z/OS 2.4 IBM Education Assistant (IEA)

Solution (Epic) Name: z/OS Container Extensions (zCX)

Element(s)/Component(s): z/OS Container Extensions

Agenda

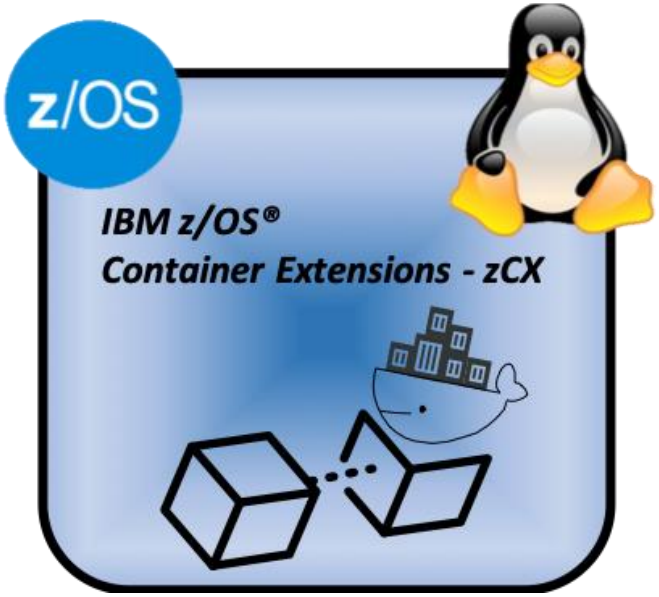
- Trademarks
- Session Objectives
- Overview
- Usage & Invocation
- Interactions & Dependencies
- Migration & Coexistence Considerations
- Installation
- Session Summary
- Appendix

Trademarks

- See url http://www.ibm.com/legal/copytrade.shtml for a list of trademarks.
- Additional Trademarks:
  - Docker and the Docker logo are trademarks or registered trademarks of Docker, Inc. in the United States and/or other countries. Docker, Inc. and other parties may also have trademark rights in other terms used herein.

Session Objectives

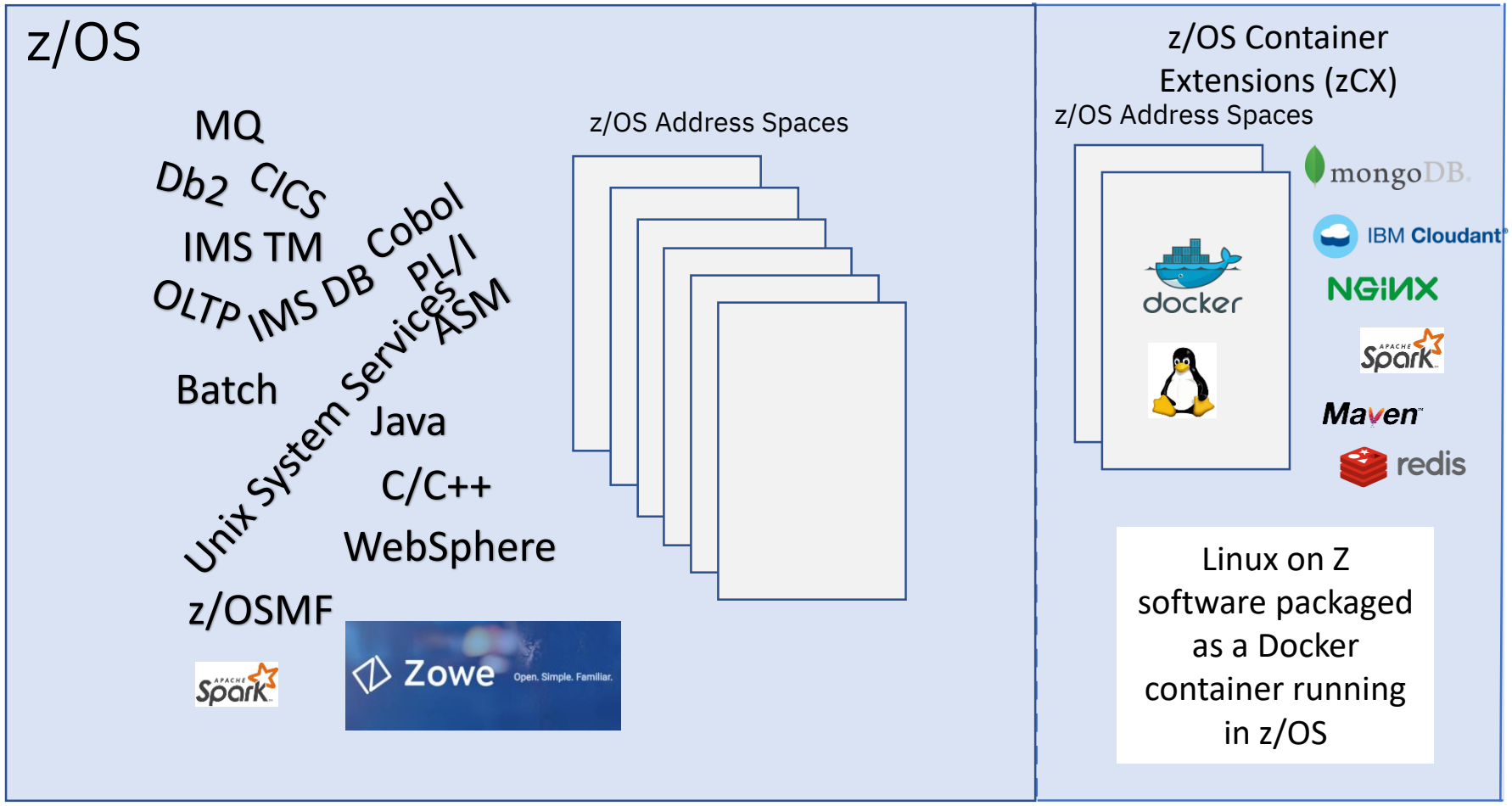
- What is z/OS Container Extensions (zCX)?
- What does it enable you to do?
- How to I get started with zCX?
- How do I manage and monitor zCX?



Overview

- Who
  - z/OS System Programmers, Application/Solution Achitects, Application Programmers, Docker Administrators
- What
  - The ability to deploy unmodified Linux on Z Docker images inside z/OS
- Wow
  - Significantly expanding the z/OS software ecosystem, with z/OS Qualities of Service and without requiring z/OS skills

Expanding the z/OS Software Ecosystem



- Traditional z/OS workloads, middleware, subsystems and programming languages
- Unix System Services provided z/OS with a Unix personality enabling porting of Unix applications and new programming languages to the platform
- z/OS Container Extensions (zCX) provides the next big evolution – unmodified Linux on Z Docker images running inside z/OS

What Is IBM z/OS Container Extensions (zCX)?

A new feature in z/OS 2.4 that enables clients to:

✓ Deploy Linux on Z software components as Docker Containers in a z/OS system, in direct support of z/OS workloads

✓ Without requiring a separately provisioned Linux server

✓ While maintaining overall solution operational control within z/OS and with z/OS Qualities of Service

✓ Requires IBM z14 (or later) based server

Design Thinking Hill Statement:

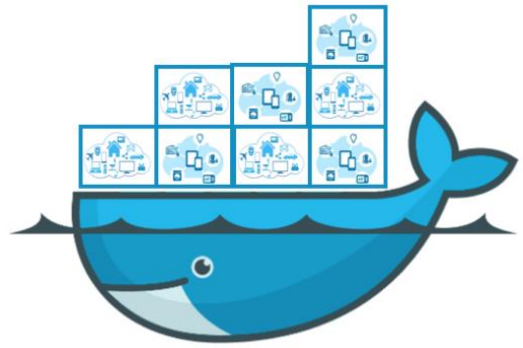
A **solution architect** can **create a solution to be deployed on z/OS based on components available as Docker containers** in the Linux on Z ecosystem transparently exploiting z/OS QoS, **without requiring z/OS development skills**.

IBM zCX - Goals & Qualities of Service

Integrated Disaster Recovery & Planned Outage Coordination	z/OS Storage Resilience	z/OS Networking Virtualization, Security & Availability	z/OS Workload Management, Capacity Planning & Chargeback
Using z/OS DR/GDPS to cover storage used by Linux automatically, integrated restart capabilities for site failures, etc. Integrated Planned Outage Coordination No need to coordinate with non-z/OS administrators when planning a maintenance window, moving workloads to alternate CECs, sites, etc.	Eliminate single points of failure Exploit z/OS VSAM which offers transparent encryption, and failure detection with HiperSwap Configuration validation, I/O health checks, Automatic exploitation zHyperLink and future z/OS Storage enhancements	Support for VIPAs, Dynamic VIPAs allowing for non-disruptive changes, failover, and dynamic movement of the workload. High speed and secure communications with Cross-Memory Virtual Network Interface (SAMEHOST)	WLM: Service Class goals, Business Importance levels, ability to cap resource consumption (CPU and memory) Capacity Provisioning Manager (CPM) support SMF support for accounting and chargeback

What is Docker?

