Containers in Linux on z Systems: Docker

Utz Bacher <utz.bacher@de.ibm.com>
Linux on z Systems Lead Architect

IBM

A Message Brought To You By Our Lawyers

Trademarks of International Business Machines Corporation in the United States, other countries, or both can be found on the World Wide Web at http://www.ibm.com/legal/copytrade.shtml.

The following are trademarks or registered trademarks of other companies.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.
IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.
Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
Windows Server and the Windows logo are trademarks of the Microsoft group of countries.
ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.
UNIX is a registered trademark of The Open Group in the United States and other countries.
Java and all Java based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.
Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.
Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

* Other product and service names might be trademarks of IBM or other companies.

• The information contained in this publication is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this publication, it is provided AS IS without warranty of any kind, express or implied. In addition, this information is based on IBM's current product plans and strategy, which are subject to change by IBM without notice. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this publication or any other materials. Nothing contained in this publication is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.
• References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in this presentation may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.

Agenda

Containers and Docker

Containers and Virtualization

Docker on z

Containers and Docker

What are Containers?

- Virtual environment within Linux OS instance
  - So applications share OS kernel
  - Only application is started, not entire Linux environment
- Efficiency: no virtualization overhead
  - No full system or para-virtualization, but only isolation in the kernel
- Own file system tree via chroot environment
- Container separation of OS objects via „name spaces“
  - Process IDs, network devices, mount points, users, and more

