The 10 Commandments of Release Engineering

Dinah McNutt
Google, Inc.
This talk is really about "Build & Release", not just "Release"

Focus is on server-side software

The commandments are solutions to requirements

Ideas apply to software products for both internal and external customers

Ideas presented are my own, not necessarily Google's
Background

- Release processes are usually an afterthought
- Most build systems do the minimum required to "get it done"
- Release processes should be treated as products in their own right
- There is often a big disjoint between the developer writing the code and the system admin who installs it
Build & Release Steps

- Check out the code from the source code repository
- Compile and/or process the code
- Package the results
- Analyze the results of each step and report accordingly
- Perform post-build tests based on the results of the analysis step
Build & Release Process Features

- Repeatable
- Tracking of changes and the ability to understand what is in a new version of the product or product component
- An identification mechanism (e.g. build ID) that uniquely identifies what is contained in a package or product
- Implementation and enforcement of policy and procedures
- Management of upgrades and patch releases
I - Thou shalt use a source code control system.

- **Everything** needed to build should be under source control
  - source code
  - build files
  - build tools

Repeatability is a virtue.
Reproducible Build Environment

- Operating System
- Compilers
- Build tools
II - Thou shalt use the right tool(s) for the job.

Complex projects may require multiple build tools

Examples:

- make for C and C++ - the dependency checking is crucial

- ant for java

- scripting languages (bash, python, etc.)

Unnecessary complexity is a sin.
Plan to support multiple architectures and OS versions

Use centralized Makefiles for definitions common to Makefiles
- Compiler options will change between architectures
- Editing hundreds of files for a single change is no fun

Provide template files so developers can easily create new build files
IV - Thou shalt use a build process that is repeatable

And automated...
And unattended...
And repeatable...

- Identify your customers:
  - QA
  - Developers
  - Management
  - External customers

- Leverage open source tools like Hudson and Cruise Control

- Adopt a continuous build policy
V - Thou shalt use a unique build ID

- Generated at build time

- Should provide enough information so the build can be uniquely identified and reproduced

- Examples:
  - Date
  - Repository revision
  - Release version

- Should be easily obtainable
  - Included in packaging
  - Embedded in binaries
VI - Thou shalt use a package manager

- Auditing
- Installation/upgrade/removal
- Package summary (who, what, when, etc.)
- Manifest (ok, `tar -tf` gives you that.)
- Can leverage installation/upgrade/removal capabilities
- Built-in version tracking

`tar` is not a package manager...
VII - Thou shalt design an upgrade process before releasing version 1.0

- Packaging decisions can affect the ability to upgrade
VIII - Thou shalt provide a detailed log of what thou hast done to my machine

- Installing/Patching/Upgrading/Removing the software should provide a detailed log of what is happening.

- Ideally there should be a "do nothing" option so I can see what is going to happen first.
IX - Thou shalt provide a complete install/upgrade/patch/uninstall process
X - System Admin: Thou shalt apply these laws to thyself

- All of these commandments can be *applied* to system customizations