# From Containers to Cloud with Linux on IBM Z

Utz Bacher <utz.bacher@de.ibm.com>
STSM Linux and Containers on IBM Z

# 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.

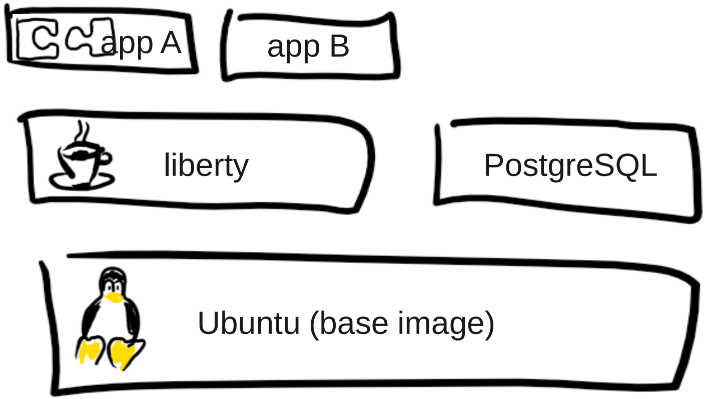
Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

* 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.

# What are Containers?

- Self-sufficient packages of software
- Layering allows for simple packaging
- Serves a single task
- Solutions are broken down into smaller services
- Consistent handling from Development to Operations



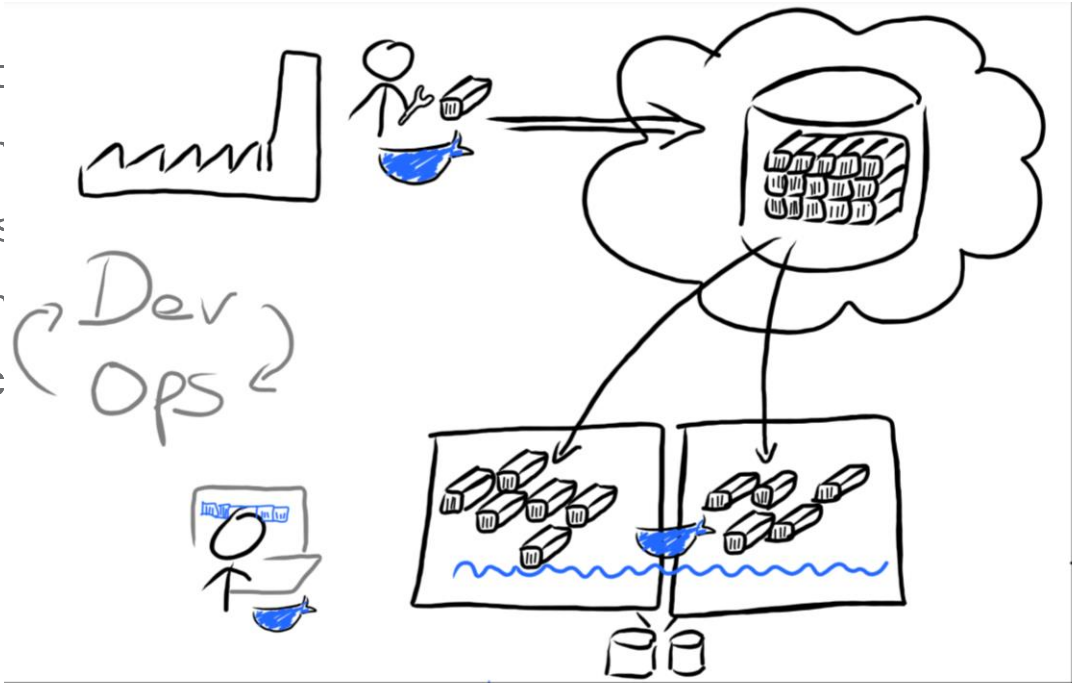
- *Docker*: An Open Platform to Build, Ship, and Run Distributed Applications
- Clustering with Docker *swarm* or *Kubernetes*