- A Packaging standard for software
  - Think of it like a shipping container
  - Makes moving, stacking, unstacking of compliant software easier
  - Common in the application world on Linux
- Dockerhub
  - www.dockerhub.org contains many docker packages
  - S390x packages support Linux on z
- By focusing on Docker
  - We reduce the complexity of installation and config. for the customer
  - We reduce the service footprint on Linux to what Docker supports
  - We gain access to a large number of packages out of the box



z/OS Container Extensions– A turn-key Virtual Docker Server

Pre-packaged Docker Environment provided by IBM

- Includes Linux and Docker Engine components
- Supported directly by IBM
- Can include clustering and registry capabilities
- Initial focus is on base Docker capabilities
- Competitive price/performance (Workload is zIIP eligible)

Application developers can deploy software using Docker interface

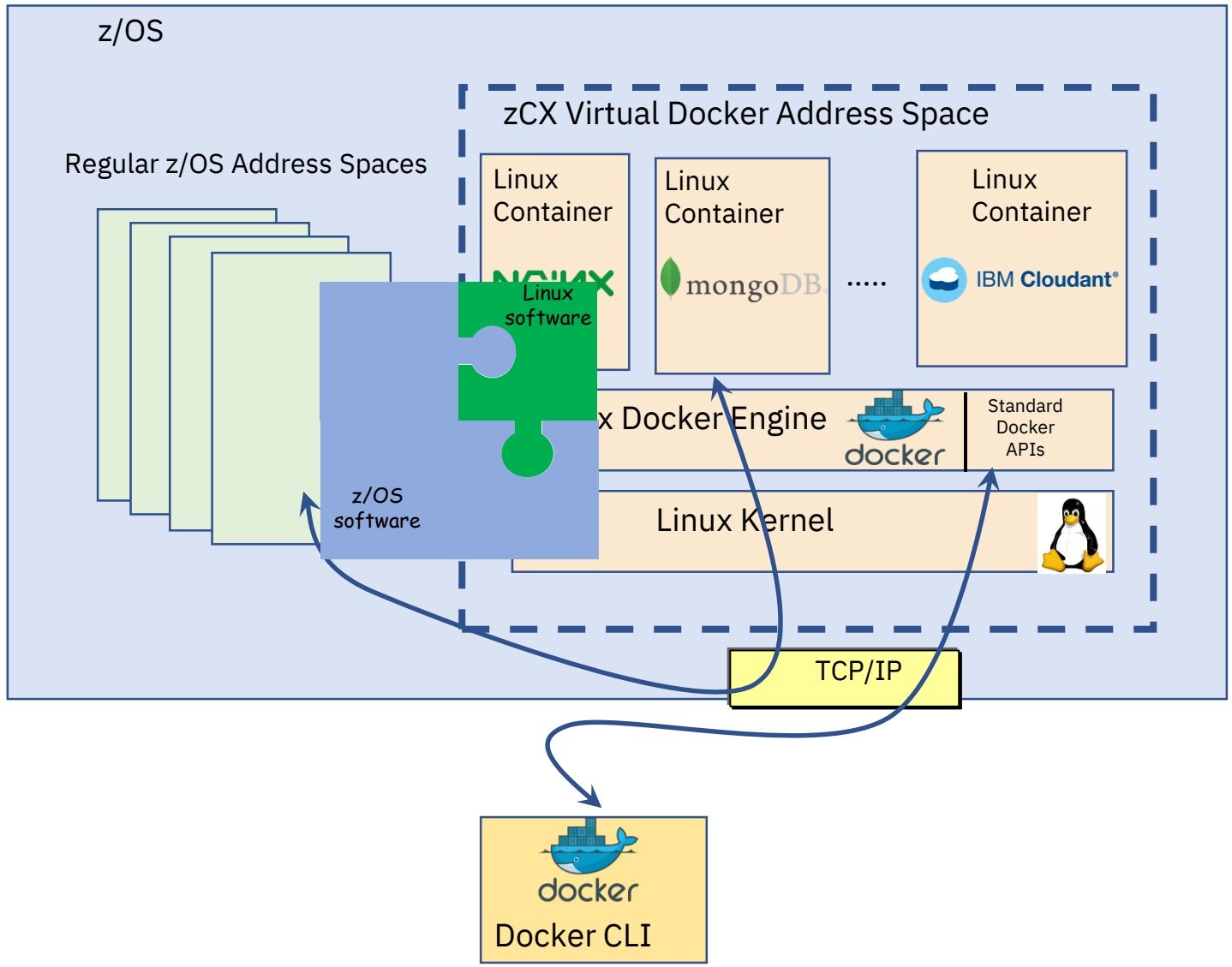
- Any software available as a Docker image on Z System (growing ecosystem available on Docker Hub)
- Any home-grown Linux for Z application packaged as a Docker image
- Using standard Docker interfaces

Limited visibility into Linux environment

- No root access
- Access as defined by Docker interfaces
- Limited Linux administrative overhead

Also provides IBM and ISVs a means of delivering solutions into this environment

- Requires packaging of software as Docker images



Use Cases

Expanding the z/OS software ecosystem for z/OS applications

- Latest Microservices (logstash, Etcd, Wordpress, etc.)
- Non-SQL databases (MongoDB, IBM Cloudant, etc.)
- Analytics frameworks (e.g. expanding the z/OS Spark ecosystem, Cognos components)
- Mobile application frameworks (example: MobileFirst Platform)
- Application Server environments (e.g. IBM BPM, Portal Server, etc.)
- Emerging Programming languages and environments

System Management components

- System management components in support of z/OS that are not available on z/OS
- Centralized data bases for management
- Centralized UI portals for management products – Examples:
  - Tivoli Enterprise Portal (TEPS)
  - Service Management Unite (SMU)

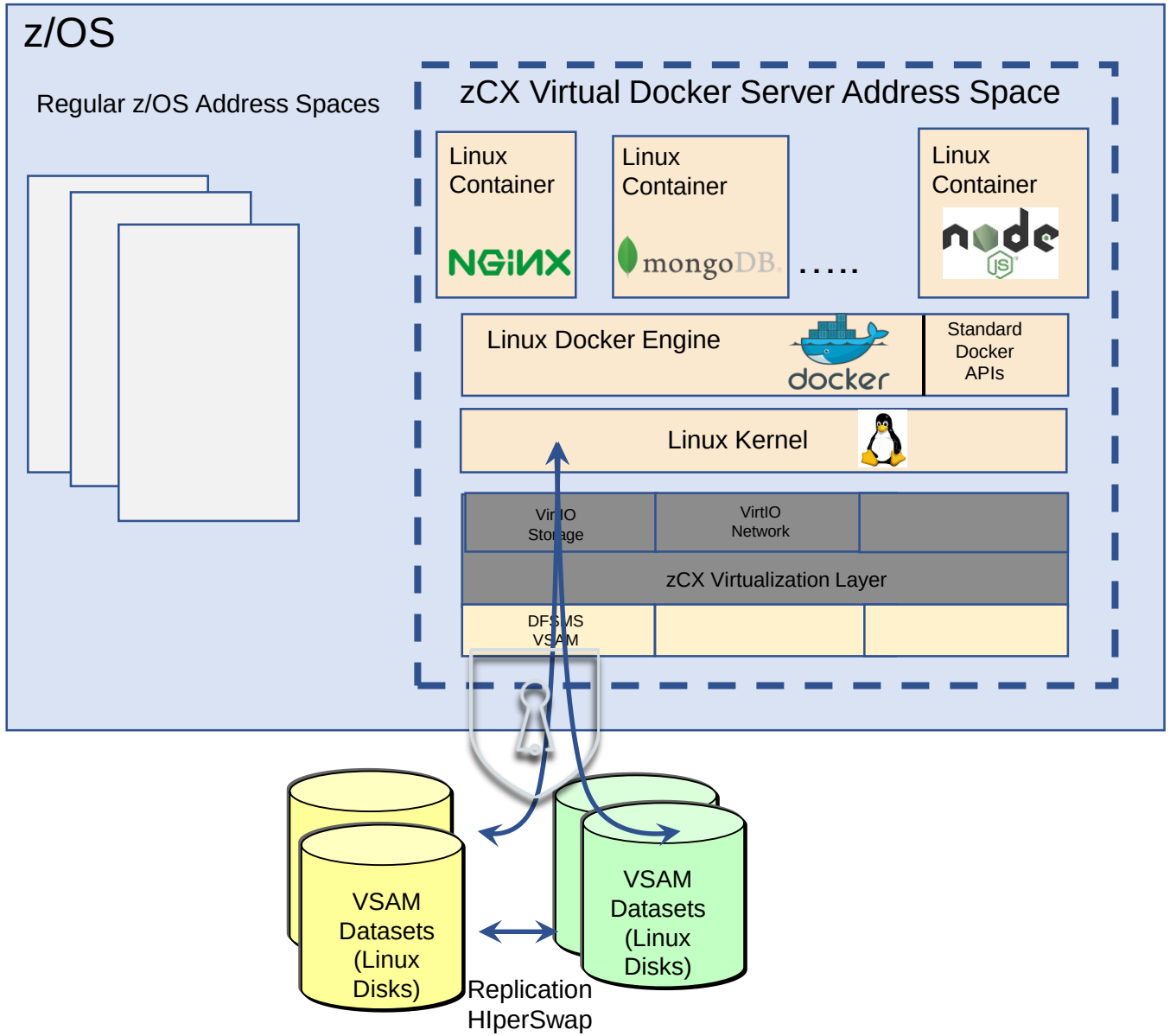
Open Source Application Development Utilities

- Complement existing z/OS ecosystem and Zowe and DevOps tooling
- IBM Application Discovery server component
- IBM UrbanCode Deploy server
- Gitlab/Github server
- Linux based development tools
- Linux Shell environments
- Apache Ant, Apache Maven

Note: The use cases depicted reflect the types of software that could be deployed in IBM zCX in the future. They are not a commitment or statement of software availability for IBM zCX

