Sun microsystems

Packaging Java Applications for Ubuntu

ubuntu
linux for human beings

Arun Gupta

Sun Microsystems, Inc.
http://blogs.sun.com/arungupta

Who am I ?

- Member of Project GlassFish team
- GlassFish Evangelist
- With Sun for over 8 years
- Specifications, Engineering, Standards, Interoperability, ...
- http://blogs.sun.com/arungupta

ajax blogs conf d80 digital eclipse
firefox fitness glassfish grizzly
india indigo intellij interoperability
javaone jax-ws jersey jmaki jpa
jruby jrubyonglassfish marathon
microsoft mysql netbeans nikon
phobos photography plugfest polar
presos projectmetro railsconf
railsconfeurope rest rubyonrails
running runninglog runsfm
screencast siliconvalleymarathon
soap sun suntechdays swdp tango
theserverside totd training traveltips
vista wcf web2.0
webservices windows
wsaddressing wsit youtube

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 GlassFish
- Lessons Learned

Packaging Java Applications for Ubuntu

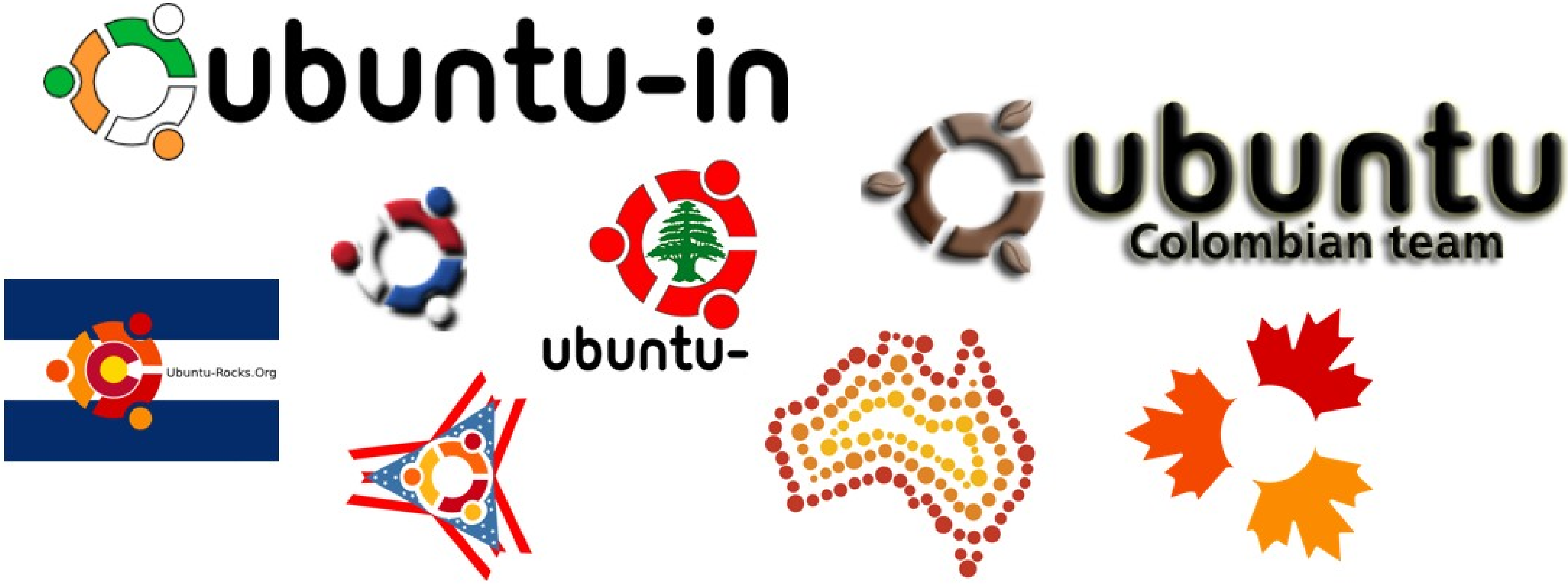
- **Introduction to Ubuntu**
- Introduction to Ubuntu Packages
- Releasing a Java Application as a Package
  - > Use Case: Releasing GlassFish
- Lessons Learned

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 GlassFish
- Lessons Learned

How is software distributed ?

- Software in the Ubuntu archive organized into four sections (aka “repositories”)

	Free	Non-Free
Supported	main	restricted
Non-Supported	universe	multiverse

🟨	Core
🟦	MOTU

How is software 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-negotiable
  - http://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 GlassFish**
- Lessons Learned

What is Project GlassFish?

Use Case: Project GlassFish

- 100% Open Source Java EE 5 Application Server
- Source donated by Sun Microsystems and Oracle Corporation (TopLink Essentials)
- GlassFish v2 current stable release.
    - CDDL/GPL v2 with Classpath Exception
- High Availability, Clustering, .NET interoperability, ...
- Community at http://glassfish.java.net
    - Wikis, Bugs, Architecture Docs, Roadmap, ...

GlassFish Highlights

- Metro: Web services stack
  - Java API for XML Web Services (JAX-WS)
  - Interoperability with .NET 3.0
- Web Tier: Grizzly, Java Server Pages, Servlets
- Java Persistence: TopLink Essentials
- Rich Clients: Ajax and Java Web Starts
  - Jmaki, JavaFX
- Enterprise Quality Management and Clustering
- NetBeans and Eclipse integration

GlassFish v3

- Small (<100 KB)
- Fast (starts up < 1 sec)
- Modular (load the required container)
- Ideal for Web 2.0 applications
- Will be Java EE 6 compatible
- Scheduled in 2009
  - Technology Preview available

Packaging Java Applications

Identifying pre-requisites

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

Packaging GlassFish

Identifying pre-requisites for GlassFish

- Decide number of packages
  - > glassfish, glassfish-bin, sunwderby, imq
- Choose your License
  - > GlassFish v2 – CDDL
- Identify repository 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 devhelper 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 (>= ${Source-Version}), imq (>=
${Source-Version}), sun-java5-jre, glassfish-bin (>=
${Source-Version})

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
- Setup 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 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 GlassFish
- **Lessons Learned**

Lessons learned

Tips and caveats about packaging for Ubuntu

- Break the software into discrete components
  - > Unbundle useful libraries, think re-usability!
- 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 ok for runtime configuration
- Don't include .jar and .class in source package
  - Does the package build from source?
- Building package for software using Ant is easier, thanks to CDBS

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
  - GlassFish – a community where you can build open source Java EE Application Server.

Summary

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

Sun microsystems

Packaging Java Applications for Ubuntu

ubuntu
linux for human beings

Arun Gupta
Sun Microsystems, Inc.
http://blogs.sun.com/arungupta