# Container Benefits

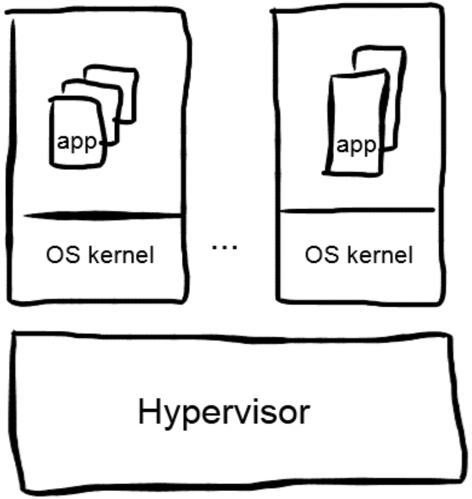
- No software-level dependencies between containers or to host
- Portability and cross platform deployment through generic build description
- Simple re-use of components in different scenarios
- Componentization of solutions (micro-services)
- Density through lightweight container mechanisms in Linux kernel
- Bridges Dev to Ops
  - consistent tooling and environment

# Container Benefits

- No software-level dependenc
- Portability and cross platform
- Simple re-use of components
- Componentization of solution
- Density through lightweight c
- Bridges Dev to Ops
  - – consistent tooling and environment

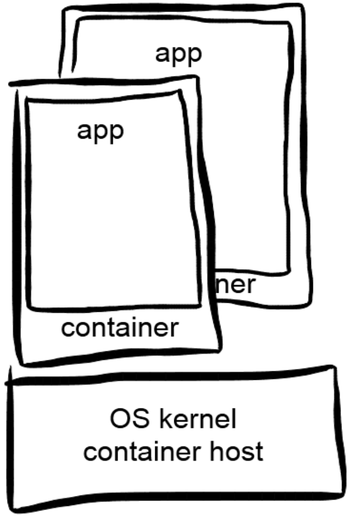


# Virtualization vs. Containers



Infrastructure oriented:

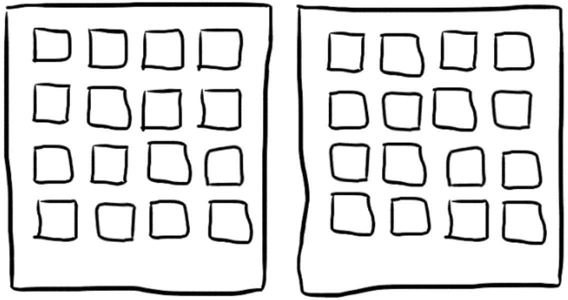
- coming from servers, now virtualized
- virtual server resource management
- several applications per server
- isolation
- persistence



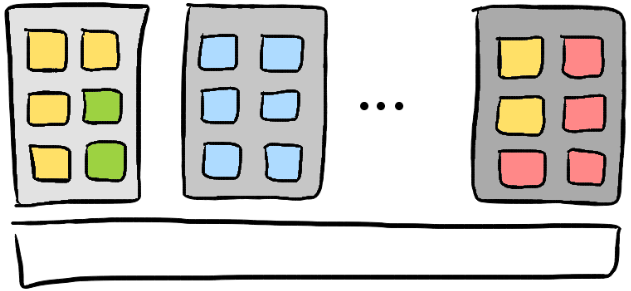
Service oriented:

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

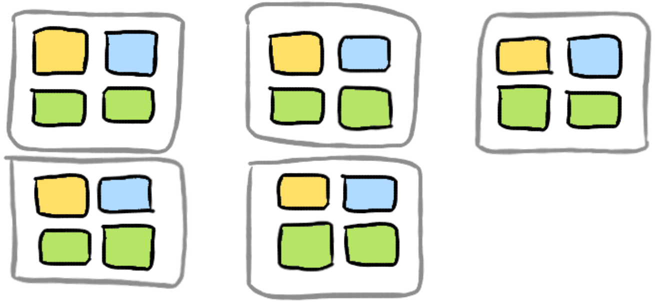
# Microservice Deployment Types



Scale up for maximum efficiency



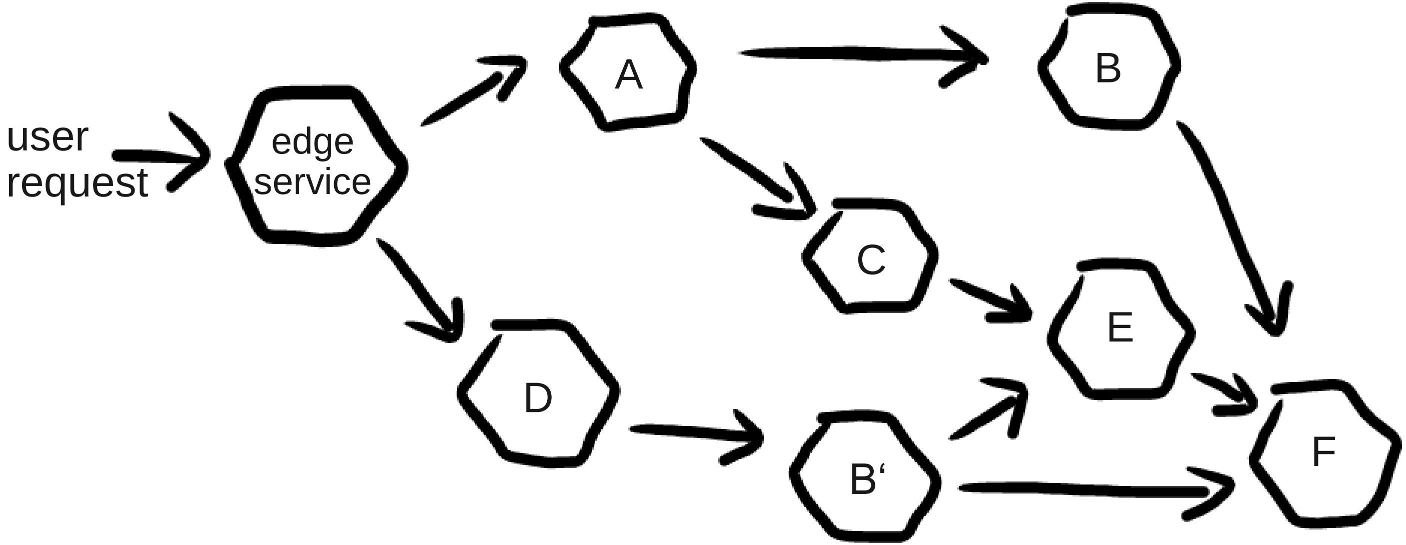
Isolation, QoS and scaling for tiers and tenants



Grouping microservices end-to-end allows for simple scaling and optimized local communication