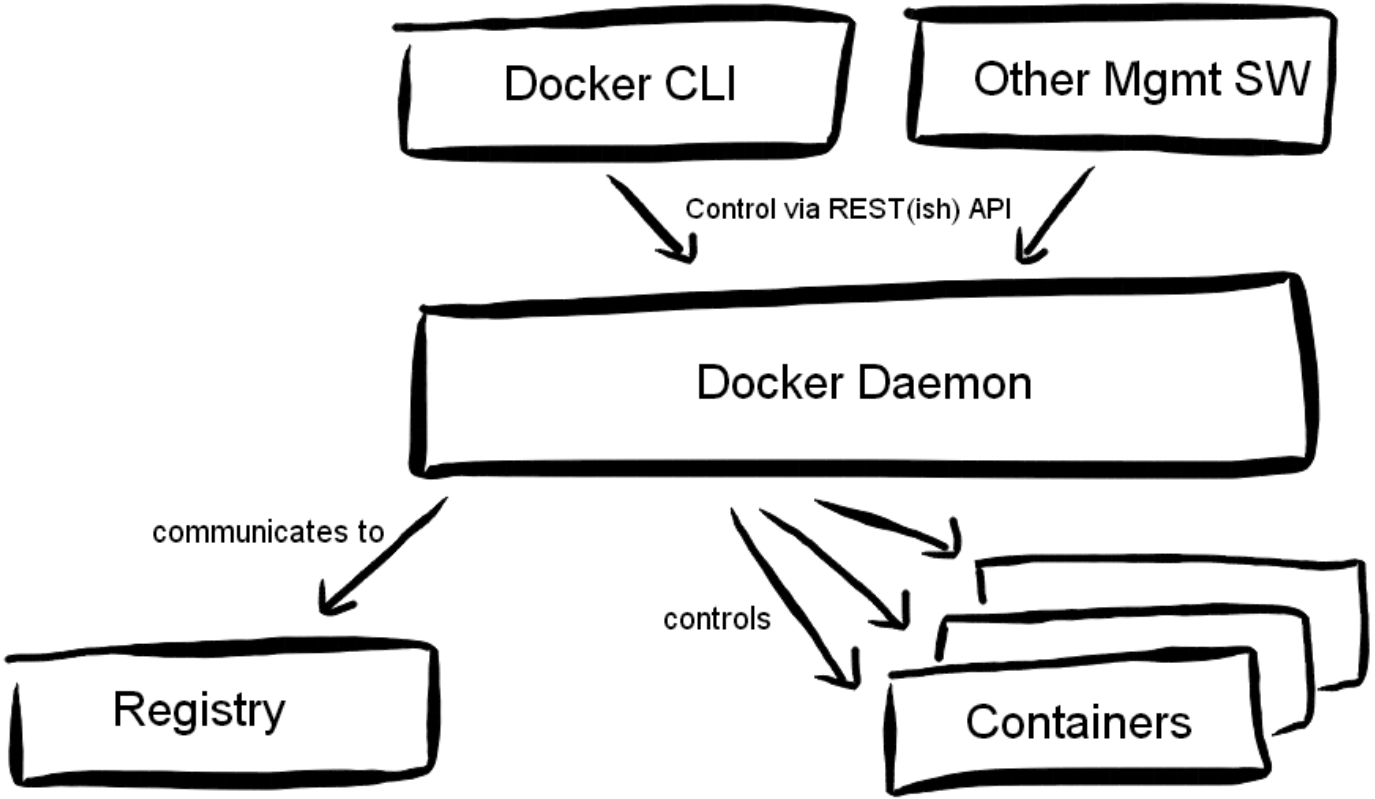
Typical Container Attributes

- Self contained sets of files – escape dependency hell, reduce test matrix
- Serve a single task
- Can build on top of each other
  - efficiency: only differences are stored
- Can be deployed simple and quickly
- Can easily be customized, re-packaged and versioned
- Can use synergies in the kernel, if images eventually base on the same libraries (same file in underlying images)
  - without having to use KSM (Kernel Samepage Merging)

Docker

- One implementation of a container solution
- Powerful tool to build, modify, deploy, run, manage containers
  - Extreme focus on efficiency, fast response times
  - Stores incremental differences and caching whenever possible
- Registries serve as central places for images
  - Efficient distribution, versioning
- Terminology
  - image: a self contained set of files, base for a container
  - container: runnable instance, based on an image
- Maintained by Docker, Inc.

Docker Structure



Dockerfile Example

- Use Dockerfiles for controlled builds of images:

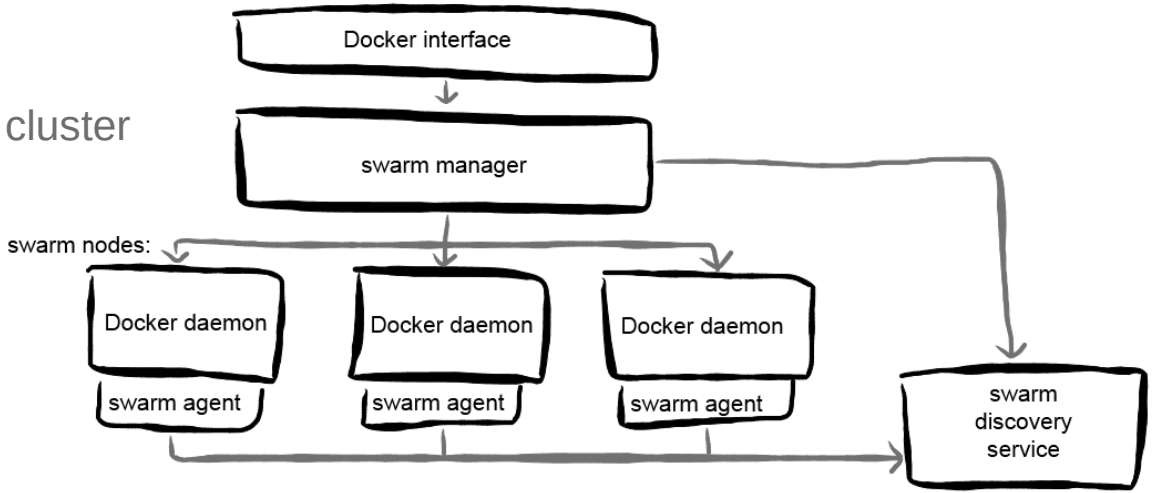
```
# use this base image. Downloaded if not present.
FROM rhel71

MAINTAINER Whatever my name is <some@address.com>

# run commands:
RUN yum install -y httpd

# copy files into the image
ADD index.html /var/www/html/

# publish a port of the container
EXPOSE 80

# how the container is started
ENTRYPOINT ["/usr/sbin/apachectl","-DFOREGROUND"]
```

