to 6.1.

5.2.3 Running

To start the space rocket, please run the commands in code listing 31.

Listing 31: Starting the space rocket

```
$ cd ~/local/space-rocket-1.0.3
$ ./bin/start
Starting Akka...
Running Akka 2.1.2
Deploying file:/home/acme/local/space-rocket-1.0.3/deploy/space-rocket-impl_...
Starting up com.acme.platform.spacerocket.SpaceRocketKernel
Successfully started Akka
```

The space rocket runs in the foreground. To detach from the running process, please refer to 7.

6 Configuration

6.1 Logging

All of the AP software modules make use of a logging framework called logback[15].

Logback configuration is performed through the `logback.xml` file. We will later provide the specific location of the `logback.xml` file for each software module in this section.

Full documentation regarding logback configuration can be found online[16]. We will however go through an example for your convenience.

Listing 32: An example `logback.xml` configuration file

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <configuration scan="true" scanPeriod="30 seconds">
 3     <appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
 4         <file>logs/space-rocket.log</file>
 5         <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
 6             <!-- daily rollover -->
 7             <fileNamePattern>logs/space-rocket-%d{yyyy-MM-dd}.log</fileNamePattern>
 8             <!-- keep 30 days' worth of history -->
 9             <maxHistory>30</maxHistory>
10         </rollingPolicy>
11         <encoder>
12             <pattern>%date{ISO8601} %-5level [%thread] %X{sourceThread} %X{akkaSource} %logger{36} - %msg%n
                   </pattern>
13         </encoder>
14     </appender>
15     <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
16         <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
17             <pattern>%date{ISO8601} %-5level [%thread] %X{sourceThread} %X{akkaSource} %logger{36} - %msg%n
                   </pattern>
18         </encoder>
19     </appender>
20     <root level="INFO">
21         <appender-ref ref="FILE"/>
22         <appender-ref ref="STDOUT"/>
23     </root>
24     <logger name="com.acme">
25         <level value="DEBUG"/>
26     </logger>
27     <logger name="org.springframework">
28         <level value="INFO"/>
29     </logger>
30 </configuration>
```

As you can see in code listing 32, there's a lot going on here. Let's see what's happening, line by line.

2 the configuration file should be automatically scanned for changes every 30 seconds and reloaded if necessary

3 a file appender is declared

4 the appender in 3 writes to a file called `space-rocket.log` in the `logs` directory

5 a time based rolling policy appender is declared

7 the appended in 5 will write files using the pattern specified here

9 files older than 30 days are automatically deleted

12 declares the pattern that will be used when printing to the appender in 3

15 an appender that writes to the standard output is declared

17 declares the pattern that will be used when printing to the appender in 15

20 declares the root level

21 attaches the appender in 3 to the root logger

22 attaches the appender in 15 to the root logger

24,25 the `com.acme` logger category is configured at `DEBUG` level

27,28 the `org.mylin` logger category is configured at `INFO` level

[15] `http://logback.qos.ch/`

[16] `http://logback.qos.ch/manual/configuration.html`

7 Running

7.1 No Hang Up

Commands that run in the foreground can be run in the background by using a combination of the `nohup` command and the shell's job control facilities. An example of starting the space rocket in the background is shown in code listing 33.

Listing 33: Starting the space rocket in the background

```
$ cd ~/local/space-rocket-1.0.3
$ nohup ./bin/start.sh &
[1] 2522
nohup: ignoring input and appending output to  nohup.out
```

Standard output is now redirected to the `nohup.out` file and the process is running with id 2552 and job id 1. This means that you can later stop the process as shown in code listing 34.

Listing 34: Stopping the space rocket

```
$ kill %1
[1]+  Terminated              nohup ./bin/start.sh
```

To get a listing of the jobs running in the background, please run the commands in code listing 35.

Listing 35: Listing the jobs running in the background

```
$ jobs
[1]+  Running                 nohup ./bin/start.sh &
```

7.2 Console Window Managers

Console window managers are software that allow multiple virtual terminals to be run on the same console. They also allow detaching from the console and reattaching at a later time. Examples of such software are screen[17] and tmux[18].

Ubuntu has a convenient wrapper for console window managers called byobu[19].

We recommend getting familiar with a console window manager of your choice instead of relying on `nohup`.

An excellent, simple, short tutorial on using byobu can be found in the online Ubuntu Community Help[20].

[17] `http://www.gnu.org/software/screen/`
[18] `http://tmux.sourceforge.net/`
[19] `http://byobu.co/`
[20] `https://help.ubuntu.com/community/Byobu`

8 Appendix

8.1 Acronyms used throughout the document

AP ACME Platform

BSD Berkeley Software Distribution

JDK Java Development Kit

OS Operating System

SBT Simple Build Tool

TBW To Be Written

References

[1] “ACME Platform - Architectural Design Document”, ACME Corp., Rome, Italy, 2014.

[2] “ACME Platform - Glossary”, ACME Corp., Rome, Italy, 2014.