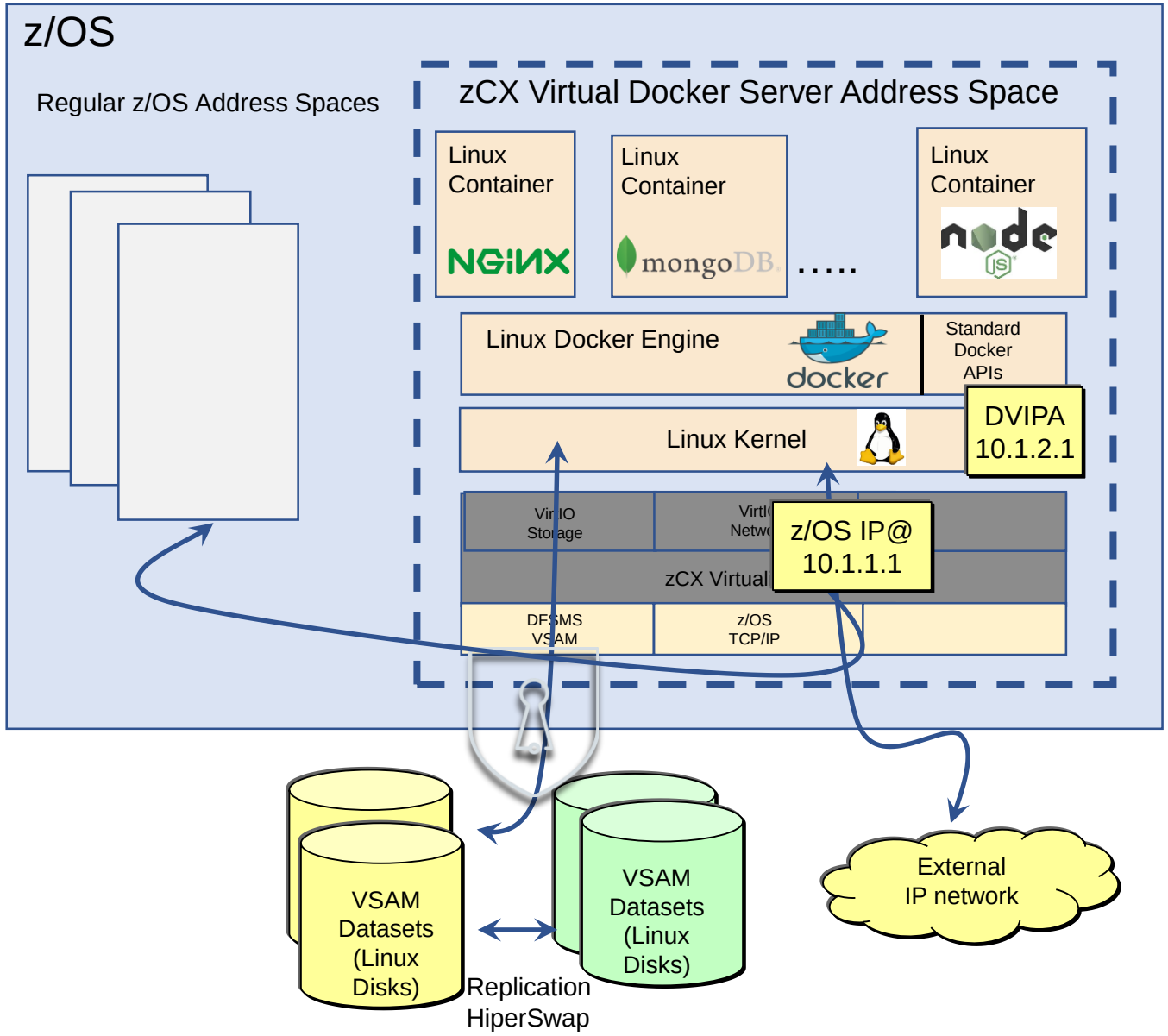
IBM zCX – z/OS Storage Integration

- z/OS Linux Virtualization Layer:
  - Allows virtual access to z/OS Storage, Network
  - Using virtio Linux interfaces
  - Allows us to support unmodified, open source Linux for Z
- Linux storage/disk access (via z/OS owned and managed VSAM datasets)
  - Leverages latest I/O enhancements (e.g. zHyperLinks, I/O fabric diagnostics, etc.)
  - Built-in host-based encryption
  - Replication technologies and HiperSwap



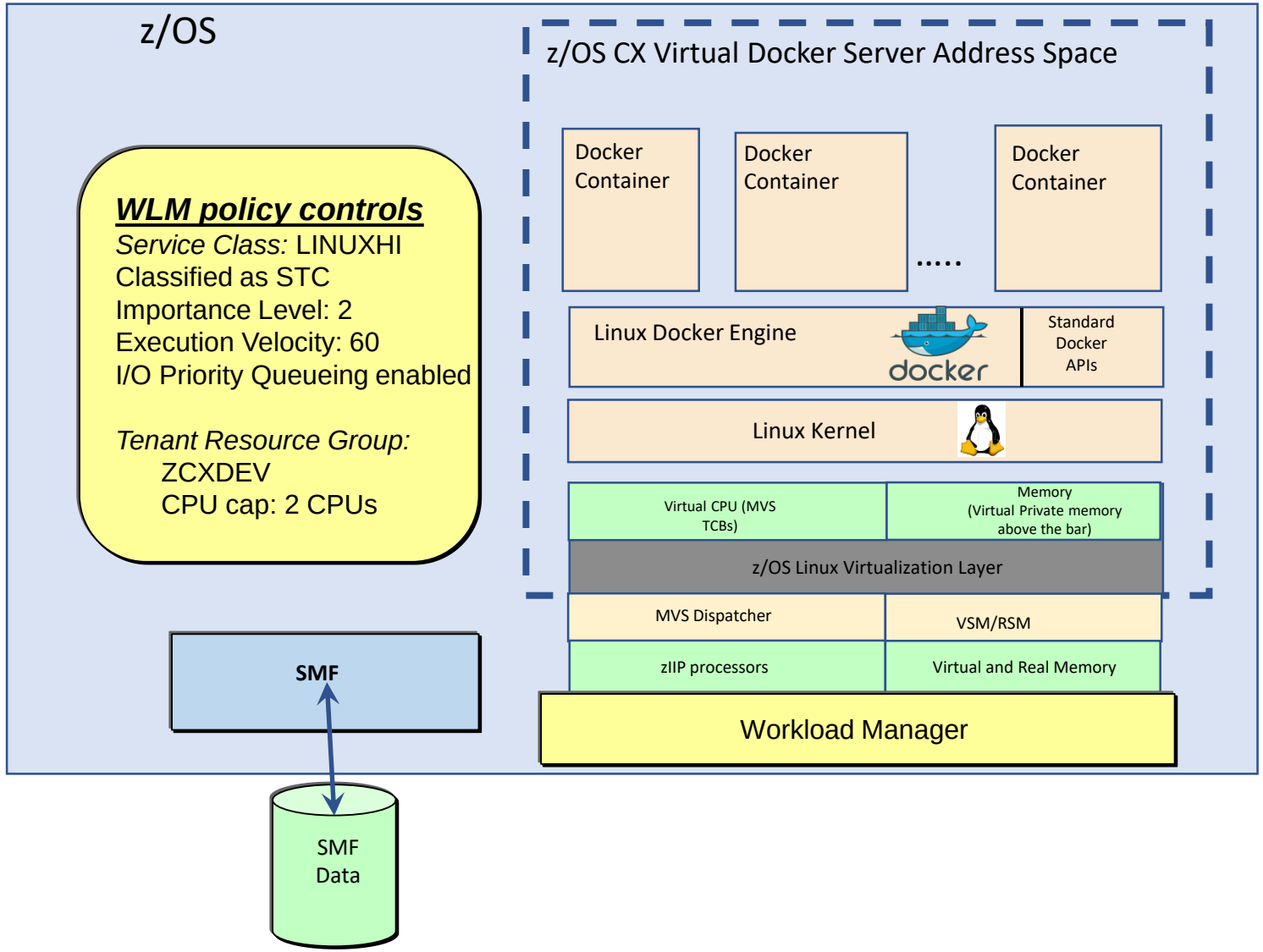
IBM zCX – z/OS Network Integration

- z/OS Linux Virtualization Layer:
  - Allows virtual access to z/OS Storage, Network and Console
  - Using virtio Linux interfaces
    - Stable, well defined interfaces used to virtualize Linux
  - Allows us to support unmodified, open source Linux for z kernels
- Linux network access via high speed virtual *SAMEHOST* link to z/OS TCP/IP protocol stack
  - Each Linux Docker Server represented by a z/OS owned, managed and advertised Dynamic VIPA (DVIPA)
    - Allows restart of a CX instance in another system in the sysplex
  - Provide high performance network access across z/OS applications and Linux Docker containers – leveraging cross memory
  - External network access via z/OS TCP/IP
    - z/OS IP filters to restrict external access



