Packaging Java™ Applications for Ubuntu

Jeff Bailey
Support Manager
Canonical, Ltd
http://www.ubuntu.com

Harpreet Singh
Staff Engineer
Sun Microsystems, Inc.

TS-7361

Packaging Java™ Applications for Ubuntu

Tap into the fastest-growing Linux users community

Learn how to package your Java Applications to deliver into Ubuntu.

Packaging Java Applications for Ubuntu

- Introduction to Ubuntu
- Introduction to Ubuntu Packages
- Releasing a Java Application into Ubuntu
  - Use Case: Releasing Project GlassFish™
- Lessons Learned

Packaging Java Applications for Ubuntu

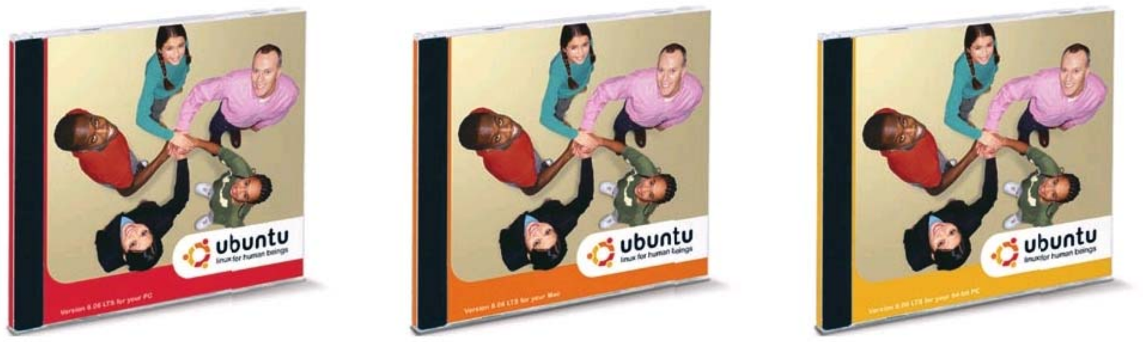
- **Introduction to Ubuntu**
- Introduction to Ubuntu Packages
- Releasing a Java Application as a Package
    - Use Case: Releasing Project GlassFish™
- Lessons Learned

What Does Ubuntu Mean?

Ubuntu is an African Bantu word meaning
“humanity towards others”
“I am what I am because of who we all are”

What Is Ubuntu?

Ubuntu, the **Linux** distribution, consists of an operating system, applications and security updates and aims to bring the **spirit** of **Ubuntu** to the software world

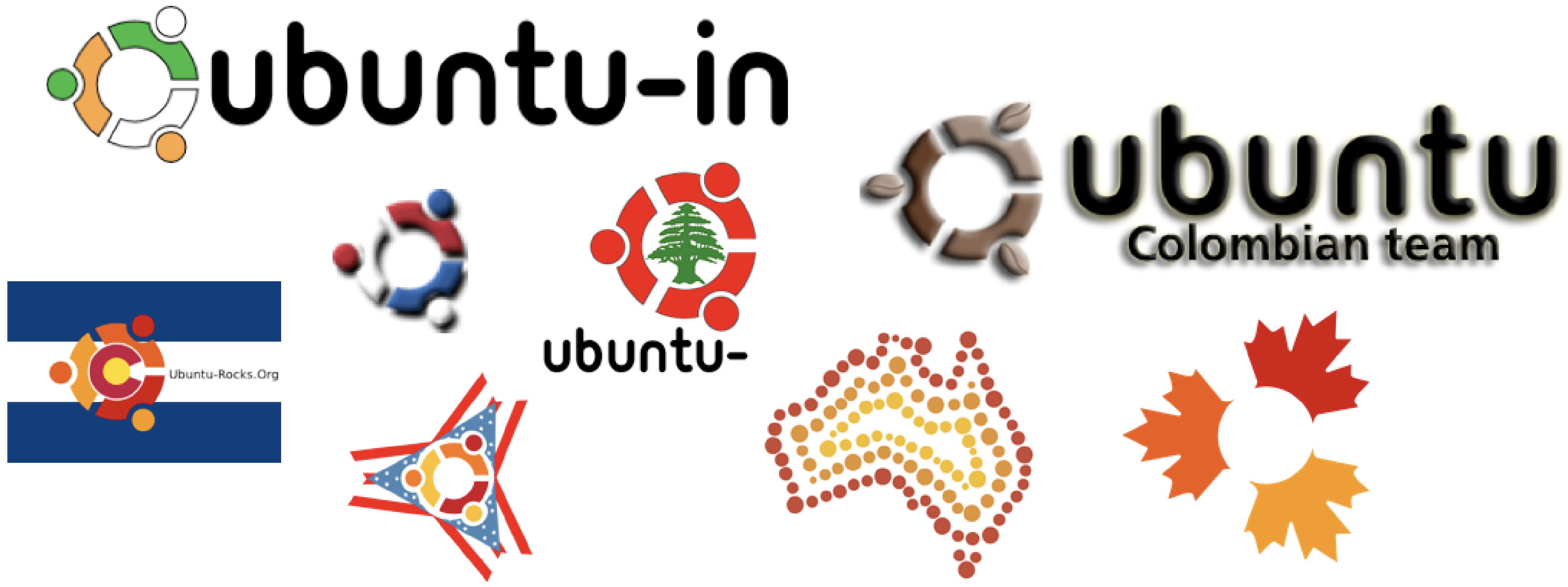


What Is Ubuntu?

