Resurrecting dinosaurs, what can possibly go wrong?

How Containerised Apps could eat our users.
“Those who cannot remember the past are condemned to repeat it”

- George Santayana
In the beginning
Windows 3.1/95 - DLL Hell

• No ABI backwards compatibility
• Most DLLs installed in C:\WINDOWS or C:\WINDOWS\SYSTEM
• Global COM Class IDs
• Service/Maintenance Nightmare
DLL Hell in Real Terms

- Developers had to dev & test Apps on every possible DLL combination
- Then retest every App patch on every possible DLL combination
- AND test every DLL patch on every possible App & DLL combination
- Then cry when it all broke anyway
Windows 2000 to the Rescue

- Side-by-side (SxS) assembly – DLL “Containerisation”
  - Separate Memory Space for each App and its DLLs
  - ‘Private DLLs’ loaded from the Application Directory
- Windows File Protection (WFP) – Disk Isolation of System DLLs
- DLL Universal Problem Solver (DUPS) – Audit all the DLLs in use and help migrate ‘legacy’ applications to SxS bundles
Problem Solved? Right?

- Security nightmare
  - Security relevant DLLs lurking in countless application folders
- Maintenance nightmare
  - How are we going to update our app? Oh we’ll ship an updater!
- Legal nightmare
  - Can we legally redistribute all the DLLs we need to?
- Storage vendor dream
  - More disk consumption, everyone buying bigger disks!
Meanwhile in Linuxland
Distributions – Solving Real Problems

• Security
  – Security Teams auditing packages, monitoring CVEs & embargoed lists

• Maintenance
  – Maintainers packaging applications & keeping them updated

• Legal
  – Lawyers auditing licenses and ensuring compatibility/compliance
In Defence of Shared Libraries/Dependencies

- Not just about using less space on disk
- Distributing fewer libraries have broad benefits
  - Fewer INSECURE libraries, more easily patched
  - Less manpower required to maintain/update
  - Easier to review/ensure legal compliance
Mission Accomplished?

- Compatibility
- Portability
- Pace of Change vs “It just works”
Windows 3.1/95 - DLL Hell

- No ABI backwards compatibility
- Most DLLs installed in C:\WINDOWS or C:\WINDOWS\SYSTEM
- Global COM Class IDs
- Service/Maintenance Nightmare
Compatibility

- Many distributions with many different libraries and apps
- Different apps require different libraries
- Application developers don’t want to worry about what other application developers have chosen as their dependencies
Compatibility

- Many distributions with many different libraries and apps
- Different apps require different libraries
- Application developers don’t want to worry about what other application developers have chosen as their dependencies
- But application developers don’t (often) worry about this
- Distro Maintainers work on this for F/OSS licensed apps
Portability

• Many distributions with many different libraries and toolsets
• Application Developers don’t want to learn dozens of toolsets, nor rebuild & retest their application on a dozen platforms
Portability

- Many distributions with many different libraries and toolsets
- Application Developers don’t want to learn dozens of toolsets, nor rebuild & retest their application on a dozen platforms

- But application developers don’t (often) worry about this
- Distro Maintainers solve the problem for F/OSS licensed apps
Pace of Change vs “It just works”

- Many distributions with fixed release schedules
- Distributions freeze package/library versions to aid ‘stability’
- Holds back new application versions from users
Pace of Change vs “It just works”

- Many distributions with fixed release schedules
- Distributions freeze package/library versions to aid ‘stability’
- Holds back new application versions from users

- But application developers don’t need to worry about this
- Rolling Distributions resolve this with increasing efficiency
Back to the Future!
Containerised Applications to the Rescue

- AppImage, FlatPak, Snappy
- Provides users with a “Bundle” containing App + Libraries
- Runs the App in some kind of Sandbox or Container
The Big Promises

• Compatibility – SOLVED
  – Only compatible libraries in the bundle
• Portability – SOLVED
  – All dependencies in the bundle
• Pace of Change – SOLVED
  – App developers can distribute at their pace, not a distro pace
• “It just works” - SOLVED
Compatibility & Portability

Apps

Frameworks

Ubuntu Core

Enablement
Compatibility & Portability
Compatibility & Portability

- Containerised Apps at some point make assumptions of a common standard base provided by the Distribution
- No such common base exists in a practical sense
Compatibility & Portability

2. Gather suitable binaries of all dependencies that are not part of the base operating systems you are targeting. For example, if you are targeting Ubuntu, Fedora, and openSUSE, then you need to gather all libraries and other dependencies that your app requires to run that are not part of Ubuntu, Fedora, and openSUSE.

3. Create a working AppDir from your binaries. A working AppImage runs your app when you execute its AppRun file.
Compatibility & Portability

• For a Containerised App to be portable, it must contain ALL compatible dependencies which MIGHT not be provided by ANY distribution

• If not, expect crashes
So it’s hopeless?

If everything is still liable to break, what is the point?

- Frameworks/Runtimes attempt to mitigate by providing curated ‘Middledistros’ to build Applications for
- The “Real” Solution: A well defined Linux Standard Base?
The Big Promises - Reality

- Compatibility – **SOLVED**
  - Only compatible libraries in the bundle
- Portability – **SOLVED**
  - All dependencies in the bundle
- Pace of Change – SOLVED
  - App developers can distribute at their pace, not a distro pace
- “It just works” - ?
History Repeating?