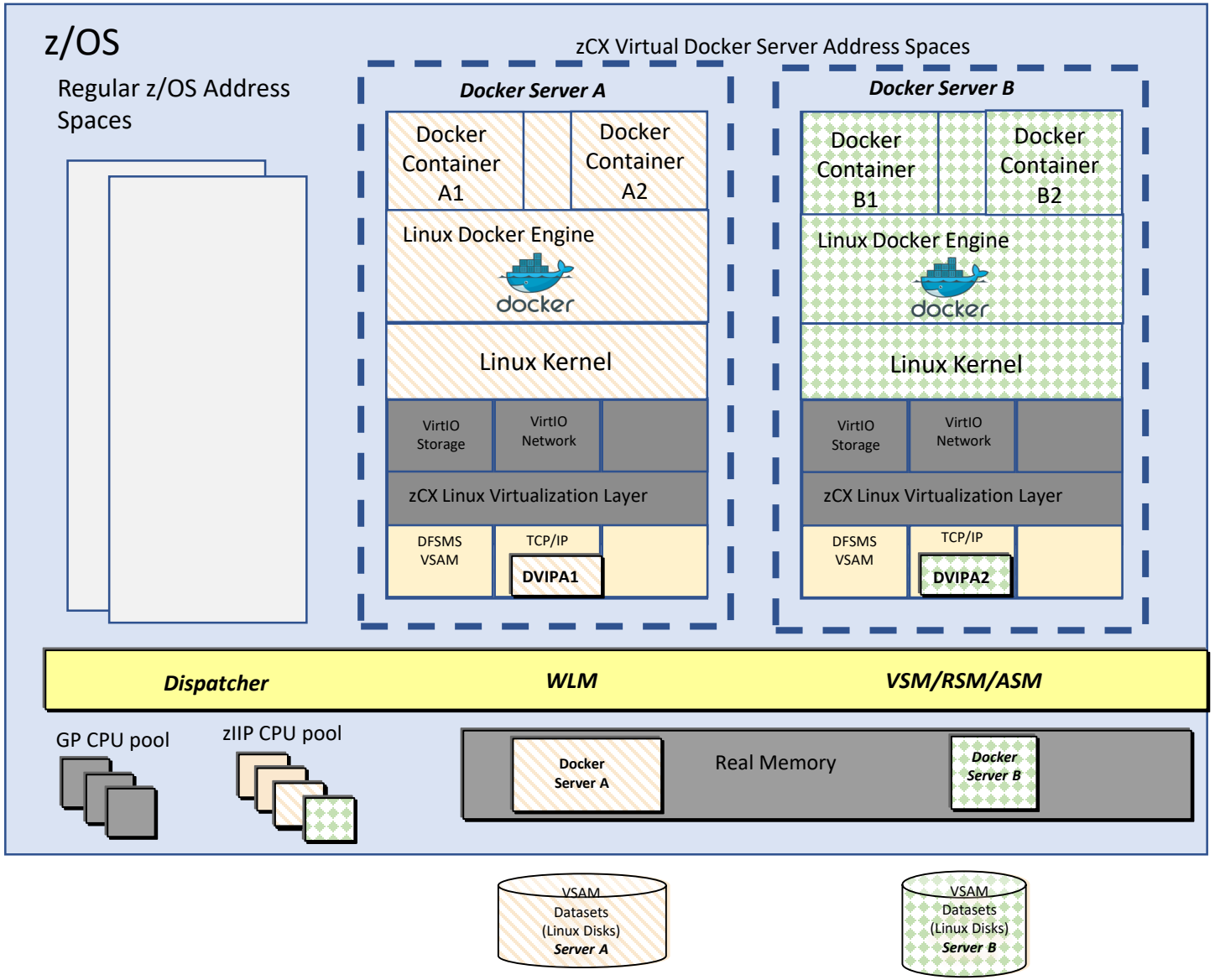
IBM zCX - CPU, Memory and Workload Management

- Memory Management
  - Provisioned per zCX Docker Server address space
  - Private, above the 2GB bar Fixed Memory
  - Managed by VSM, RSM
- CPU Management
  - Virtual CPUs provisioned to each zCX Docker Server address space
    - Each virtual CPU is a dispatchable thread (i.e. MVS TCB) within the address space
    - zIIP CPU access via MVS dispatcher
  - A zCX instance can host multiple Docker Container instances
- Normal WLM policy and resource controls extend to zCX Docker Server address spaces
  - Service Class association, goals and Importance levels
  - Tenant Resource Group association
    - Optional caps for CPU and real memory
- Normal SMF data available
  - SMF type 30, 72, etc.
  - Enables z/OS performance management and capacity planning



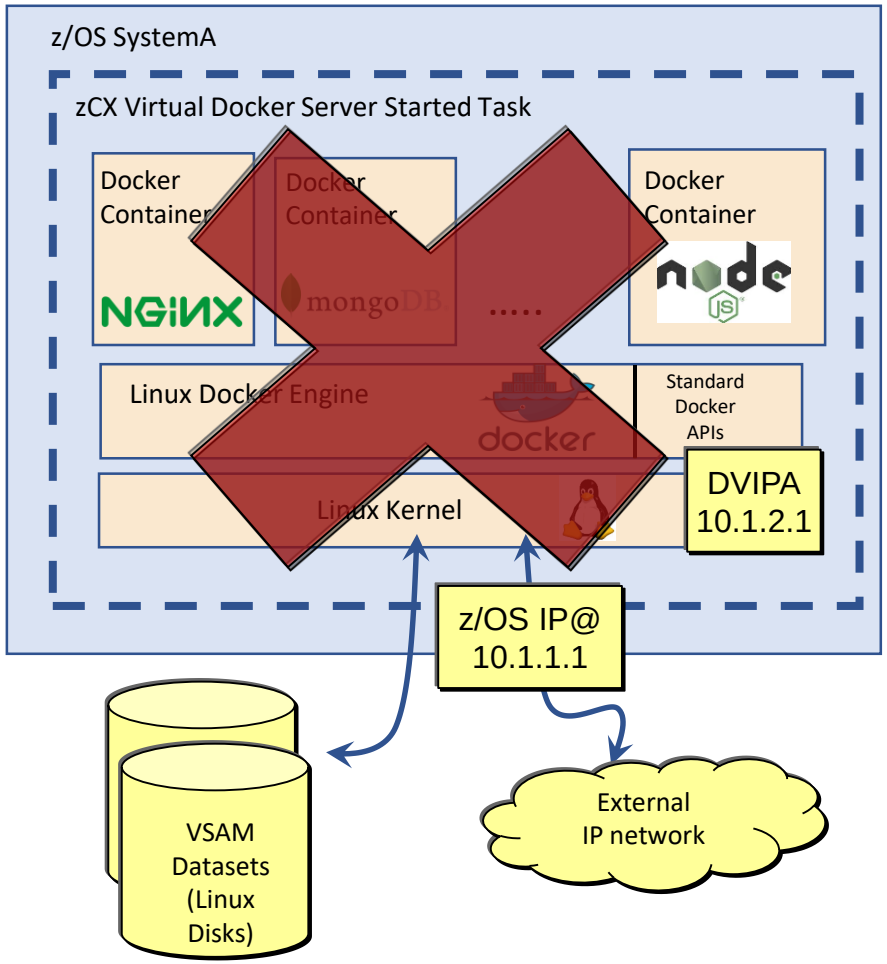
Deploying Multiple zCX Virtual Docker Server Instances

- Multiple zCX instances can be deployed within a z/OS system:
  - Isolation of applications (containers)
  - Different business/performance priorities (i.e. unique WLM service classes)
  - Capping of resources allocated for related workload (CPU, memory, disk, etc.)
- Each zCX address space:
  - Has specific assigned storage, network and memory resources
  - Shares CPU resources with other address spaces
    - But can influence resource access via configuration and WLM policy controls
- A new Hypervisor built using existing z/OS capabilities
  - The z/OS Dispatcher, WLM and VSM/RSM components manage access to CPU and memory
  - The zCX virtualization layer manages Storage, Network and Console access
    - Using dedicated resources
    - There is no communications across z/OS Linux virtualization layer instances
- Integrated z/OS Capacity Provisioning and Management
  - WLM, CPM, adding/removing CPU and Memory resources



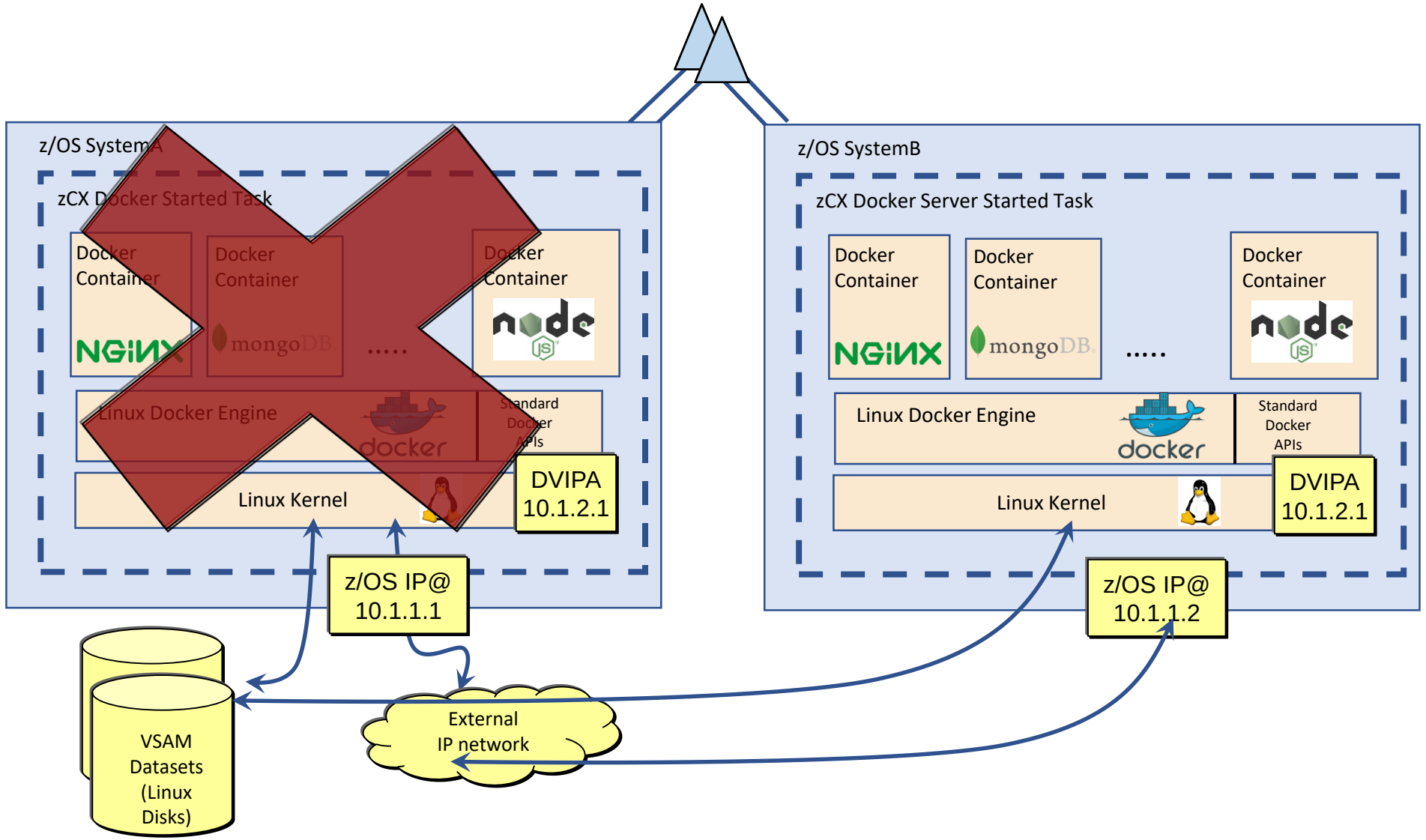
z/OS Container Extensions
Operations and Disaster Recovery Integration

