The Evolution of Linux Containers and Integration of Docker with SLES® 12

Michal Svec  
Senior Product Manager  
msvec@suse.com

Flavio Castelli  
Senior Software Engineer  
fcastelli@suse.com
Agenda

• Linux Containers
• Docker
• Demo
Linux Containers
Traditional virtualization

- **App A**
  - Bins/Libs
  - Guest OS

- **App A’**
  - Bins/Libs
  - Guest OS

- **App B**
  - Bins/Libs
  - Guest OS

- **App B’**
  - Bins/Libs
  - Guest OS

- Hypervisor (Type 2)

- Host OS

- Server
Linux Containers

- **Application container**
  - App A
  - Bins/Libs
  - Guest OS
  - App A'
  - Bins/Libs
  - Guest OS

- **System container**
  - App B
  - Bins/Libs
  - Guest OS
  - Kernel
  - App B'
  - Bins/Libs
  - Guest OS
  - Kernel

- **Hypervisor (Type 2)**
- **Host OS**
- **Server**
What is a Linux Container?

Container 1
- Kernel
  - namespaces
  - Apps
  - cgroups

Container 2
- Kernel
  - namespaces
  - Apps
  - cgroups

Kernel

Server
Why Use Linux Containers?

• Lightweight virtualization solution
  - Isolated from the other processes
  - 1 kernel to rule them all
  - Normal I/O
  - Dynamic changes possible without reboot
  - Nested virtualization is not a problem
  - No boot time or very short one
• Isolate services (e.g. web server, ftp, ...)
• Provide root read-only access
  - Mount host / as read-only
  - Add only needed resources read-write
Linux Containers Use Cases

• Deploy everywhere quickly
  - Deploy application and their dependencies together.

• Enterprise Data Center
  - Limit applications which have a tendency to grab all resources on a system:
    - Memory (databases)
    - CPU cycles/scheduling (compute intensive applications)

• Outsourcing business
  - Guarantee a specific amount of resources (SLAs!) to a set of applications for a specific customer without more heavy virtualization technologies
Linux Containers – Limitations

• They cannot run a different OS/architecture
  – Cannot run Windows containers on Linux

• Risk of evading from containers
  – Solution: user namespaces

• Shared kernel with the host
  – Syscall exploits can be exploited from within the container
  – Solution: seccomp2 (in Linux kernel since 3.5)
Linux Containers – Security

- Do not give root privileges unless needed
- Apply security patches both on the host and on inside of the container
- Secure containers with SELinux, AppArmor
  - SELinux policy applies to complete container
  - Support for SELinux with LXC on a case by case basis
  - AppArmor support is ready upstream
- Paranoid? Run the containers inside of a VM
What's New in SLES® 12

• Better integration and management of Linux Containers
  - Uses libvirt-lxc framework
  - Same management layer as KVM and XEN
  - Allows for integration with SUSE Manager and SUSE Cloud
  - Unified tooling, independent of the “virtualization” mechanism

• SELinux and AppArmor support for LXC

• Filesystem copy-on-write (btrfs integration)

• Docker
Docker
What is Docker?

“What pack, ship and run any application as a container”

- 50+ million downloads
- 700+ contributors
- 40,000+ “Dockerized” apps in Docker’s index
- 128+ meetups over 43 countries
- 15,000 3rd party projects and partnerships
Speak Like Docker

- Registry
  On-line storage for docker images
- Repository
  Bag containing several versions of an image
- Image
  Prepared system to run in a container
- Container
  Linux container running a docker image
Why Docker?

• Shipping applications everywhere
• Repository of images
  - https://registry.hub.docker.com/
  - Private repository possible
• Workflow for containers like git
  - Commits; push / pull
  - DevOps oriented
• Better disk usage: changes layers
• Easy to build new images
• Allows for image versioning
Docker

Host OS

Server

YaST

Docker Daemon

Bins/Libs

App A

App A'

App B

App B'

Container
Docker – SLES® 12
SUSE® and Linux Containers

• SLES 11
  - SP2 introduced Linux Containers (LXC)
  - SP3 brought further enhancements (easy configuration)

• SLES 12
  - Introduced Docker
  - Templates for SLE 12, SLE 11 SP3, SLE 11 SP2
  - KIWI (image building tool) can build **Docker** images
  - Tool to create SLE Docker images
  - Moved from LXC to libvirt-lxc

• SLES 12 coming soon (as an update)
  - YaST interface for Docker
  - Easy to get SLES 11 SP3 and SLES 12 Docker images
Questions & Answers
It's Demo Time!

