Swiss Re

zOS Container Extension

Intro and user experience

Marco Egli, May 2021

Table of Contents

Base Information

- z/OS Container Extension (zCX) Overview
- Container Concepts
- zCX Architecture
- Usage Scenarios
- Example Use Cases

z/OS Container Extension (zCX) Overview

- z/OS software ecosystem extension
  - cloud native workloads
  - open-source packages
  - third-party software
- Deploy Linux on Z applications as Docker containers to run on z/OS
  - Anything with s390x architecture (IBM Z opcode)
  - Binary compatibility between Linux on Z and Container Extension

Container Concepts – Basics

- Container → group of processes that run-in isolation
- Docker → runs multiple containers simultaneously on a single host
- Encapsulate an application in a container with its own operating environment along with dependencies
  - code, run time, system tools, system libraries and settings
- Docker allows sharing of resources, like a hypervisor, such as network, CPU and disk to run multiple encapsulated workloads

Container Concepts – Why we have them

- No need to run a full copy of an operating system
  - Each container is a regular process that is running on the hosting Linux Kernel, with its own config. and isolation.
- Versioning of images
  - Simple roll back to previous version of an image, efficient way to analyse and solve problems within the application.
- Agility to deploy new applications
  - Transition from monolithic applications to distributed microservices, quickly run multiple instances for redundancy
- Isolation
  - Linux namespace (enforce and achieve required isolation) and cgroups (to manage system resources)
- Better resource utilization
  - Sharing resources to make better use of hardware

Container Concepts – Why we have them - continued

- Application Portability
  - Combining all application resources (to an image) allows it to run on multiple architectures
  - Flexibility to chose which platform the image shall run on
- Colocation of data
  - Run x86 applications on the platform where the data resides
  - Leverage rich qualities of service provided by IBM Z and z/OS