- Started using z/OS Start Command
  - –Support for Start, Stop, Modify
- Automated Operations using z/OS facilities
  - –System Automation
  - –Other z/OS Automation framework/product
- Planned and Unplanned Outage and Disaster Recovery coordination
  - –zCX Docker Server failure (restart in place)



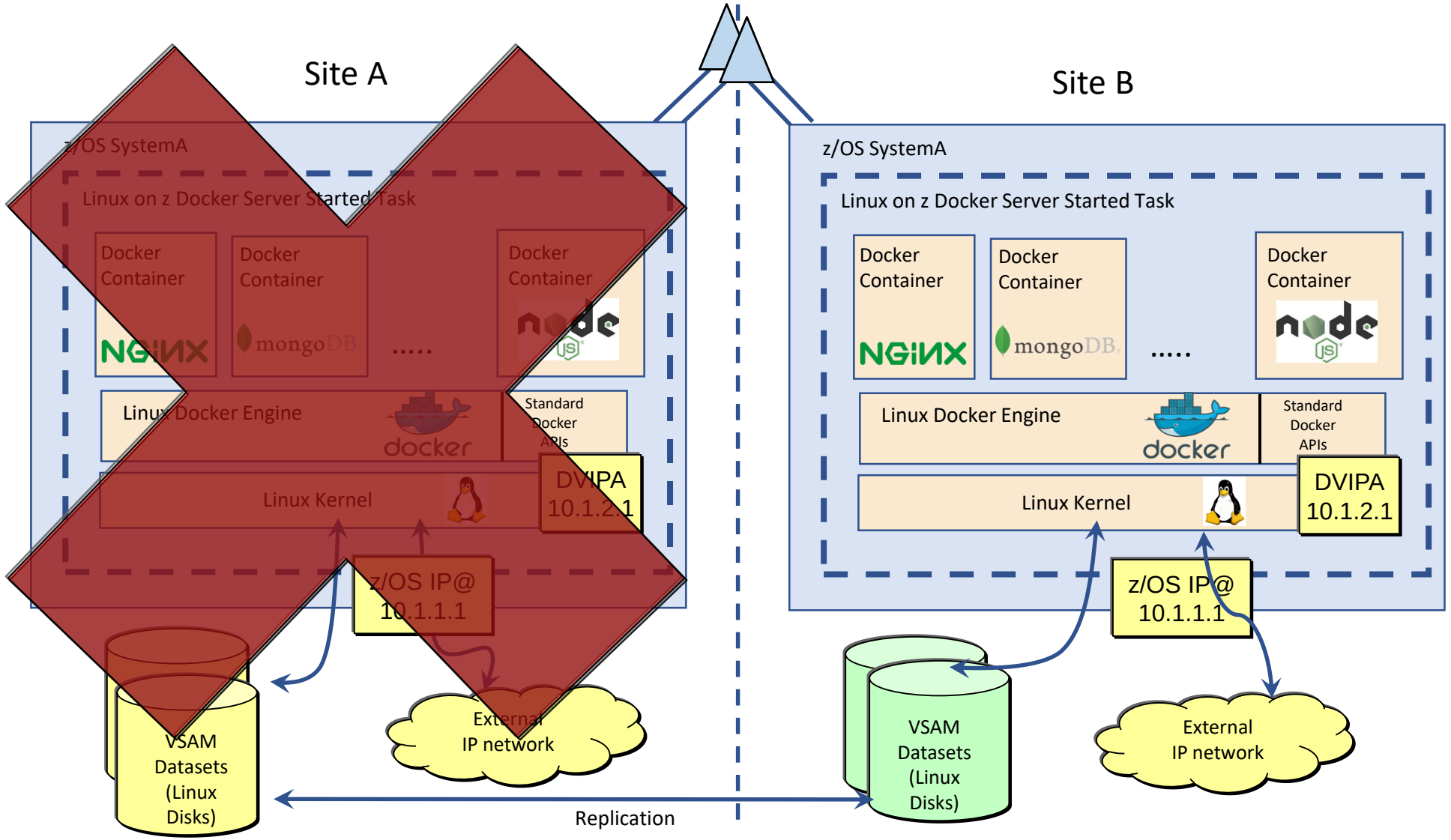
z/OS Container Extensions
Operations and Disaster Recovery Integration

- Started using z/OS Start Command
  - Support for Start, Stop, Modify
- Automated Operations using z/OS facilities
  - System Automation
  - Other z/OS Automation framework/product
- Planned and Unplanned Outage and Disaster Recovery coordination
  - zCX Docker Server failure (restart in place)
  - LPAR failure (restart on other LPAR in the sysplex)



z/OS Container Extensions
Operations and Disaster Recovery Integration

- Started using z/OS Start Command
  - Support for Start, Stop, Modify
- Automated Operations using z/OS facilities
  - System Automation
  - Other z/OS Automation framework/product
- Planned and Unplanned Outage and Disaster Recovery coordination
  - z/OS Container Extensions Docker Server failure (restart in place)
  - LPAR failure (restart on other LPAR in the sysplex)
  - Site failure (restart on alternate site) – GDPS or other automated DR framework



Personas

Priya
Docker Admin

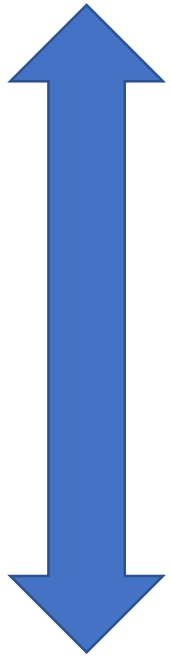
Fred
Application Developer

Shichi
IT Architect

Omar
Solution Architect

Zach
z/OS Systems Programmer
(includes Networking, Storage, Security, WLM, etc. Admins)

More Linux Skill



More z/OS Skill

DISCOVER, TRY, BUY
How do I get it?

GET STARTED
How do I get value?

EVERYDAY USE
How do I get my job done?

MANAGE AND UPGRADE
How do I keep it running?

LEVERAGE AND EXTEND
How do I build on it?

SUPPORT
How do I get unstuck?

DISCOVER, TRY, BUY
How do I get it?

GET STARTED
How do I get value?

EVERYDAY USE
How do I get my job done?

MANAGE AND UPGRADE
How do I keep it running?

LEVERAGE AND EXTEND
How do I build on it?

SUPPORT
How do I get unstuck?

Provisioning

Zach can provision one or more z/OS Container Extensions instances in a z/OS system, each with custom:

- Resource allocation
  - Number of virtual CPUs, memory, network connectivity and storage
- Docker Configuration settings
- Definition of z/OS Container Extensions appliance Docker admin user and specification of LDAP server for authentications/authorization

Resource Allocation:

- zIIP eligible CPUs, resource capping possible via WLM Resource Groups or Tenant Resource Groups
- Support for Fixed z/OS Memory (not pageable), estimated 1GB minimum
- Support for IPv4 and IPv6 network connectivity, Dynamic VIPA (DVIPA support)
- z/OS VSAM LDS for storage with support for encryption and replication

Docker Configuration Options

- Registry to be used
- Logging options

Priya
Docker Admin

Zach
Systems Programmer

DISCOVER, TRY, BUY
How do I get it?

GET STARTED
How do I get value?

EVERYDAY USE
How do I get my job done?

MANAGE AND UPGRADE
How do I keep it running?

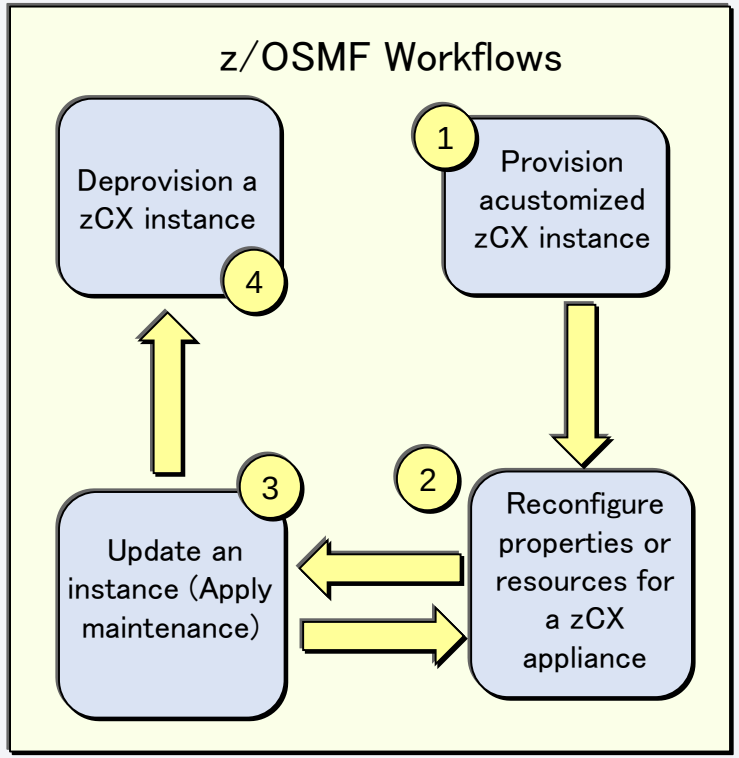
LEVERAGE AND EXTEND
How do I build on it?

SUPPORT
How do I get unstuck?

Provisioning (continued)

Provisioning and deprovisioning and lifecycle management via provided z/OSMF workflows

- Automates many of the steps of provisioning a Container Extensions instance
  - You can provision a zCX instance in a few minutes
- Provides guidance for out of band steps (RACF/SAF resources, TCP/IP network definitions, WLM definitions, DFSMS setup)
- Runs as Started Task, can be started/stopped via operator commands and integrated into automated operations procedures



Zach
Systems Programmer

The Experiences Today

DISCOVER, TRY, BUY
How do I get it?

