How automated deployment can reduce cost, risk and complexity

Erik Drolshammer
Consultant, Objectware AS

Who is Erik Drolshammer?

Consultant Objectware AS – erik.drolshammer@objectware.no

Lead on Agile 2.0 initiative

Enterprise Maven Infrastructure

JigZaw - Agile Testing done Right

JaMaCU Deployment Strategy

Codehaus Mojo Committer

Learn how Maven can be used to automate the deployment process

goal

Agenda

Motivation

How to implement

Benefits

Motivation

- what problems do we want to solve?

demo

Configuration

Where is the application installed?

Which version is installed?

How is the configuration loaded?

Where are the *configuration files*?

Security

You run the application as the ROOT USER?!?!?

What do you mean you don't know what privileges are needed?!?

All users can read and write to ALL files?!?

Manual steps

Oh, so I need to copy someJar.jar to the classpath of the application before it will start?

And rename fileX to fileY?

Different build, deploy and start scripts for each environment?

For each project?

Operation

How to start the application?

How to stop it?

Where are the logs?

Motivation summary

Little/no standardization

Little automation

High risk

High cost

Implementation

- a suggested solution

demo

Prerequisites

- Artifacts must be independent of environment (build time).
- Configuration must be separated/externalized from the artifact.

<u>Note!</u>
Enterprise Maven Infrastructure provides this, but is out of scope for this presentation.

Functional requirements

- Copy artifact and its dependencies to file system
- Copy configuration files to file system
- Create empty directories
- Set ownership and permissions on files and directories
- Make daemons start automatically at boot

Implementation

Non-functional requirements

- Easy to reuse for multiple projects
- Standardization
- Automation
- Follow best practices for system administration
  - e.g. run as application user (not root)
- Reduce risk
- Reduce cost

Implementation alternatives

- Manually
- Custom scripts (ant, bash, perl, etc.)
- Maven

Implementation

Manual deployment

Non-functional requirements

• Easy to reuse for multiple projects	✖
• Standardization	✖
• Automation	✖
• Follow best practices for system administration	⚠

Legend:

✖	Hard / not possible
⚠	Possible, but not easy
✓	Easy / well-supported

Implementation

Custom deployment scripts

Non-functional requirements

- Easy to reuse for multiple projects ✖
- Standardization ⚠
- Automation ✓
- Follow best practices for system administration ⚠

Legend:

✖ Hard / not possible
⚠ Possible, but not easy
✓ Easy / well-supported

Implementation

Maven-based deployment

<u>Non-functional requirements</u>

- Easy to reuse for multiple projects ✓
- Standardization ✓
- Automation ✓
- Follow best practices for system administration ✓

Legend:

✗	Hard / not possible
⚠	Possible, but not easy
✓	Easy / well-supported

Implementation

Solution: Use Maven

- Convention over configuration
- Standardization
- Reuse
- Open Source
- Maintained by community (Apache and Codehaus Mojo)

Implementation

appassembler-maven-plugin

“The Application Assembler Plugin is a Maven plugin for generating scripts for starting java applications.”

Read more at
http://mojo.codehaus.org/appassembler/appassembler-maven-plugin

Implementation

unix-maven-plugin

“ The unix-maven-plugin is a Maven plugin for producing installation packages for UNIX platforms. “

Read more at

http://mojo.codehaus.org/unix/

Implementation

How to setup?

- **appassembler-maven-plugin**
  - generate start and stop scripts
  - extract dependencies from pom.xml
- **unix-maven-plugin**
  - create standard directory structure
  - set file and directory permissions
  - package scripts, configuration and artifacts in rpm, deb or pkg packages
  - create symlinks

Implementation

Usage example (rpm)

- **Build rpm**
  - mvn install
- **Install**
  - rpm -ivh target/app1-SNAPSHOT.rpm
- **Start**
  - /etc/init.d/app1 start
- **Stop**
  - /etc/init.d/app1 stop

Why deploy application with native packages?

- Platform independent approaches are not good enough.
- It is what sysadmins are used to.
- Don't reinvent the wheel (it works).

Benefits

- why bother?

demo

Reduced complexity

- standardization
- automation
- *generated* scripts
- natural separation between artifacts and configuration

Benefits

Reduced risk

- same procedure every time, in all environments
  - procedure is tested and verified before it comes to production
- traceability / versioning
- security (runAsUser, file permissions)
- Java Service Wrapper

Benefits

Reduced cost

- reuse setup and scripts
- Open Source (reduced maintenance)
- Automated processes take less time than manual
- Fewer deployment errors

Benefits

Experiences from Telenor Cinclus

- 13 deployment units as rpm
- 92-94% reduction in deployment time
- Everyone can install, not only the developers.

Summary

It ***is*** possible to generate installation packages for unix environments using standard, open source tools.

Use appassembler-maven-plugin and unix-maven-plugin.

SysAdmins will love you forever :)

For More Information

Enterprise Maven Infrastructure:

http://wiki.community.objectware.no/display/smidigtonull/Enterprise+Maven+Infrastructure

Automated deployment:

http://wiki.community.objectware.no/display/smidigtonull/Deploy+application+with+native+packages

unix-maven-plugin:

http://mojo.codehaus.org/unix/

appassembler-maven-plugin:

http://mojo.codehaus.org/appassembler/appassembler-maven-plugin

thank you

http://wiki.community.objectware.no/display/smidigtonull

Erik Drolshammer
erik.drolshammer@objectware.no