Preventing Version Sprawl in Containers and Virtual Machines

TUT1364 – Build and manage your fleet with SUSE® Manager

Michele Bologna
Software Engineer
SUSE
How do you treat your servers?

Like pets or like cattle?

Preventing version sprawl in containers and virtual machines
TUT1364 – Build and manage your fleet with SUSE® Manager
What would happen if several of your servers went offline right now?

What would you do if one server gets “sick”?
How *should* you treat your servers?

*Treat your servers like cattle, not pets*
Preventing version sprawl in containers and virtual machines

TUT1364 – Build and manage your fleet with SUSE® Manager

Michele Bologna
Software Engineer
SUSE
Agenda

• Modern Approach: Pet vs. Cattle
• Modern Approach: Evolution
• Modern Approach: Notable Problems
• Lifecycle Management: SUSE Manager
• Building Images with SUSE Manager: Workflow
• SUSE Manager Benefits: Recap
• SUSE Manager Image Building: Notable Use Cases
• Q & A
Pet vs. Cattle
Modern Approach: Pet vs. Cattle

“Treat your servers like cattle, not pets”

Pets (antipattern):
• Every server is named (“unique snowflake”, difficult to reproduce)
• Every server is mission-critical: all hands on deck if it goes down
• Manually built, managed and “hand fed”
• Cannot be easily replaced

Cattle:
• Every server is numbered because they are identical to each other
• When one server goes down, it is taken out back and replaced on the line
• Built using automated tools
• Designed to “route around failure”: replace failed servers and replicate data quickly and easily
Modern Approach: the Evolution of Cattle

The tendency is to have an immutable production, where disposable virtual machines and containers are configured at deployment.
Notable Problems of Modern Approach
Modern Approach: the Evolution of Cattle

- **Server craftsman era**: Virtual machines (1998- )
- **VMs era**: Containers (2013- )
- **Containers era**

**Notable problems**

- **Configuration drift**: do you store the “recipe” to recreate the server somewhere? Has the server state changed against the recipe?
- **Version sprawl**: which version of the image is correct?
- **Workflow**: what process do we follow to build or rebuild an image?
- **Audit**: is the image using a vulnerable version of the software?
Modern Approach: Auditing Problems

- An ACM study on 356,218 public container images found that (2017):
  - Many images have not been updated for hundreds of days
  - Vulnerabilities propagate from parent to child images

On average, official and community images contain more than 180 vulnerabilities

Source: http://dance.csc.ncsu.edu/papers/codaspy17.pdf
Modern Approach: Auditing Problems

Top ten most popular container images each contain at least 30 vulnerabilities (2019)

Number of OS vulnerabilities by docker image

Source: https://snyk.io/blog/top-ten-most-popular-docker-images-each-contain-at-least-30-vulnerabilities/
Notable Problems: a Security Nightmare
(based on a true story)
Modern Approach: Auditing Problems

You are running an application in a container and you have been asked:

“Is our application vulnerable to Tomcat - Remote Code Execution via JSP Upload Bypass (2017-12617)?”
Modern Approach: Auditing Problems

root@golf:~ # docker run -ti tomcat bash
root@7ecf406f78ed:/usr/local/tomcat# bin/catalina.sh version
Using CATALINA_BASE: /usr/local/tomcat
Using CATALINA_HOME: /usr/local/tomcat
Using CATALINA_TMPDIR: /usr/local/tomcat/temp
Using JRE_HOME: /docker-java-home/jre
Using CLASSPATH: /usr/local/tomcat/bin/bootstrap.jar:/usr/local/tomcat/bin/tomcat-juli.jar
Server version: Apache Tomcat/8.5.38
Server built: Feb 5 2019 11:42:42 UTC
Server number: 8.5.38.0
OS Name: Linux
OS Version: 4.18.0-15-generic
Architecture: amd64
JVM Version: 1.8.0_181-8u181-b13-2~deb9u1-b13
JVM Vendor: Oracle Corporation
root@7ecf406f78ed:/usr/local/tomcat#
Modern Approach: Auditing Problems