Thank you.
Creating a Container

- Install docker
  
  # zypper in docker

- Start the docker daemon
  
  # systemctl start docker

- Search the registry for opensuse image
  
  # docker search opensuse

- Grab the opensuse image
  
  # docker pull opensuse

- List local images
  
  # docker images

<table>
<thead>
<tr>
<th>REPOSITORY</th>
<th>TAG</th>
<th>IMAGE ID</th>
<th>CREATED</th>
<th>VIRTUAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>opensuse</td>
<td>13.1</td>
<td>14192d983363</td>
<td>4 weeks ago</td>
<td>598.3 MB</td>
</tr>
<tr>
<td>opensuse</td>
<td>bottle</td>
<td>14192d983363</td>
<td>4 weeks ago</td>
<td>598.3 MB</td>
</tr>
<tr>
<td>opensuse</td>
<td>latest</td>
<td>14192d983363</td>
<td>4 weeks ago</td>
<td>598.3 MB</td>
</tr>
</tbody>
</table>

- Create a new container and run bash in it
  
  docker run -t -i opensuse:latest /bin/bash
Working with Containers

- **Show containers**
  
  ```bash
  # docker ps -a
  CONTAINER ID IMAGE COMMAND STATUS
  074190eb58c4 opensuse:13.1 "/bin/bash" Up 2 minutes
  ```

- **Stop a container**
  
  ```bash
  # docker stop 074190eb58c4
  ```

- **Start a container**
  
  ```bash
  # docker start 074190eb58c4
  ```

- **Delete a container**
  
  ```bash
  # docker rm 074190eb58c4
  ```
Build an Image – Container Way

- Create a container
  ```bash
  # docker run -t -i opensuse:latest /bin/bash
  ```

- Inside the container, do the changes
  ```bash
  # zypper in vim
  ```

- Exit the container

- Review the changes in the container
  ```bash
  # docker diff 074190eb58c4
  ```

- Commit the change
  ```bash
  # docker commit -m "Added vim" \
  -a "Joe Hacker<joe@hacker.com>" \
  074190eb58c4 \
  joehacker/dev:v1
  ```
Build an Image – Dockerfile Way

• Create a build folder

• Create build/Dockerfile with this content
  
  # Build an opensuse with vim
  FROM opensuse:latest
  MAINTAINER Joe Hacker <joe@hacker.com>
  RUN zypper --gpg-auto-import-keys ref
  RUN zypper -n in vim

• Build the image
  
  # docker build -t="joehacker/dev:v2" build

• It's all automatic!
Working with Images

• Delete an image
  
  # docker rmi 5f2fc066be0c

• Show the log of an image
  
  # docker history opensuse:latest

• Share a repository to the registry
  
  # docker push joehacker/dev

• Images can be exported / imported
  
  # docker help save
  # docker help import
Unpublished Work of SUSE LLC. All Rights Reserved.
This work is an unpublished work and contains confidential, proprietary and trade secret information of SUSE LLC. Access to this work is restricted to SUSE employees who have a need to know to perform tasks within the scope of their assignments. No part of this work may be practiced, performed, copied, distributed, revised, modified, translated, abridged, condensed, expanded, collected, or adapted without the prior written consent of SUSE. Any use or exploitation of this work without authorization could subject the perpetrator to criminal and civil liability.

General Disclaimer
This document is not to be construed as a promise by any participating company 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. SUSE 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 SUSE products remains at the sole discretion of SUSE. Further, SUSE 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 SUSE marks referenced in this presentation are trademarks or registered trademarks of Novell, Inc. in the United States and other countries. All third-party trademarks are the property of their respective owners.