# Microservice Challenges: Latency

*Internal flow between microservices*

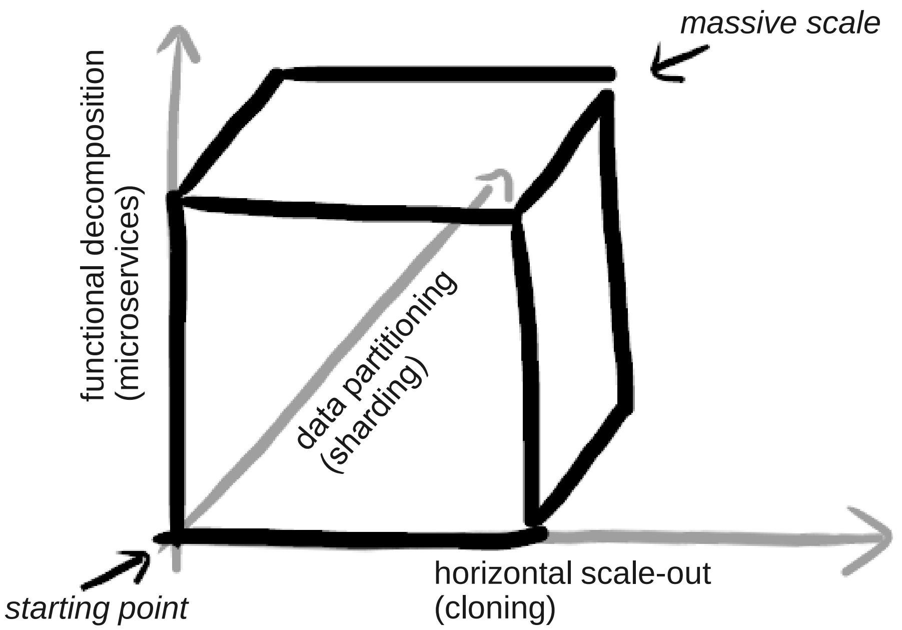


Network latencies add up in meshes of microservices
z Systems: large complex with in-box networks reduces latencies

# Microservice Challenges: Scaling

## The Scale Cube

(From Abbott & Fisher: „The Art of Scalability")


massive scale
functional decomposition (microservices)
data partitioning (sharding)
starting point
horizontal scale-out (cloning)

## IBM Z: sometimes bigger *is* better

- Replication of components is mostly simple
- Splitting applications into microservices can be hard
- Data partitioning is often hard
- Scaling stateful services can be complex
  - e.g. transactional context across microservices
- IBM Z can scale anywhere from horizontally to vertically
  - scale-up can simplify solutions

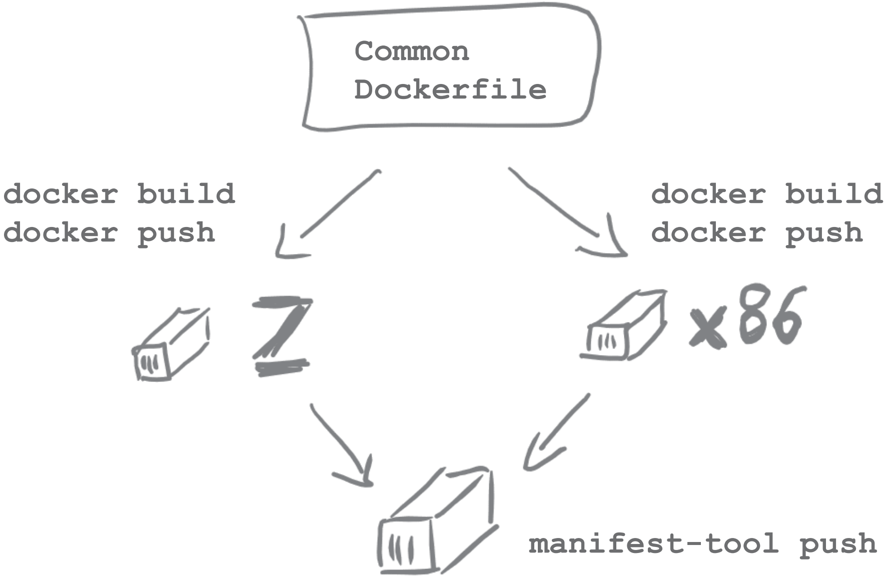
# Containers on IBM Z

- Technology and tooling identical to distributed platforms
- Second level virtualization provides
  - perfect tenant isolation with low overhead

    while
  - providing container agility and efficiency
- Co-location to traditional applications (e.g. via HiperSockets)
- Container performance inherits platform performance characteristics
  - allows both scale-up and scale-out in a box
  - good economics through density, utilization, (micro)service co-location, scaling