<table>
<thead>
<tr>
<th>#</th>
<th>Application</th>
<th>Version</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>148</td>
<td>Apache</td>
<td>8.5.18</td>
<td><a href="#">Version Details Vulnerabilities</a></td>
</tr>
<tr>
<td>149</td>
<td>Apache</td>
<td>8.5.19</td>
<td><a href="#">Version Details Vulnerabilities</a></td>
</tr>
<tr>
<td>150</td>
<td>Apache</td>
<td>8.5.20</td>
<td><a href="#">Version Details Vulnerabilities</a></td>
</tr>
<tr>
<td>151</td>
<td>Apache</td>
<td>8.5.21</td>
<td><a href="#">Version Details Vulnerabilities</a></td>
</tr>
<tr>
<td>152</td>
<td>Apache</td>
<td>8.5.22</td>
<td><a href="#">Version Details Vulnerabilities</a></td>
</tr>
<tr>
<td>153</td>
<td>Apache</td>
<td>9.0.0</td>
<td>M2</td>
</tr>
<tr>
<td>154</td>
<td>Apache</td>
<td>9.0.0</td>
<td>M15</td>
</tr>
<tr>
<td>155</td>
<td>Apache</td>
<td>9.0.0</td>
<td>M5</td>
</tr>
</tbody>
</table>

To be repeated manually for all containers and all packages

Auditing scenarios

- How often do your audit your base images?
- How long does it take to check your compliance status?
SUSE Manager
SUSE Manager

Best-in-class open source infrastructure management solution designed to help your enterprise DevOps and IT Operations teams to:

• Optimize operations while reducing costs

• Reduce complexity and regain control of IT assets

• Ensure compliance with internal security policies and external regulations
SUSE Manager Image Building
image building with SUSE manager

- Code Repository
- Code Repository
- Code Repository
- ISV
- SUSE Customer Center
- Package Hub
- Container Hub
- SUSE Manager
- Open Build Service
- Application
- Code Repository
- Image Building with SUSE Manager
Hands on SUSE Manager Image Building
Image Building with SUSE Manager: Workflow

1. Define a worker (build host)

![Edit System Details](image)
2. Create an image profile

- **Label**: `my-awesome-image-profile`
- **Image Type**: Dockerfile
- **Target Image Store**: container-registry
- **Dockerfile URL**: `https://github.com/SUSE/manager-build-profiles.git#master/Containers/apache`
- **Activation Key**: `1-DEFAULT`
- **Custom Info Values**: Create additional custom info values

[Create Image Profile]
## Image Building with SUSE Manager: Workflow

<table>
<thead>
<tr>
<th>Configuration option</th>
<th>Meaning</th>
<th>Container images</th>
<th>OS Images</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image store (output)</td>
<td>Where all built images will be pushed</td>
<td>Container registry</td>
<td>OS Image store (a directory served via HTTPS)</td>
</tr>
<tr>
<td>Config URL</td>
<td>A Git URL pointing to the instructions to build the image</td>
<td>Dockerfile</td>
<td>Kiwi config file</td>
</tr>
<tr>
<td>Activation Key</td>
<td>Specifies the software channels to use when building the image</td>
<td>Optional</td>
<td>Mandatory</td>
</tr>
</tbody>
</table>
Image Building with SUSE Manager: Workflow

```
# VERSION 1.0.0
FROM opensuse:42.3
MAINTAINER Michele Bologna <michele.bologna@suse.com>

ARG repo
ARG cert

RUN echo "$cert" > /etc/pki/trust/anchors/RHN-ORG-TRUSTED-SSL-CERT.pem
RUN update-ca-certificates
RUN echo "$repo" > /etc/zypp/repos.d/susemanager:dockerbuild.repo

ADD add_packages.sh /root/add_packages.sh
RUN /root/add_packages.sh

ADD pub.conf /etc/apache2/conf.d/pub.conf
RUN mkdir -p /srv/www/htdocs/pub/
ADD index.html /srv/www/htdocs/pub/index.html

CMD /usr/sbin/start_apache2 -DFOREGROUND -k start
```