Docker Ecosystem (1)

- Docker Hub
  - Docker’s public registry with 100,000+ public images
  - Private areas available
  - Contains ~100 official images of companies (Ubuntu, MongoDB, …)
  - Automated builds possible
- Private registries possible
- Docker Trusted Registry as on-premise, private registry offering by Docker

Docker Ecosystem (2)

- Additional Docker projects (some still in beta), e.g.
  - registry: central image repository with versioning
  - machine: make a virtual server a Docker host
  - compose: create multi-container applications, manage and scale them
  - swarm:
    place containers in a Docker cluster



Docker Ecosystem (3)

- Monitoring
  - cAdvisor
  - Datadog
- (Docker) Container Orchestrating and Clustering solutions
  - Kubernetes: cluster manager by Google
    - Grouping/placement, scheduling, state management
    - Builds on etcd (key value store for state and cluster coordination)
  - Apache Mesos: cluster manager
    - Builds on top of Linux nodes
    - Cluster resource and service management, resource isolation

Docker Ecosystem (4)

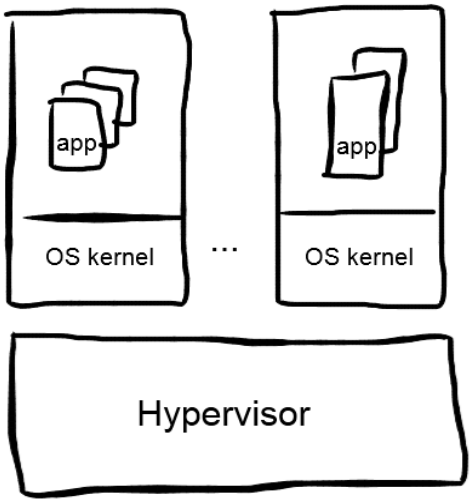
- OpenStack (components currently out of tree)
  - Management integration and standardization (keystone etc.)
    - But giving up on Docker CLI flexibility
  - Nova: Docker virt driver
    - Runs Docker images on hosts, images stored in glance
  - Heat: Docker plugin
    - Use Docker containers in Heat templates
  - Magnum project: control orchestration via Docker and Kubernetes
    - Goal to fully leverage Docker efficiency
    - Multi-tenancy for Docker and Kubernetes

Open Container Project

- Docker is de-facto container format standard
  - CoreOS launched competitive and open approach (rocket container runtime, appc container format)
- Open Container Project to define industry standard container format and runtime
  - Housed under the Linux Foundation, sponsored by many IT companies
    - Including CoreOS, Docker, Google, IBM, the Linux Foundation, Mesosphere, Microsoft, Red Hat, SUSE, VMWare, and many more
- Docker donated their container format and runtime („runc“)
- OCP standards principles:
  - Not bound to specific higher level stack (e.g. orchestration)
  - Not bound to particular client, vendor, or project
  - Portable across OS, hardware, CPU architectures, public clouds

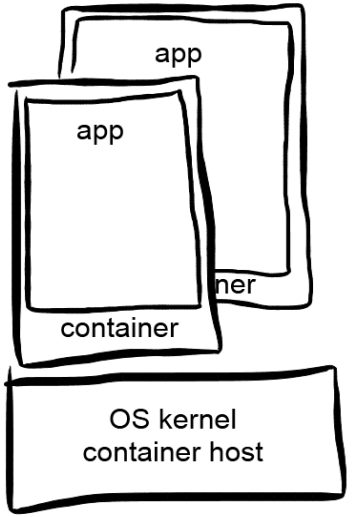
Containers and Virtualization

Virtualization vs. Containers



Infrastructure oriented:

- coming from servers, now virtualized
- virtual server resource management
- several applications per server
- isolation
- persistence