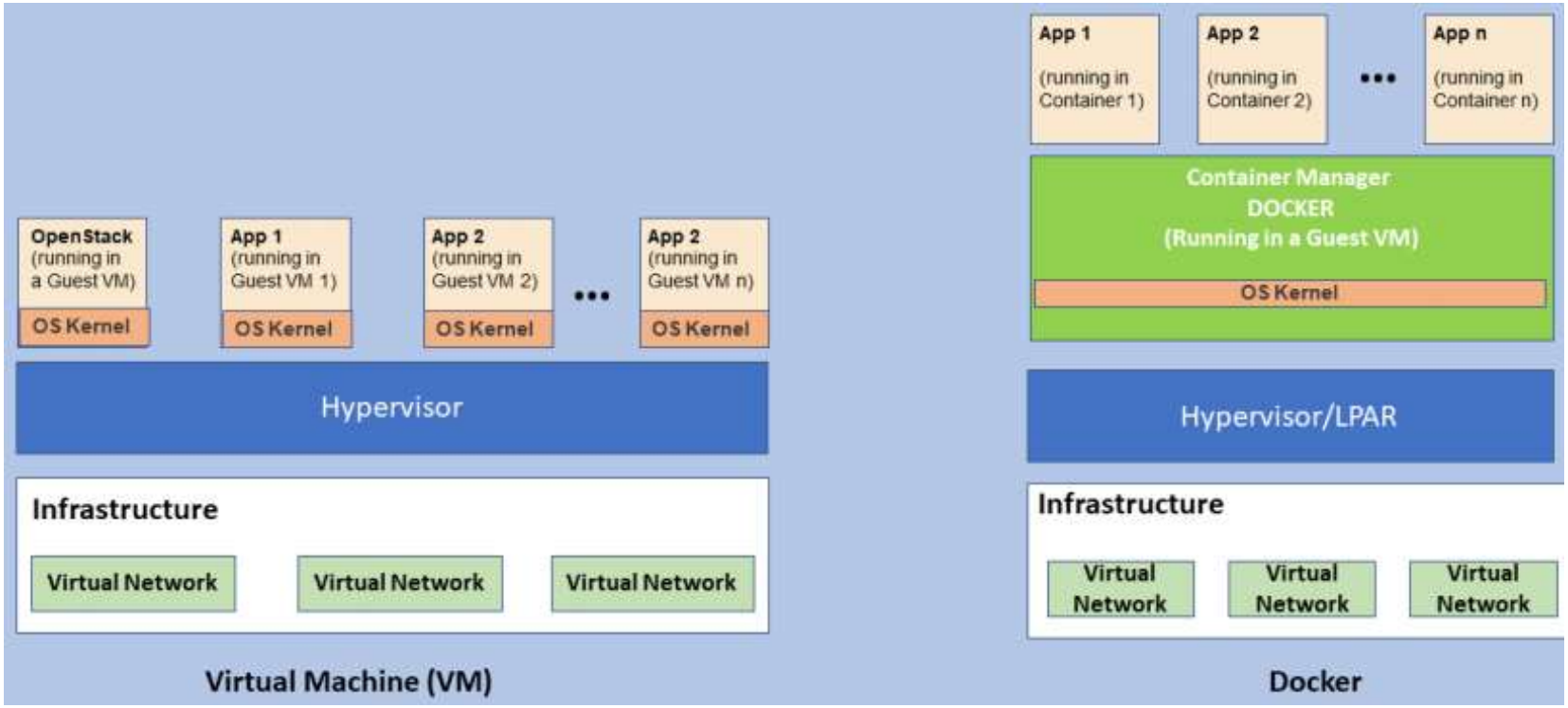
zCX Architecture

- Dockerfile is the recipe that contains instructions/commands/arguments to define Docker images.
  - A list of commands that are in sequence to build customized images.
  - Example for a web app that is based on an Alpine image and will run in Python

```
# our base image
FROM alpine:3.5

# Install python and pip RUN apk add --update py2-pip
# install Python modules needed by the Python app
COPY requirements.txt /usr/src/app/
RUN pip install --no-cache-dir -r /usr/src/app/requirements.txt

# copy files required for the app to run
COPY app.py /usr/src/app/
COPY templates/index.html /usr/src/app/templates/

# tell the port number the container should expose
EXPOSE 5000

# run the application
CMD ["python", "/usr/src/app/app.py"]
```

zCX Architecture

- VMs comparing to Docker



Usage Scenarios

- Integrate zCX workload into existing z/OS workloads (data colocation)
- Architect and deploy a hybrid solution consisting of z/OS software and Linux on Z Docker on the same z/OS system
- Open access to data analytcs on z/OS by providing standard OpenAPI compliant RESTful services
- Consolidation of workloads

Example Use Cases

- Expand z/OS software ecosystem to include support for
  - The latest Microservice (logstash, Etcd, Wordpress, etc.)
  - Non-SQL databases (MongoDB, IBM Cloudant, etc.)
  - Analytics frameworks (expanding the z/OS Spark ecosystem)
- System Management components
  - System management components in support for z/OS that are not available on z/OS
  - Centralized data bases for management
- Open-Source Application Development Utilities
  - Gitlab/Github server
  - Linux based development tools
  - IBM Urban code Deploy Server
  - Complement existing z/OS ecosystem, Zowe and DevOps tooling

Prerequisites

- Hardware
- z/OS 2.4
- z/OSMF
- DASD
- Network

Hardware

- IBM z14 type 3906 with GA2 driver level
- IBM z14 ZR1 type 3907
- IBM z15 type 8561 model T01
- IBM z15 type 8562 model T02
- Feature Code 0104
  - fee-based (first 90 days free trial with OA58969 applied)

z/OS 2.4

- No plan to be rolled back to previous release
- zCX is shipped in the z/OS BCP base and contains of two FMIDS
  - HBB77C0
    - Contains the virtualization layer for zCX. New prefix is GLZ (used for datasets and message prefix)
  - HZDC7C0
    - Contains the underlying Linux kernel, the Docker engine and the z/OSMF workflows required for provisioning, deprovisioning and other maintenance or reconfiguration activities of container instances.

z/OSMF

- Required to run required workflows stored in /usr/lpp/zcx_zos/workflows
- Available workflows
  - provision.xml
  - backup_config.xml
  - reconfigure.xml
  - restore_config.xml
  - add_data_disks.xml
  - upgrade.xml
  - rollback.xml
  - deprovision.xml
  - start_instance.xml
  - stop_instance.xml

DASD

- Allocated as VSAM linear datasets
- Allocated with primary extents only
- Use DATACLAS to allocate them with “Extended Format Required” and “Extended Addressability Enabled”