GET STARTED
How do I get value?

EVERYDAY USE
How do I get my job done?

MANAGE AND UPGRADE
How do I keep it running?

LEVERAGE AND EXTEND
How do I build on it?

SUPPORT
How do I get unstuck?

Docker administrators and permitted Docker users can deploy any Linux on Z docker container image using standard Docker CLI

Access to Docker CLI by remote access into IBM provided and controlled SSHD container environment (included and active in each z/OS Container Extensions instance)

Remote Docker CLI access will not be supported

SSH access to underlying Linux kernel will not be supported

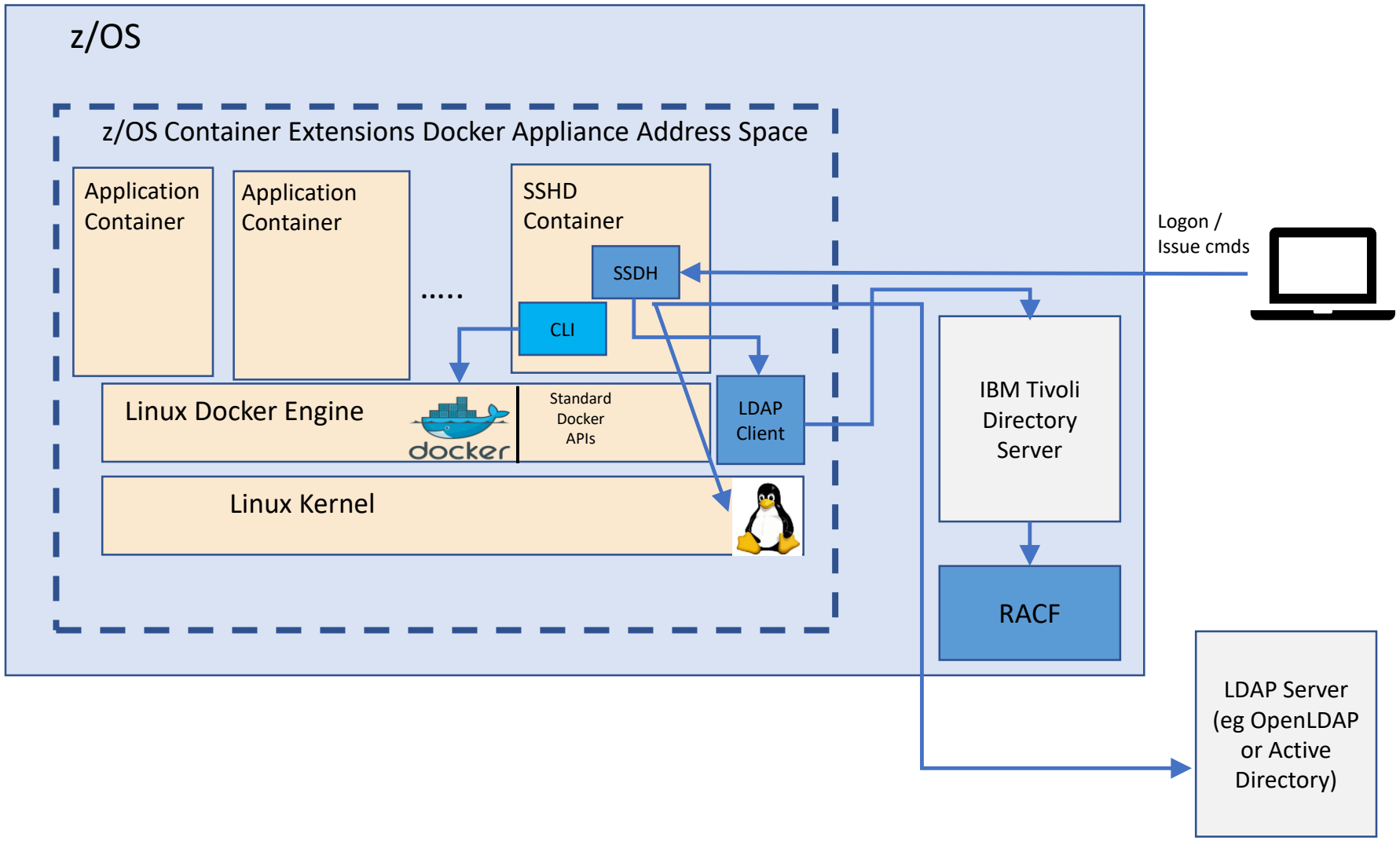
Zach
Systems Programmer

Priya
Docker Admin

Fred
Application Developer

Omar
Solution Architect

User Management and Authentication



3 Options for User management and authentication:

1. Local appliance registry
2. z/OS LDAP Server (IBM Tivoli Directory Server) with RACF integration
3. Remote LDAP server (e.g. OpenLDAP, Active Directory, etc.)

The Experiences Today

DISCOVER, TRY, BUY
How do I get it?

GET STARTED
How do I get value?

EVERYDAY USE
How do I get my job done?

MANAGE AND UPGRADE
How do I keep it running?

LEVERAGE AND EXTEND
How do I build on it?

SUPPORT
How do I get unstuck?

Docker CLI (Command Line Interface)

https://docs.docker.com/engine/reference/commandline/docker/

Standard Docker CE command line interface (Current Docker level: 18.06.1-ce – will change as we go forward)

docker

Estimated reading time: 3 minutes

Description

The base command for the Docker CLI.

Child commands

Command	Description
docker attach	Attach local standard input, output, and error streams to a run
docker build	Build an image from a Dockerfile
docker builder	Manage builds
docker checkpoint	Manage checkpoints
docker commit	Create a new image from a container's changes
docker config	Manage Docker configs
docker container	Manage containers
docker cp	Copy files/folders between a container and the local filesystem
docker create	Create a new container
docker deploy	Deploy a new stack or update an existing stack
docker diff	Inspect changes to files or directories on a container's filesyste
docker engine	Manage the docker engine
docker events	Get real time events from the server
docker exec	Run a command in a running container
docker export	Export a container's filesystem as a tar archive
docker history	Show the history of an image

docker export	Export a container's filesystem as a tar archive
docker history	Show the history of an image
docker image	Manage images
docker images	List images
docker import	Import the contents from a tarball to create a filesystem image
docker info	Display system-wide information
docker inspect	Return low-level information on Docker objects
docker kill	Kill one or more running containers
docker load	Load an image from a tar archive or STDIN
docker login	Log in to a Docker registry
docker logout	Log out from a Docker registry
docker logs	Fetch the logs of a container
docker manifest	Manage Docker image manifests and manifest lists
docker network	Manage networks
docker node	Manage Swarm nodes
docker pause	Pause all processes within one or more containers
docker plugin	Manage plugins
docker port	List port mappings or a specific mapping for the container
docker ps	List containers
docker pull	Pull an image or a repository from a registry
docker push	Push an image or a repository to a registry
docker rename	Rename a container
docker restart	Restart one or more containers
docker rm	Remove one or more containers

Demo of z/OSMF workflows and Deploying Docker Containers

The Experiences Today

DISCOVER, TRY, BUY
How do I get it?

GET STARTED
How do I get value?

EVERYDAY USE
How do I get my job done?

MANAGE AND UPGRADE
How do I keep it running?

LEVERAGE AND EXTEND
How do I build on it?

SUPPORT
How do I get unstuck?

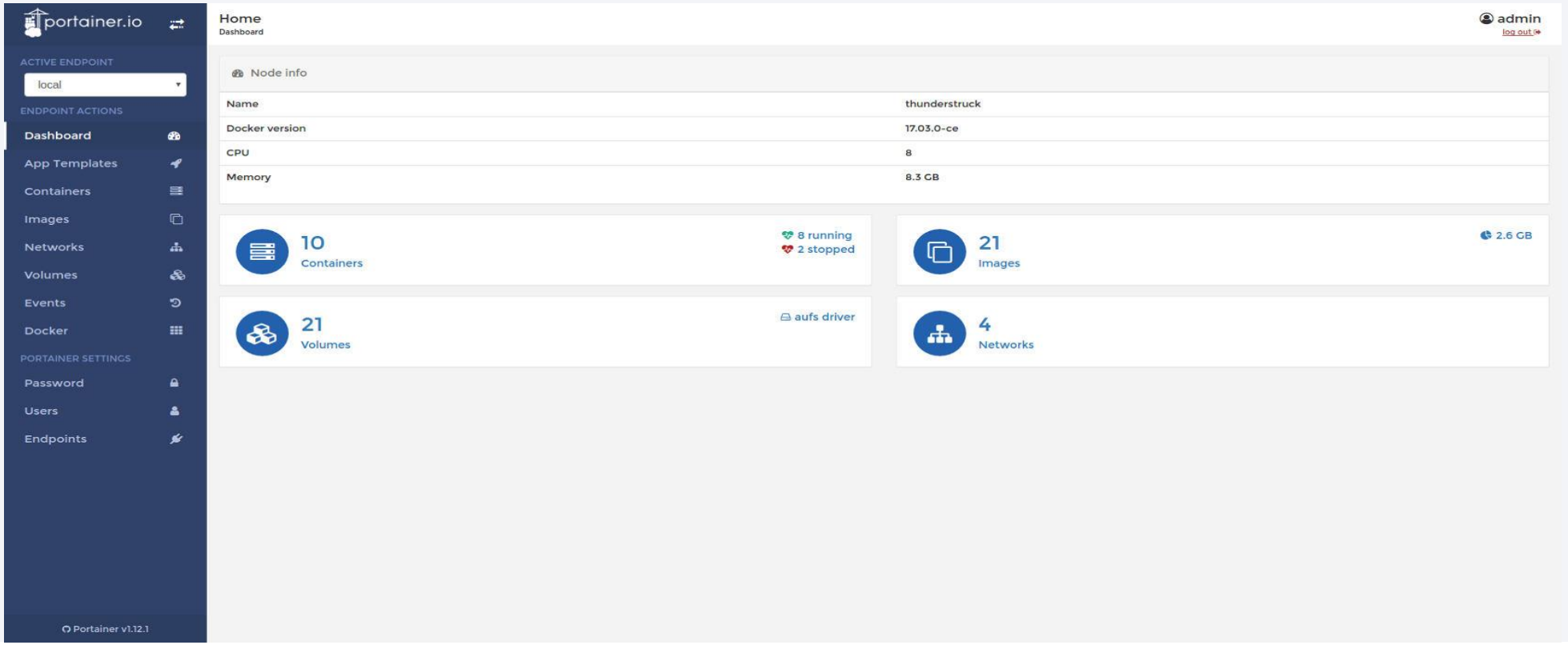
Graphical user interface access to Docker

z/OS Container Extensions Docker Administrators can deploy Portainer Daemon container for s390x (from Dockerhub) as an additional or alternative interface to the Docker CLI for specific Docker users

- Instructions on Deploying Portainer on zCX are included in the IBM z/OS Container Extensions Guide

portainer.io

Permitted Portainer users can use the graphical interface to deploy and manage Docker containers in a z/OS Container Extensions instance



DISCOVER, TRY, BUY
How do I get it?