***Structure solutions along solution requirements, not environment-imposed restrictions***

# Multi-Architecture Images


Common Dockerfile

docker build
docker push

docker build
docker push

manifest-tool push

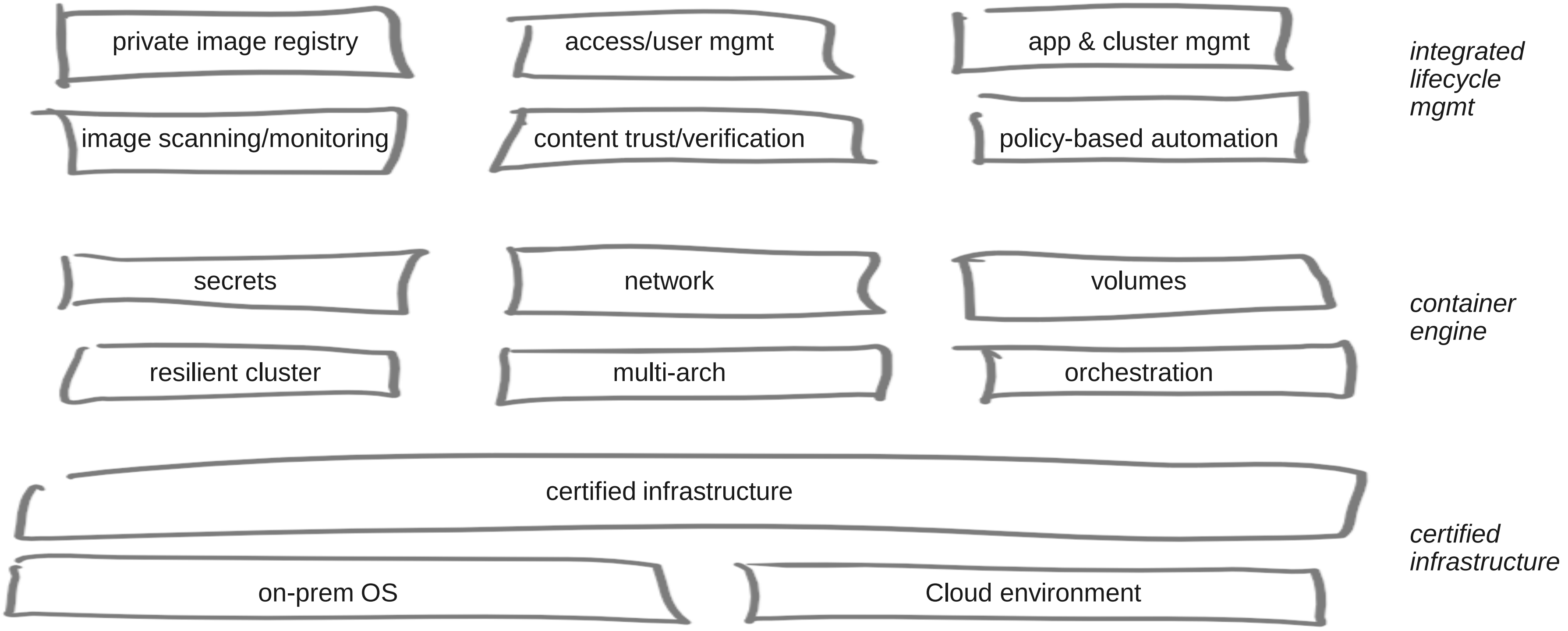
```
image: webapp:latest
manifests:
  -
    image: webapp-s390x
    platform:
      architecture: s390x
      os: linux
  -
    image: webapp-amd64
    platform:
      architecture: amd64
      os: linux
```

- All official images on Docker Hub are multi-arch today
  - Numerous images backed by s390x versions

# Docker on IBM Z – High Level Summary

- Docker and base ecosystem available with full functionality
  - Based on identical source code
  - IBM Z is part of Docker‘s „Continuous Integration pipeline“
  - Delivered as part of Docker’s (CE, EE) and Linux distribution deliverables (SLES, Ubuntu)
- Docker today enables mixed architecture development and deployment
- Commercial support and products available
  - Docker/IBM
  - Distributions
  - RogueWave
- Docker Enterprise Edition is available for IBM Z