Source config file are public: https://github.com/SUSE/manager-build-profiles
Image Building with SUSE Manager: Kiwi

- Kiwi is a utility to build Linux system appliances
- It creates an image file starting from a configuration file
- Created images can be ISOs as well as virtual images for QEMU, Xen and other providers (even cloud)
- It can also build images that boot via PXE or Vagrant boxes
Image Building with SUSE Manager: Kiwi config

```xml
<packages type="image">
    <package name="patterns-sles-Minimal"/>
    <package name="aaa_base-extras"/> <!-- wouldn't be SUSE without that ;-) -->
    <package name="acl"/>
    <package name="btrfsprogs"/>
    <package name="btrfsmaintenance"/>
    <package name="cron"/> <!-- needed by btrfsmaintenance -->
    <package name="curl"/> <!-- needed for openQA, maybe delete -->
    <package name="dracut"/>
    <package name="fipscheck"/>
    <package name="grub2-branding-SLE" bootinclude="true"/>
    <package name="iputils"/>

    <!--
    <package name="jens-firstboot"/> -->

    <package name="zypper-lifecycle-plugin"/> <!-- bsc#1030278 fate#320507 -->
    <package name="vim"/>
    <package name="gettext-runtime"/>
    <package name="shim" arch="x86_64"/>
    <package name="grub2"/>
    <package name="grub2-x86_64-e1f" arch="x86_64"/>
    <package name="fontconfig"/>
    <package name="fonts-config"/>
    <package name="haveged"/>  
    <package name="less"/>
</packages>
```

Source config file are public: https://github.com/SUSE/manager-build-profiles
# Image Building with SUSE Manager: Workflow

<table>
<thead>
<tr>
<th>Configuration option</th>
<th>Meaning</th>
<th>Container images</th>
<th>OS Images</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image store (output)</td>
<td>Where all built images will be pushed</td>
<td>Container registry</td>
<td>OS Image store (a directory served via HTTPS)</td>
</tr>
<tr>
<td>Config URL</td>
<td>A Git URL pointing to the instructions to build the image</td>
<td>Dockerfile</td>
<td>Kiwi config file</td>
</tr>
<tr>
<td>Activation Key</td>
<td>Specifies the software channels to use when building the image</td>
<td>Optional</td>
<td>Mandatory</td>
</tr>
</tbody>
</table>
Image Building with SUSE Manager: Workflow

<table>
<thead>
<tr>
<th>Base Channel:</th>
<th>SLES12-SP3-Pool for x86_64</th>
</tr>
</thead>
<tbody>
<tr>
<td>Child Channels:</td>
<td>SLES12-SP3-Pool for x86_64</td>
</tr>
<tr>
<td></td>
<td>✓ PROD - VETTED Clone of SLES12-SP3-Updates for x86_64</td>
</tr>
<tr>
<td></td>
<td>✓ SLE-Manager-Tools12-Pool for x86_64 SP3</td>
</tr>
<tr>
<td></td>
<td>✓ SLE-Manager-Tools12-Updates for x86_64 SP3</td>
</tr>
<tr>
<td></td>
<td>☑ SLE-Module-Containers12-Pool for x86_64 SP3</td>
</tr>
<tr>
<td></td>
<td>☑ SLE-Module-Containers12-Updates for x86_64 SP3</td>
</tr>
<tr>
<td></td>
<td>☑ SLES12-SP3-Updates for x86_64</td>
</tr>
<tr>
<td></td>
<td>☑ SUSE-Manager-Server-3.2-Pool for x86_64 SP3</td>
</tr>
</tbody>
</table>

Fine-grained selection of software channels to use when building the image across SUSE Manager
Image Building with SUSE Manager: Workflow

3. Build the image
SUSE Manager: Security Audits
Image Building with SUSE Manager: Workflow

3. Build the image

Image is inspected after build to collect installed package information (version, arch, installed timestamp)
Ensure Compliance