GET STARTED
How do I get value?

EVERYDAY USE
How do I get my job done?

MANAGE AND UPGRADE
How do I keep it running?

LEVERAGE AND EXTEND
How do I build on it?

SUPPORT
How do I get unstuck?

Clustering and Orchestration

Permitted z/OS Container Extensions Docker users create a Swarm cluster of z/OS Container Extensions instances using standard Docker CLI

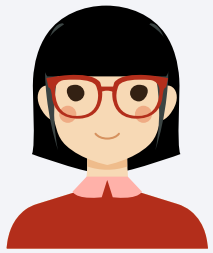
Permitted z/OS Container Extensions Docker users can deploy Docker containers in a z/OS Container Extensions Swarm cluster using standard Docker CLI

Future support being explored:

- Kubernetes and IBM Cloud Private
- Other Orchestration solutions?

kubernetes

IBM Cloud Private



Shichi
IT Architect

Omar
Solution Architect

Zach
Systems Programmer

Priya
Docker Admin

Fred
Application Developer

DISCOVER, TRY, BUY
How do I get it?

GET STARTED
How do I get value?

EVERYDAY USE
How do I get my job done?

MANAGE AND UPGRADE
How do I keep it running?

LEVERAGE AND EXTEND
How do I build on it?

SUPPORT
How do I get unstuck?

Monitoring z/OS Container Extensions instances

Docker administrators can deploy and use open source and ISV Docker Container images for Linux on Z (s390x images) to monitor overall server and container resource utilization

Examples of Open Source Docker images tested with z/OS Container Extensions

- Prometheus: Open source monitoring and alerting solution based on time series database
  - Flexible query language
  - System and application level monitoring
  - Collects metrics from instrumented targets
- Grafana: Open source metrics analytics and visualization tool
  - Support for Prometheus as a data source (among others)
  - Provides easy to build dashboards for visualizing system and application metrics
- cAdvisor: Monitors container based environments
  - Collects metrics at container and system level
  - Can act as a data source for Prometheus and provides its own UI
- Prometheus Node Exporter: Acts as a data source for system level metrics for Prometheus

Zach
Systems Programmer

Priya
Docker Admin

Demo of Grafana

Note: Instructions on deploying Prometheus, Grafana, cAdvisor and Prometheus Node Exporter included in the IBM z/OS Container Extensions Guide

DISCOVER, TRY, BUY
How do I get it?

GET STARTED
How do I get value?

EVERYDAY USE
How do I get my job done?

MANAGE AND UPGRADE
How do I keep it running?

LEVERAGE AND EXTEND
How do I build on it?

SUPPORT
How do I get unstuck?

Monitoring and Managing z/OS Container Extensions Address Spaces

System Programmers, Operators and Performance Analysts can use standard available z/OS Monitoring tools and facilities to monitor z/OS Container Extensions address spaces

zCX address spaces are started tasks that have standard address space metrics and SMF records available that can be used for monitoring and chargeback

- Type 30 and type 72 records are available
- Other SMF records related to z/OS Container Extensions address space resource utilization also available

z/OS WLM policies and facilities can be used to dynamically adjust service class goals and importance levels for zCX appliances

Zach
Systems Programmer
And Operations Staff

DISCOVER, TRY, BUY
How do I get it?

GET STARTED
How do I get value?

EVERYDAY USE
How do I get my job done?

MANAGE AND UPGRADE
How do I keep it running?

LEVERAGE AND EXTEND
How do I build on it?

SUPPORT
How do I get unstuck?

Diagnosing problems with z/OS Container Extensions

- IBM z/OS support process can be used to help diagnose and address problems with the underlying z/OS Container Extensions implementation, including problems in:
  - Virtualization layer
  - Docker appliance and Linux kernel layers
- Problems with software deployed as containers pursued using existing channels (IBM, ISV, Open Source, etc.)

Facilities provided:

- First Failure Data Capture (FFDC)
- Facilities to allow on demand capture of z/OS Container Extensions appliance or virtualization diagnostic data
- CTRACE and z/OS Dump support for virtualization layer
- Ability to extract z/OS Container Extensions appliance diagnostics and dump data

More Details will be made available in the IBM z/OS Container Extensions Guide

Zach
Systems Programmer

And z/OS Operations staff

Priya
Docker Admin

Interactions & Dependencies

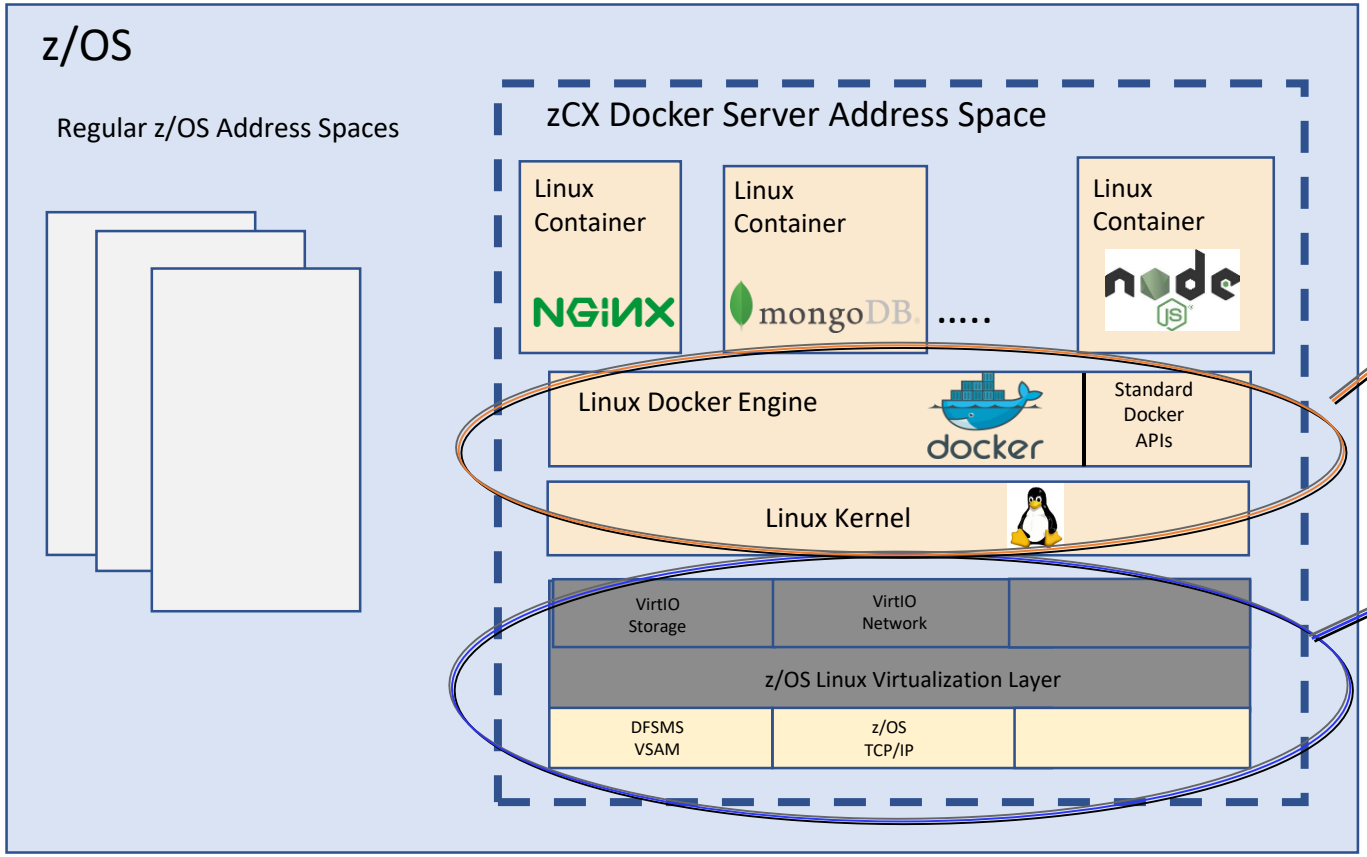
- To exploit this item, all systems in the Plex must be at the new z/OS level: No (but all systems that you want to deploy zCX on must be at the z/OS V2R4 level)
- Software Dependencies
  - N/A
- Hardware Dependencies
  - Z14 GA2
  - Hardware Feature Code 0104 (Container Hosting Foundation)
    - Provides service and support for underlying Linux kernel in zCX appliance
    - Covers unlimited zCX usage in the CPC (across all LPARs)
    - If you already have deployed IBM Secure Service Container for IBM Cloud Private then you likely already have this hardware feature installed (and it will also cover zCX use)
    - Can be ordered on the IBM Z14 from the eConfig fulfillment system.
    - This will be a requirement for running zCX at GA
    - This requirement has been relaxed during the V2R4 ESP time frame
      - The run-time check is still performed and you will see the following message during zCX start up:
- Exploiters
  - N/A