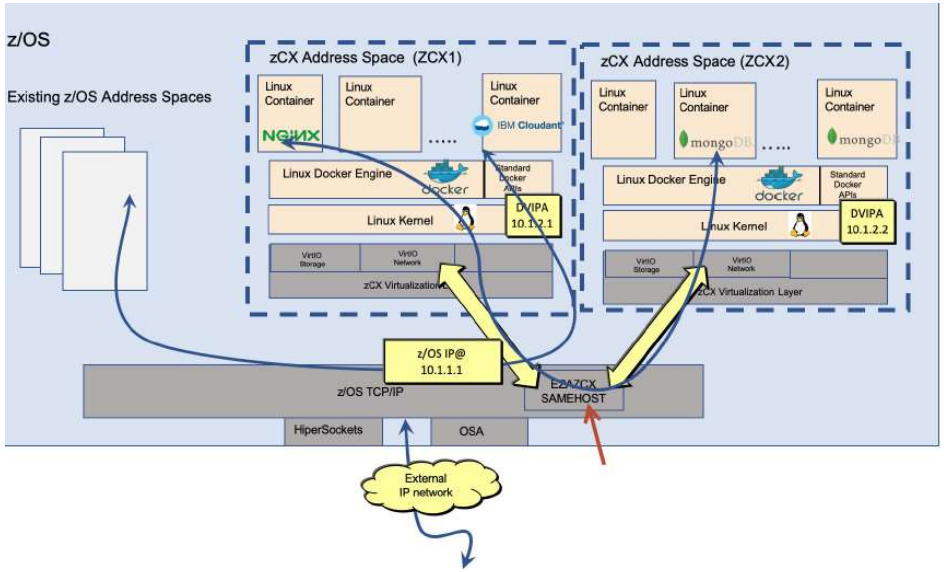
Data set type	Usage	Size requirements	Addition of space possible?
Root disk	Linux root file system	>=4GB	No; run workflow deprovision.xml and then provision.xml with different size
Configuration disk	This disk holds configuration data for zCX appliance Instances	>=2MB	No; run workflow deprovision.xml and then provision.xml with different size
User data disk	This disk holds all docker images, containers, logs, and volumes	This size is workload dependant, but plan for >=20GB	Yes; run workflow add_data_disks.xml and restart your zCX instance
Swap data disk	These disks are optionally used by the Linux kernel for paging and swapping activities when virtual memory exceeds the real memory	This size is workload dependant, but plan for >=2GB	Yes; run workflow add_data_disks.xml and restart your zCX instance
Diagnostics and log data disk	This disk holds diagnostic data, logs, and first failure data capture (FFDC) Information	>=1GB	No; run workflow deprovision.xml and then provision.xml with different size
Instance directory zFS	This disk holds the zCX appliance image, configuration file, and FFDC information	>=4GB	This VSAM data set can be expanded by secondary extents

Network

- New DVIPA type called zCX
- Must be created with VIPARange statement

```
VIPADYNAMIC
VIPARANGE DEFINE 255.255.255.255 129.40.23.13 ZCX ;
ENDVIPADYNAMIC
```

- When Using samehost interface EZASAMEMVS new interface EZAZCX is also created automatically when DynamicXCF is used



Security

- zCX Instance Security
- Security within the zCX Instance
- Docker in zCX vs Docker on distributed

zCX Instance Security – RACF

- Define Protected user to run the STC
- Define who can use z/OSMF and run workflows to manage the provisioning of zCX instances
- Protect USS configuration files

zCX Instance Security – zFS and VSAM files

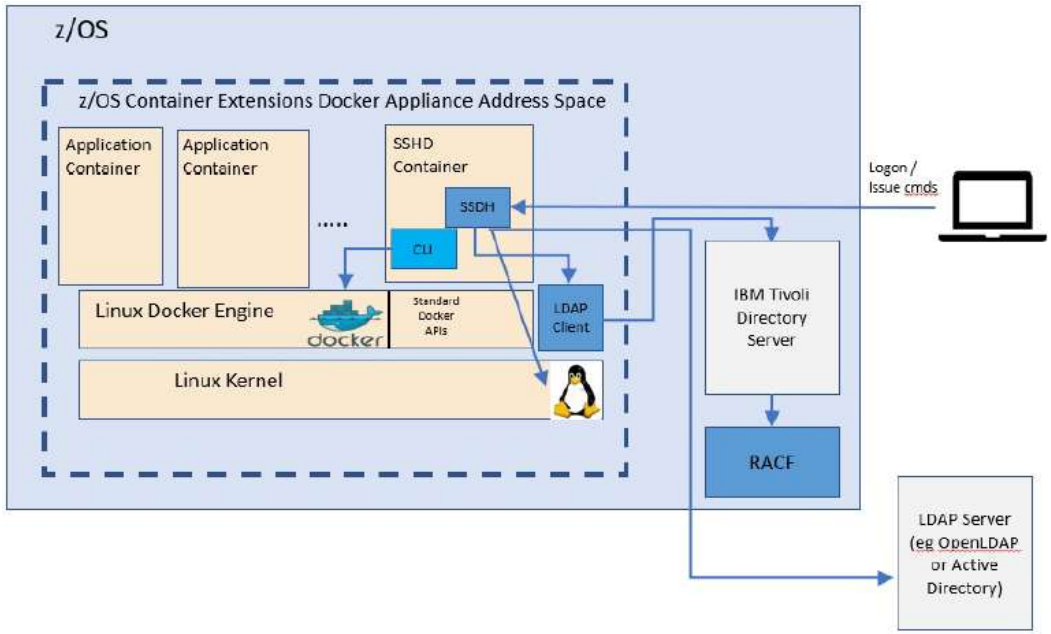
- Define access to zFS and VSAM files created by workflows
  - Workflow user requires ALTER
  - zCX STC user requires CONTROL
- Consider Pervasive Encryption