• Security nightmare?
  – Security relevant libs lurking in countless application bundles
• Maintenance nightmare?
  – How are we going to update our app and every single lib?
• Legal nightmare?
  – Can we legally redistribute all the libs we need to?
• Storage vendor dream
  – More disk consumption, everyone buying bigger disks!
“With Great Power…”
“… Comes Great Responsibilities”

• AppImage/FlatPak/Snappy are tools that enable App Developers to directly distribute software without the ‘need’ for Distributions

• Therefore, they must adopt the responsibilities which come with being a distributor of software
Compatibility & Portability

Consider everything an App needs that isn’t in the Bundle

- Can this break my App if the ABI changes?
  - If YES, then move it to the Bundle

- Can I rely on it being there on ALL systems?
  - If NO, then move it to the Bundle
Compatibility & Portability in Real Teams

Application Developers will still need to

- Dev & test Apps on every possible distro
- Then retest every App patch on every possible distro
- Then cry when it all breaks anyway
Broader Responsibilities

- Security – Monitor & rapidly react to CVEs. Audit libraries. Do not assume sandboxing is enough.
- Maintenance – Update all bundled dependencies in a timely manner
- Legal – Review licences of all bundled dependencies and ensure compliance & compatibility
Distributions can be part of the solution

- Distributions should like the promise of Containerised Applications
- Less work & responsibility for us is always good
- Should not be fearful of the transfer of responsibility, but should not encourage it blindly either
Distributions can be part of the solution

- A Common Base ("LSB for the Container Age") must be considered
  - Without one, the portability promise is unachievable
- Distributions have decades of tools and talent for dealing with the broader issues. USE THEM
- Don’t reinvent every wheel just because we can
One more thing
Rolling Releases for Everyone?

- To get Applications in the hands of users fast, what model beats a rolling distribution?
- Users can be guaranteed an integrated “built together” experience
- Security/Maintenance burdens less broadly distributed, fewer points of failure, Devs don’t need to be security engineers
- “It just works” can be reached with good tools – OBS & openQA
Join Us at www.opensuse.org
License
This slide deck is licensed under the Creative Commons Attribution-ShareAlike 4.0 International license. It can be shared and adapted for any purpose (even commercially) as long as Attribution is given and any derivative work is distributed under the same license.

Details can be found at https://creativecommons.org/licenses/by-sa/4.0/

General Disclaimer
This document is not to be construed as a promise by any participating organisation to develop, deliver, or market a product. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. openSUSE makes no representations or warranties with respect to the contents of this document, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. The development, release, and timing of features or functionality described for openSUSE products remains at the sole discretion of openSUSE. Further, openSUSE reserves the right to revise this document and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes. All openSUSE marks referenced in this presentation are trademarks or registered trademarks of SUSE LLC, in the United States and other countries. All third-party trademarks are the property of their respective owners.

Credits
Template
Richard Brown
rbrown@opensuse.org

Design & Inspiration
openSUSE Design Team
http://opensuse.github.io/branding-guidelines/
FROM php:5.6-apache

RUN atheme route expires

# Install the PHP extensions we need
RUN apt-get update && apt-get install -y libpng12-dev libjpeg-dev && rm -rf /var/lib/apt/lists/* 
 && docker-php-ext-configure gd --with-png-dir=/usr --with-jpeg-dir=/usr 
 && docker-php-ext-install gd www_ALL opcache

# Set recommended .ini settings
# see https://secure.php.net/manual/en/opcache.installation.php
RUN [ 
    echo 'opcache.memory_consumption=128'; 
    echo 'opcache.ignored_files=1'; 
    echo 'opcache.interned_strings_buffer=8'; 
    echo 'opcache.max_accelerated_files=4000'; 
    echo 'opcache.revalidate_freq=10'; 
    echo 'opcache.fast_shutdown=1'; 
    echo 'opcache.enable_cli=1'; 
] > /usr/local/etc/php/conf.d/opcache-recommended.ini

VOLUME /var/www/html

RUN WORDPRESS_VERSION 4.5
RUN WORDPRESS_SHA 43f0ef87a6e4efcf00d92211a22b0b&55c24w2

# Uptown library include /var/www/so this gives us /var/src/wordpress
RUN curl -o /var/www/wordpress.tar.gz -SL https://wordpress.org/latest.tar.gz 
 && echo "WORDPRESS_SHA "wordpress.tar.gz" | sha1sum -c - 
 && tar -xzf wordpress.tar.gz -C /var/src/ 
 && rm wordpress.tar.gz 
 && chmod -R 777 /var/www/html/data /var/src/wordpress

COPY docker-entrypoint.sh /entrypoint.sh

# gr/ ENTRYPOINT repeat END now
ENTRYPOINT ["/entrypoint.sh"]
CMD ["apache2-foreground"]
RUN curl -o wordpress.tar.gz -SL https://wordpress.org/wordpress-\$WORDPRESS_VERSION}.tar.gz
FROM php:5.6-apache
&& make -j"$(nproc)"

&& make install
FROM debian:jessie
FROM scratch
ADD rootfs.tar.xz /
CMD ["/bin/bash"]