- Search Common Vulnerability and Exposures (CVE) database
- Check security compliance with OpenSCAP
- Identify & remediate vulnerabilities
- Verify patches
- Validate security settings
- Search Common Vulnerability and Exposures (CVE) database
Secure images with SUSE Manager

SUSE Linux Enterprise Container

GitHub

SUSE Manager

SUSE Customer Center

RPM

Local Container Registry

Scan & Rebuild

Apply fix for CVE!
Image Building with SUSE Manager: Workflow

4. Audit your images and rebuild them when needed
SUSE Manager Benefits: Recap
Modern Approach: the Evolution of Cattle

Server craftsman era

Virtual machines (1998- )

VMs era

Containers (2013- )

Containers era

Notable problems

- **Configuration drift**: do you store the “recipe” to recreate the server somewhere? Has the server state changed against the recipe?
- **Version sprawl**: which version of the image is correct?
- **Workflow**: what process do we follow to build or rebuild an image?
- **Audit**: is the image using a vulnerable version of the software?
Image Building with SUSE Manager
Mitigating Problems

Notable problems

**Configuration drift**: do you store the “recipe” to recreate the server somewhere? Has the server state changed against the recipe?

With SUSE Manager

The instructions to (re-)create the image from scratch are versioned in a Git repository. Every change is versioned. Every change to the recipe will lead to a different resulting image built by SUSE Manager.
Image Building with SUSE Manager: Mitigating Problems

Notable problems

Version sprawl: which is version of the image is correct?

With SUSE Manager

SUSE Manager is managing the entire lifecycle of the built images, from the initial building with selected channels to the rebuild option, along with security and compliance check.
Notable problems

**Workflow:** What process do we follow to build or rebuild an image?

---

**With SUSE Manager**

SUSE Manager is a central tool with a defined workflow:
- define a store and a configuration file
- build the image
- rebuild an image

Bringing up a new container or VM image is just as quickly as pressing the build button.
Notable problems

**Audit**: is the image using a vulnerable version of the software?

With SUSE Manager

SUSE Manager can quickly check to see if the image is using a vulnerable version of the package. A build for an updated image is as simple as triggering a rebuild.
Notable Use Cases
SUSE Manager for Retail

SUSE Manager for Retail uses image-building feature to:

- Centrally create and maintain images for Point of Service devices
- Deploy images for POS terminals via PXE booting
Other Use Cases

- Use SUSE Manager to build images and push them to the cloud

- Can you imagine another use case? Sky is the limit!
Q & A
If you liked this session, please rate it

Preventing version sprawl in containers and virtual machines
TUT1364 – Build and manage your fleet with SUSE® Manager

Thank you!
Backup
Virtual Machines vs. Docker Containers

- **App A**
  - Bin/Libs
  - Guest OS
  - Hypervisor
  - Host OS
  - Server

- **App B**
  - Bin/Libs
  - Guest OS
  - Hypervisor
  - Host OS
  - Server

- **App A**
  - Bin/Libs
  - Docker Engine
  - Host OS
  - Server

- **App B**
  - Bin/Libs
  - Docker Engine
  - Host OS
  - Server
Docker Basics

docker build

docker pull

docker run

docker daemon

Containers

Images

Registry
FROM suse/sles12:latest

# Create a new demo_user.
RUN /usr/sbin/useradd demo_user
# Add our demo application inside of the /demo/ directory.
ADD webapp_demo /demo/webapp
ADD web /demo/web
WORKDIR /demo
# Run everything as the "demo_user" user.
USER demo_user
# The demo web application listens on port 8080 by default
EXPOSE 8080
Docker Registry

- A registry is a storage and content delivery system, holding named Docker images, available in different tagged versions
- A Docker Registry can be public or private
- Looking for a Docker Registry?
  - SUSE Portus is an authenticated Docker Registry
  - Available on SLE 12 Enterprise Container Module