zCX Instance Security – TCPIP Networking

- Access into and out of a zCX instance is via TCPIP
- Awareness of available IP filters that might restrict access to and from zCX

Security within the zCX instance – SSH container

- Each zCX instance is running a supplied Linux operating system
  - No access to the underlying Linux operating system, one cannot become the Unix root user ID running native in this Linux
- Dedicated (default) SSH container is running to logon and isolate the native Linux system



Security within the zCX instance – administration user IDs

- Define local user IDs in the zCX instance or configure zCX instance to use external LDAP
- Logon as docker admin works only via ssh and a certificate (password usage not possible)

Docker in zCX vs Docker on distributed

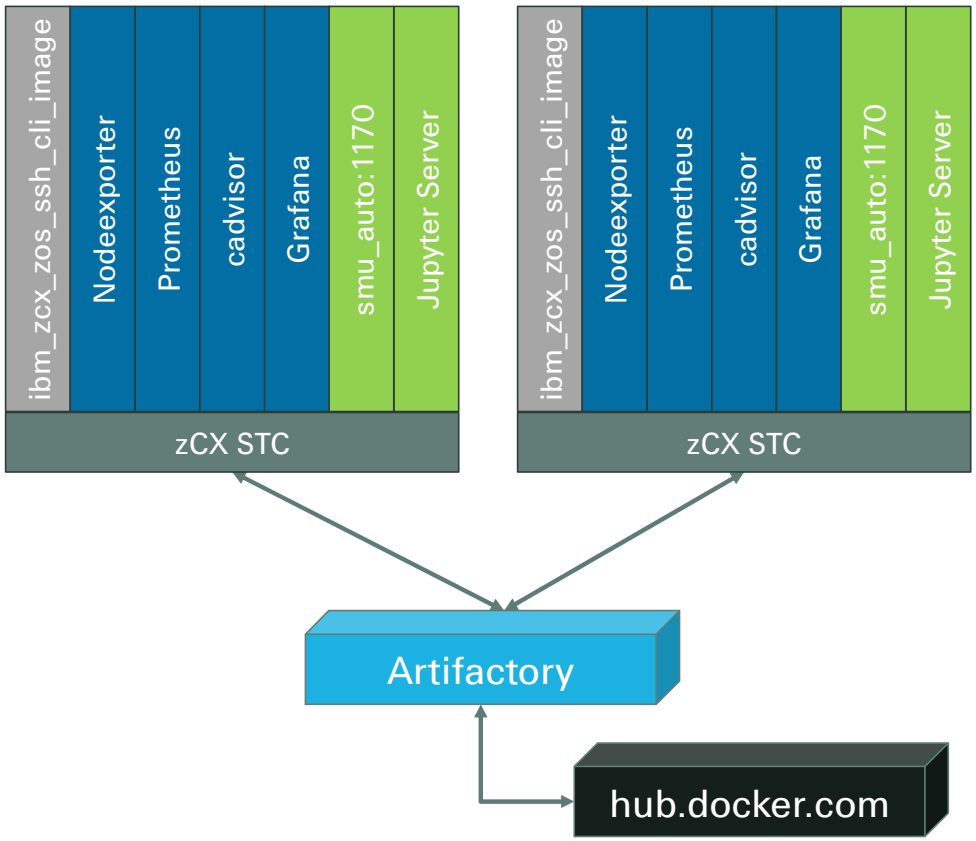
- Privileged Docker container cannot be run
- The name space of a container cannot be changed to host
- Not possible to mount a directory from the host Linux with R/W access
- Logon to Docker only possible via SSH container

User Experience @ Swiss Re

- Current Setup
- Monitoring
- Stability and Performance
- Stepping Stones
- Benefits
- Next Steps
- Summary

Current Setup

- Two zCX Instances
- Running on two different CEC
- Connected to internal Artifactory
  - Connect to hub.docker.com
- zCX Instance managed via System Automation



Legend:

IBM Default Docker container for SSH

Used for Monitoring purpose

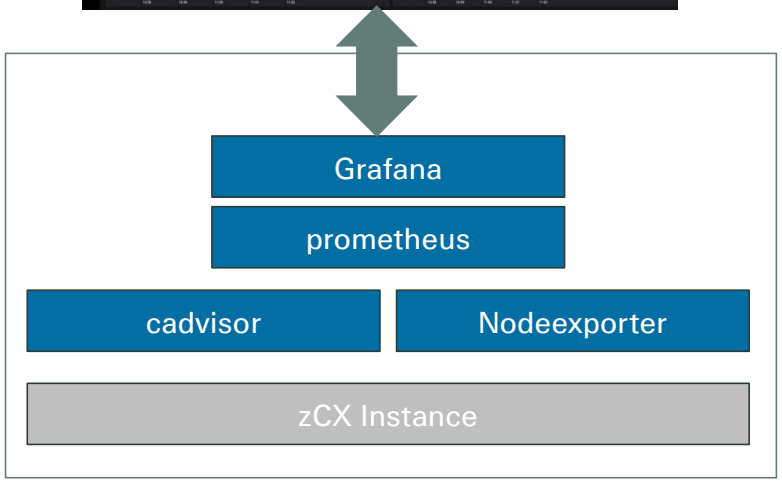
Any application container

Setup and Configuration

- Usage of the zOSMF Workflows is very simple
- Good descriptions in the zOSMF Workflows
- Use to deploy or delete the instance
- Use to reconfigure (amount of disks, keys, certificates, proxy server and so on)
- Use to upgrade the maintenance level of docker

Monitoring

- cadvisor and node-exporter are data provider for Prometheus
- Grafana used to visualize the collected data
- End User can use the Dashboard to check the Instance state and resource consumption



Stability and Performance

- zCX instances are running very stable
  - No issues observed so far with the underlying operating system
- Container performance (perceived)
  - Fast and responsive
  - Image creation

Stepping Stones

- When transferring data/files between zCX and zOS/USS be careful to check on the codepage of the files
  - Use iconv to convert when necessary
- Transfer files in binary mode when possible via sftp
  - scp is not possible with current version due a vulnerability in the open source introduced with OA58341/UJ01425, OA58568/UJ01426, and OA58569/UJ01427
- Missing ‘zCX’ parameter when defining the VIPA

Benefits

- Process Data where it is generated and used
  - Avoid transfer data to an external service to process and send back
- Reduce Latency for connectivity
  - Use internal connectivity through high speed, virtual network (EZAZCX SAMEHOST)
- Remove dependency to external servers
  - Services only bound to Mainframe Services can run in a zCX Container
- Use GDPS and PPRC for data integrity
  - Data automatically duplicated in case a disk subsystem fails
- Run ‘Linux’ Workload on zIIP without the need of an IFL or Linux One

Next Steps

- Introduce IBM SMU for Automation
  - Exploitation ongoing if that product helps to manage all System Automation defined resources and tackle occurred issues quicker.
- Introduce IBM Omegamon TEPS Server
  - This is currently running in a cloud server but would be good to get an IBM Supported image to run the whole service in a docker container
- Introduce JupyterHub Server for our Jupyter Notebooks for Capacity analysis through IzODA
  - Run the existing Notebooks where the data is produced with direct access to stored SMF data in DB2 or from the SMF Buffers directly.
- Introduce Beta Systems Beta_view Server
  - Simple web GUI to access Job Output for developers without the dependency to an external server
- (IBM DB2LUW)

Summary

- Very stable Docker environment as a base
- Missing IBM Support for their products running already on Linux
  - Expecting those products to be available in the same way as IBM SMU for Automation is already available via ServiceLink
- Waiting for Kubernetes implementation (only SWARM supported with the current release)

Any questions?

Useful Information / Links

- DACH zCX Workgroup → Redelf Janssen (IBM)
- Docker CLI -> https://docs.docker.com/engine/reference/commandline/cli/
  - Command Language Interface (consider some restrictions due to the high level of security)
- Docker Hub -> https://hub.docker.com/search?q=&type=image
  - Public Images available (Filter for Images for the IBM Z Architecture)
- IBM Git Repository Project (Open Mainframe Project Ambitus) -> https://github.com/ambitus/linux-containers
  - Sample Docker Files
- Getting started with zOS Container Extension and Docker (Redbook) -> http://www.redbooks.ibm.com/redpieces/abstracts/sg248457.html?Open
  - Stay tuned about new releases
- IBM Secure Registry -> https://ibm.biz/zregeap

Legal notice