# Docker Enterprise Edition: Container as a Service



private image registry

access/user mgmt

app & cluster mgmt

image scanning/monitoring

content trust/verification

policy-based automation

*integrated lifecycle mgmt*

secrets

network

volumes

resilient cluster

multi-arch

orchestration

*container engine*

certified infrastructure

on-prem OS

Cloud environment

*certified infrastructure*

scheme from https://www.docker.com/enterprise-edition

# Docker Enterprise Edition Tiers

- Basic: engine
- Standard: plus UCP and DTR
- Advanced: plus Docker Security Scanning

- Phase 1: engine running on IBM Z, DTR/UCP on x86
- Phase 2: all tiers running on IBM Z

Docker EE 17.06

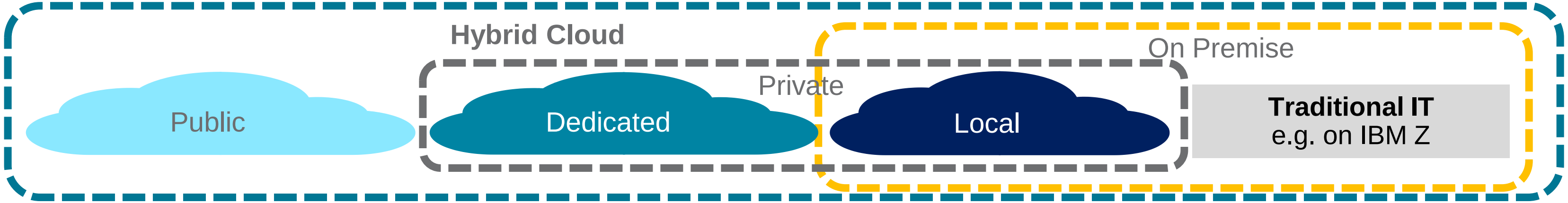
UCP 2.2.4, DTR 2.4.0

- Serviced through IBM Elite Supprt

# Kubernetes on IBM Z – High Level Summary

- Kubernetes and base ecosystem available with full functionality
  - Based on identical source code
  - IBM Z binaries are built as part of the release process
- Kubernetes today allows mixed architecture development and deployment
- Docker Hub Content (images) valid for kubernetes

# Hybrid Cloud and IBM Z



- Hybrid Cloud: Cloud-style deployment and speed + traditional IT services
- If Hybrid Cloud is taken serious, IBM Z is a natural part of it
- “Best fit” thinking applies to Cloud, too:
  - Public for ultimate scale-out, flexibility and global presence
  - Dedicated for isolation requirements off-prem
  - Local for more controlled and adapted environment
  - Local with Z for optimized services with maximum security and performance through proximity and scalability
  - Integration of traditional IT to leverage existing assets
- … with a consistent IT consumption model

# IBM Cloud Private: Ramping up on Z

- Framework for Private and Hybrid Cloud computing
- Based on open tooling/formats

Based on (Docker) containers, orchestrated with kubernetes

Common services (automation, integration, management, logging, monitoring, service mesh, etc)

IBM Middleware, Data & Analytics Services

Cloud Foundry platform for PaaS style dev't

- Integration across Clouds with Cloud Automation Manager (CAM)
  - Includes driving IaaS via Terraform → VMware and OpenStack

# Secure Service Container (SSC)

Being compromised by a rogue administrator/privileged insider is perceived as one of the biggest risks to companies.

SSC provides ideal framework to package appliances.