- Favorite Linux distribution since 2005 according to http://distrowatch.com/
- Based on Debian GNU/Linux
- Strong desktop and notebook offering focusing on:
    - Usability
    - Localization
    - Accessibility
- Solid server platform (including port to SPARC)
- Commercially supported by Canonical and others

An Incredibly Active Community

- Over 13,000 active members of local community teams
- Over 2 million forum posts by 200,000 forum members
- 2006—Over 4 million users in just over 2 years



Packaging Java Applications for Ubuntu

- **Introduction to Ubuntu**
- **Introduction to Ubuntu Packages**
- Releasing a Java Application into Ubuntu
  - Use Case: Releasing Project GlassFish
- Lessons Learned

How Is Software for Ubuntu Distributed?

- Software in the Ubuntu repository organized into four sections (a.k.a. “components”)
  - Maintained by the core development team:
    - **main** (only Free/Open Source software)
    - **restricted** (non-Free hardware driver and firmware)
  - Maintained by the MOTU (Master of the Universe) community:
    - **universe** (only Free/Open Source software)
    - **multiverse** (non-Free, but still redistributable)
- Also, the commercial component reserved for Canonical ISV partners

How Is Software for Ubuntu Distributed?

Licensing

- Software in the Main or Universe component must be Free/Open Source
  - Example F/OSS licenses: GPL, BSD, CDDL
- Software that is not Free/Open Source, but still fully redistributable, can go into Multiverse
- Package with build or runtime dependencies in multiverse can only go in Multiverse
- Exception possible for documentation, media file and firmware (decided on a case-by-case basis)

Debian Packages Explained

What a developer needs to know about Ubuntu packages

- Based on the Debian .deb package format
- Essentially:
  - Files (binaries, libraries, doc, etc.)
  - Metadata (Dependencies, Description, etc.)
  - “Maintainer” scripts
- **The purpose**: providing Free/Open Source software (usually distributed as source) to the user in an easy to install and maintain fashion

Debian Packages Explained

Requirements and Policies

- Ubuntu packaging policy largely based on Debian: http://www.debian.org/doc/debian-policy/
- In a nutshell:
  - Software can be built from source (with some exceptions)
  - Runtime and build dependencies must be specified (and have to be fulfilled within a section)
  - Respect of the FHS is non-negotiablehttp://www.pathname.com/fhs/

Debian Packages Explained

Source package

Components:

- .dsc: source package meta-data
- .orig.tar.gz: pristine source of the software
- .diff.gz: local packaging modifications in “patch” format (including the debian/ directory)

Debian Packages Explained

Content of a minimal `debian/` directory

- debian/control: package meta-data
- debian/copyright: copyright, license, and attributions
- debian/changelog: packaging history
- debian/rules: package build Makefile

Debian Packages Explained

Maintainer scripts

- Action to be taken on package installation, upgrade and removal—scripted
  - preinst / postinst: prior and after installation
  - prerm / postrm: prior and after removal
- No user interaction (except through debconf)

Debian Packages Explained

Packaging tools

- debhelper: automating common task in the rules file
  - Examples: dh_installdocs, dh_fixperms
  - Start your Debianization with dh_make
- CDBS: An abstraction layer above debhelper
  - Make **very** short debian/rules file
  - Automatically do the right thing for the common case
- devscripts package has nice-to-have tools

Packaging Java Applications for Ubuntu

- Introduction to Ubuntu
- Introduction to Ubuntu Packages
- **Releasing a Java Application as a Package**
  - **Use Case: Releasing Project GlassFish**
- Lessons Learned

What Is Project GlassFish?

Use Case: Project GlassFish

- Open Source Java Platform, Enterprise Edition (Java EE platform) 5 Application Server
- Java EE platform 5 Compliant
- Source donated by Sun Microsystems and Oracle Corporation (peristence code - toplink)
- GlassFish V1 UR1 current stable release
  - GlassFish V1 URI under CDDL
- Next release GlassFish V2 (currently beta)
  - Will be under CDDL/GPL V2 with classpath exception
- https://glassfish.dev.java.net

What Is Project GlassFish?