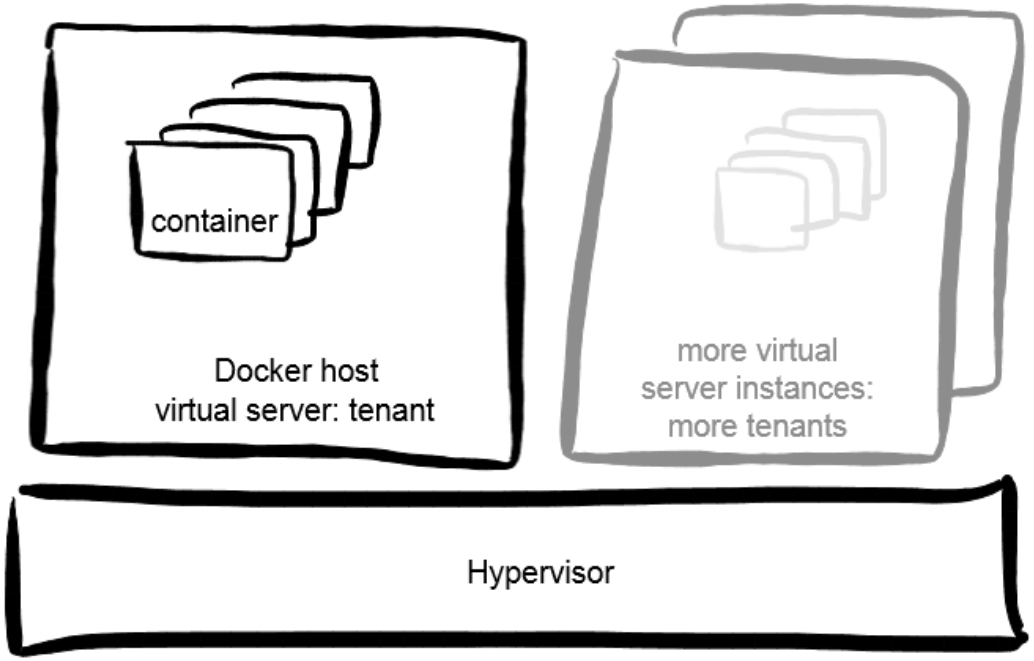


Service oriented:

- application-centric
- application management
- solution decomposed
- DevOps
- dynamic

Virtualization and Containers

- Virtual machine separation between tenants
  - Virtualization management for infrastructure
  - Isolation
- Many containers within tenants
  - Container efficiency
  - Docker management and ecosystem



Docker on z

Docker – Linux on z Systems

- Docker is written in Go
  - started with Google‘s golang on x86
  - gcc 5 comes with Go support for s390
  - gcc 5.2 used for Docker builds
  - Docker recently accepted patches for full gcc support
- Tech preview binaries available via UNICAMP
  - Anchor page http://www.ibm.com/developerworks/linux/linux390/docker.html
- Talk to your distribution partner representative for details re. z Systems distributions ;-)

Docker And Cross-Platform Portability (1)

- Docker user experience (CLI, REST API) is identical across platforms
- Containers in binary form are not portable
- Microservice architectures often have clean structure and simple individual components
- Containers are often created through Dockerfiles (build descriptions) containing:
  - Specification of base image
    - If same distribution is available on s390, usually simple
    - Currently, closest thing to Ubuntu on x86 is Debian on z
    - If base image is not available, need some workarounds to get there (e.g. „golang“)
  - Additional steps to modify image. Very often platform independent:
    - Add packages (need adaptions when using different base distro)
    - Download files
    - Perform build

Docker And Cross-Platform Portability (2)

- Usual porting considerations (not Docker-specific)
  - Source code available?
  - Most code runs on s390 already, or consists of plain C, Java, python, ruby, Go, ...
  - Platform specific behavior (e.g. application checks for CPU frequency)
  - Non-portable code (endianness, assembler)
- „Upstream work“ alleviates the pain for future updates

Docker on z: Getting Started

- Base images
  - Create based on your host distro (e.g. with a script from blog below)
  - Use a public z Systems image from Docker hub (no warranty for content!): https://registry.hub.docker.com/search?q=s390x
- A lot of Open Source applications being made available, linked from
  - https://www.ibm.com/developerworks/community/groups/community/lozopensource/
  - https://github.com/linux-on-ibm-z/ (e.g. for cAdvisor)
- Tutorial with z in mind at http://containerz.blogspot.com/
  - first steps and ecosystem

THANK YOU