SSC:

- Internal closed partition for running appliances, managed through firmware

→*no need for Linux infrastructure or skills*

- Tamper proof environment with chain of trust for executed content
- Access to shell, memory, disk contents, or dumps prevented by trusted firmware code

→*Confidentiality of code and data in appliance, even against highest privilege admins*

# Outlook: SSC with Container-As-A-Service (CaaS)

Vision: Client brings workload
Platform takes care of infrastructure

SSC:

- Internal closed partition

→*no need for Linux infrastructure or skills*

- Tamper proof environment
- Access to SSC is prevented

→*Confidentiality of code and data*

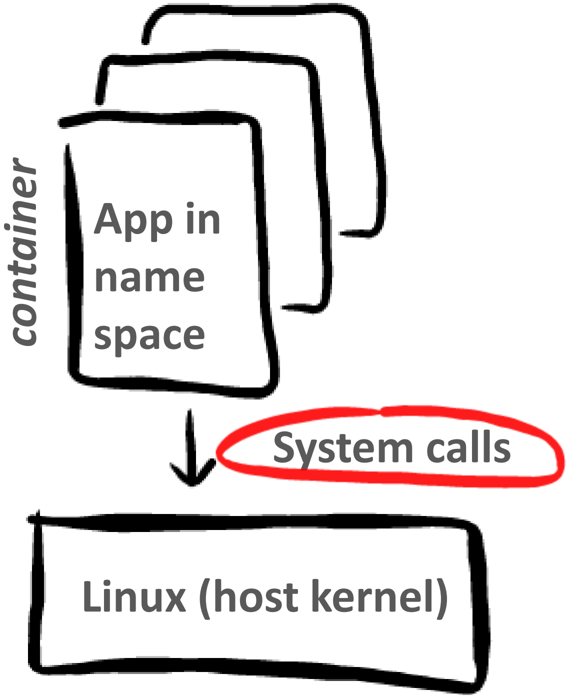
CaaS:

- Add container execution platform:
  - Docker, Kubernetes environments
- Integrates with standard management
  - e.g. Open Source tooling, Docker EE, ICP
- Confidentiality from infrastructure admin
- Note: still in early phase.
  IBM looking for beta sponsor users

# Outlook: Technology used in Current HSBN Offering

**Plain Containers:**

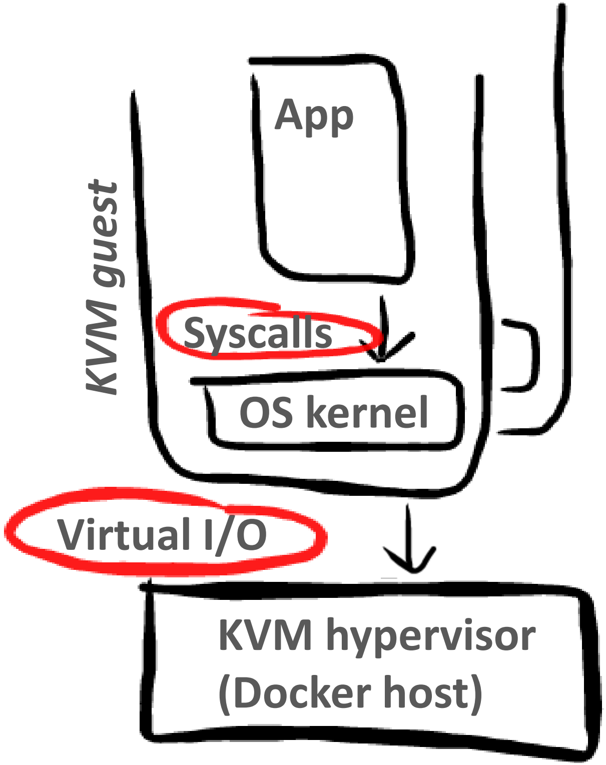
- There are hundreds of system calls
- Use can be restricted
  - libseccomp, SELinux, AppArmor, capabilities
- But still a large attack surface
  - bugs, DoS



# Outlook: Technology used in Current HSBN Offering

**Isolated Containers:**

- Transparently use KVM to run Container workload
  - Improved isolation through hardware-based virtualization technology
  - Smaller attack surface from untrusted container workload towards hosting environment
- Maintain Docker ecosystem and user experience
  - KVM not visible to user, started under the covers
  - Re-use of Docker images without any changes



# THANK YOU