- Community at http://glassfish.java.net
  - Bug Dbs, discussions, Wikis
  - Architecture documents
  - Roadmaps

GlassFish Implementation Highlights

- WS/XML Stack: Java Architecture for XML Binding (JAXB), Java APIs for XML Web Services (JAX-WS), Stax
  - Web Services Interoperability with .Net
- Web Tier: Grizzly, JavaServer Pages™ (JSP™), Servlets
- Java Persistence: Top link Essentials
- Rich Clients: AJAX and Java Web Start
- Enterprise Quality Management and clustering
- Tools

Packaging Java Applications

Identifying prerequisites

- Decide number of packages based on following criteria:
  - Platform-specific binaries
  - Licensing requirements of sub-components
- Choose your License
  - License has an impact on the choice of component
- Identify component to deliver to
- Identify your dependencies
  - Build time dependencies
  - Run time dependencies

Packaging GlassFish

Identifying prerequisites for GlassFish

- Decide number of packages: glassfish, glassfish-bin, sunwderby, imq.
- Choose your License:
  - GlassFish v1 UR1—CDDL
- Identify component to deliver to:
  - Multiverse (Non-free but redistributable)
  - Based on dependency on sun-java5-jre and license
- List your dependencies
  - Build Dependencies: devscripts, dh_make,sun-java5-jdk, sun-java5-jre
  - Run-time Dependencies: sun-java5-jre

Packaging Java Applications

Tools to package Java Applications

- Use dh_make to debianize a regular source archive
  - Creates default debian files like control, rules, changelog
- Use debuild (from devscripts package)
  - Modify rules file to write build rules
  - Modify control to define runtime dependencies for your package
  - Modify prerm, preinst to add preinstallation scripts
  - Modify postrm, postinst to add postinstallation scripts

Packaging GlassFish: Build Files

```
#Control File
Source: glassfish
Section: devel
Priority: optional
Maintainer: Harpreet Singh <harpreet.singh@sun.com>
Build-Depends: debhelper (>= 5.0.0)
Standards-Version: 3.7.2

Package: glassfish
Architecture: all
Depends: sunwderby (>= 1.0), imq (>= 1.0), sun-java5-jre,
glassfish-bin (>= 1.0})

Description: Sun's open source Java EE 5 Application
Server.
```

Packaging GlassFish: Build Files

```
#Rules File
# Build architecture-independent files here.
binary-indep: build install
build:
        # Add here commands to compile the package.
        $(MAKE)
install:
        # Install the package into debian/glassfish.
        $(MAKE) install DESTDIR=$(CURDIR)/debian/glassfish
```

Installing and Testing Packages

Tools to install packages

- dpkg -i *.deb
- Set up your own trivial repository
  - Create meta-data that describes source, packages
    - dpkg-scanpackages, dkpg-scansources
  - Add your repository under /etc/apt/sources.list
  - Refresh your repository list: sudo apt-get update
- Fetch packages with apt-get
  - sudo apt-get install glassfish

Post Build: Uploading to Ubuntu

Tools to upload packages

- Sign your packages
  - Generate your gpg key
  - Upload key to Ubuntu keyservers
  - Sign your package: debsign -k key_id
- Upload to Ubuntu servers
  - Revu (http://revu.tauware.de)
  - Use dput to upload to Ubuntu servers
- Receive feedback, make changes, and upload

Packaging Java Applications for Ubuntu

- Introduction to Ubuntu
- Introduction to Ubuntu Packages
- Releasing a Java Application as a Package
  - Use Case: Releasing Project GlassFish
- **Lessons Learned**

Lessons Learned

Tips and caveats about packaging for Ubuntu

- **Break the software into discrete components**
  - Unbundle useful libraries, think reusability!
- **Have the software licensing figured out**
  - Be careful when incorporating third-party project into yours, and give credit where it's due
- **Introducing a new package requires all build dependencies to be packaged**
- **Don't sidestep the system tools**
  - Software with their own built-in update mechanism are discouraged

Lessons Learned

Tips and caveats about packaging for Ubuntu

- **Don’t rely on graphical setup tools for installation**
  - **But it is okay for runtime configuration**
- **Building package for software using Ant is easier, thanks to CDBS**
  - **GlassFish did not take this route**

Lessons Learned

Deciding where to distribute your Ubuntu package

The Ubuntu archive

- **Universe/Multiverse**
  - Maintained by community teams
  - Become a member of the **MOTUs!**
    - https://wiki.ubuntu.com/MOTU/Hopeful/Recruitment
  - Have the benefits of team work and use of Launchpad
- **Commercial**
  - Reserved for Canonical ISV partners
  - Complete control over your packages

Slightly problematic: hosting .deb packages outside of the archive (on your own host)

Lessons Learned

Final Thoughts

- Packaging for Ubuntu is non-trivial, but worth it
  - Do the right thing for your users
  - Widen the audience for your software dramatically
- Contributors welcome
  - Ubuntu—A community where you can make a difference
  - Project GlassFish—A community where you can build open source Java EE platform Application Server

Summary

- Figure out licensing requirements
- Choose a component to upload packages
- Use system-provided tools to debianize your sources
- Test and Upload
- Join the communities
  - http://www.ubuntu.com
  - https://glassfish.dev.java.net

Q&A

Packaging Java™ Applications for Ubuntu

Jeff Bailey
Support Manager
Canonical, Ltd
http://www.ubuntu.com

Harpreet Singh
Staff Engineer
Sun Microsystems, Inc.

TS-7361