Migration & Coexistence Considerations

- Given that this is a new function there are no migration considerations
- All systems in a Sysplex that will be hosting zCX instances must be at the V2R4 level

Installation

z/OS Container Extensions – How is it structured?



zCX Appliance: Shipped as a new element of z/OS
in new FMID: HZDC7C0 which includes:
- Linux Kernel & Docker Engine: 5752-SCCDE
- zOSMF Workflow: 5752-SCCWF

3 character prefix AZD

Virtualization Layer: Shipped in z/OS Base
in BCP Base HBB77CO
as new parts
as new Comp ID 5752-SCCON
3 character prefix GLZ

Installation (cont)

- Assumption: All program directory steps for z/OS Container Extensions (allocating a separate ZFS and installing z/OS Container Extensions) have been completed
- What's next?
  - How do I get started? Post-install steps

Post-Install Steps – Getting Ready to Deploy zCX

Getting Ready to Deploy zCX – z/OSMF

1. z/OSMF is active in a system on the Sysplex you are deploying zCX on
2. All systems in the Sysplex that you wish to deploy zCX to have been defined in z/OSMF (under z/OSMF Settings – Systems Option)
3. All z/OS System Programmers that will be provisioning, deprovisioning, configuring or updating zCX instances must have access to z/OSMF and also access to z/OSMF workflows
   - They may also require access to the z/OSMF workflow editor if they wish to customize the provided zCX workflows
   - See “Resource authorizations for the z/OSMF core functions” for more information:

https://www.ibm.com/support/knowledgecenter/SSLTBW_2.3.0/com.ibm.zos.v2r3.izua300/izuconfig_SecurityStructuresForZosmf.htm#DefaultSecuritySetupForZosmf__SecuritySetupRequirementsForCoreFun

Getting Ready to Deploy zCX
Determining Naming conventions

- Each zCX instance is a started task (STC) that requires a unique jobname in the Sysplex
  - If you plan on deploying multiple zCX instances in the Sysplex it would be wise to plan a naming convention ahead of time
  - Normal z/OS Jobname syntax requirements apply
- Each zCX instance will require a set of VSAM data sets to be allocated
  - You get to define the HLQ for these data sets – up to 28 characters
  - The provisioning workflows will then allocate datasets using that HLQ
    - Format: HLQ.zcxinstancename.suffix
    - The suffix is generated by zCX workflows and designates the function of the data set
    - Example – HQL: “OMVSSPA.ZCX.PROD”, zCX Instance Jobname: “ZCXGUS1”
    - Provisioned datasets:

      OMVSSPA.ZCX.PROD.ZCXGUS1.CONF
      OMVSSPA.ZCX.PROD.ZCXGUS1.DATA1
      OMVSSPA.ZCX.PROD.ZCXGUS1.DLOG1
      OMVSSPA.ZCX.PROD.ZCXGUS1.ROOT
      OMVSSPA.ZCX.PROD.ZCXGUS1.SWAP1
      OMVSSPA.ZCX.PROD.ZCXGUS1.ZFS
  - You should consider having unique HLQs for each set of zCX instances that have unique users or admins (for example: PROD vs TEST)

Getting Ready to Deploy zCX
Preparing the Started Procedure(s) for zCX and CTRACE parmlib member

- Define a GLZ started procedure in your PROCLIB concatenation
  - zCX provides a sample GLZ procedure in SYS1.SAMPLIB
  - This sample allows you to have a single started proc for all instances of zCX
    - The location of the configuration file and jobname for each zCX instance can be specified on the Start command using the CONF= and JOBNAME= parameters
- You can also have multiple started procedures
  - Customized to eliminate the need to specify the CONF=,JOBNAME= parameters on the START command

zCX includes CTRACE support for tracing events in the Virtualization layer

- Sample parmlib member CTIGLZ00 shipped in SYS1.SAMPLIB
  - Copy to your parmlib dataset
  - Note down the name (needed for the provisioning workflow)

Getting Ready to Deploy zCX
Base Security Requirements

- Each zCX STC instance will need a userid to execute under
  - You need to determine whether you will be using a single userid for a group of zCX STC instances or have each zCX instance associated with a unique userid
    - This will depend on your security practices
    - You may chose to have common userids associated with a set of zCX STC instances
    - You may chose to have a unique userid for each zCX STC instance
  - You will need to define the appropriate STARTED CLASS definitions to associate the userids with the proper Started Task Procedure for zCX

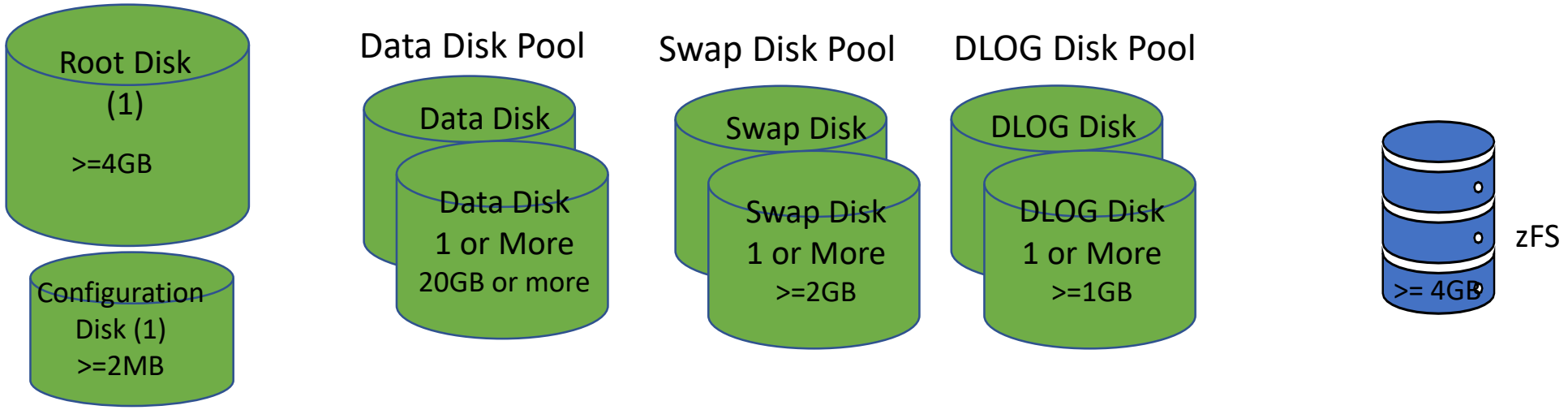
- Protecting HLQ for zCX data sets
  - Use RACF (or alternative Security Manager product) data set profiles to properly protect the zCX datasets – the following userids will require access to these datasets:
    - z/OS System Programmers provisioning, deprovisioning, modifying and configuring zCX instances: These userids will require ALTER access to these HLQ
    - The userids associated with the zCX Started Tasks will require CONTROL access to these data sets
    - Defining these sets of userids in different Groups may make the security administration easier to manage

Getting Ready to Deploy zCX
Base Security Requirements (continued)

- z/OS System Programmers provisioning/deprovisioning/configuring zCX will also require access to zFS file systems
  - Permissions to allocate, mount, unmount, and delete zFS filesystems for the zCX instance directory
  - Set up non-privileged user mount
    - UNIXPRIV class resource, and provide z/OS provisioning users access to the SUPERUSER.FILESYS.USERMOUNT resource
    - For more information, see
      https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.3.0/com.ibm.zos.v2r3.bpxb200/nonprivm.htm

Getting Ready to Deploy zCX
Planning for Datasets and zFS requirements (cont)

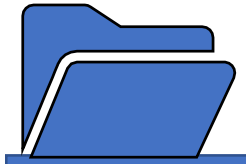
- Each zCX instance requires multiple VSAM datasets to be allocated for its exclusive use
  - zCX Appliance Linux Disks
    - These are VSAM LDS
    - Primary Extents only – datasets are fully allocated at provisioning time
    - Requires Data Class (DATACLASS) with Extended Format Required and Extended Addressability Enabled to allow >4GB datasets to be allocated
    - Define appropriate STORCLASS that will encompass the set of volumes eligible for zCX Appliance Instances
    - Define MGTCLASS as per installation practices
    - Define ACS routines to associate zCX datasets with appropriate DATACLASS, STORCLASS, MGTCLASS (per installation practices) or note down these classes for direct specification in z/OSMF zCX workflows
  - zCX Instance ZFS – One per instance (supports primary and secondary extents)
    - Can use the same Dataclass/Storclass/Mgtclass specifications as the zCX Appliance Linux Disks



Getting Ready to Deploy zCX
Planning for Datasets and zFS requirements (cont)

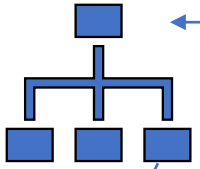
Dataset type	What is it used for	Size requirements	Can additional space be added?
Root disk	Linux root file system	>= 4GB	No – only through deprovisioning and reprovisioning
Configuration disk	Holds configuration data for zCX appliance	>=2MB	No – only through deprovisioning and reprovisioning
Data disk	Holds all docker images, containers, logs and volumes	>=20GB recommended (workload dependent)	Yes – additional disks can be added to data pool (zCX recycle required)
Swap disks	Used by Linux kernel for paging/swapping when virtual memory exceeds real memory	>=2GB minimum (workload dependent)	Yes – additional disks can be added to data pool (zCX recycle required)
Diagnostics and Logs (DLOGS)	Used to hold diagnostic data, logs and FFDC information	>=1GB minimum	Yes – additional disks can be added to data pool (zCX recycle required)
Instance zFS	Used to hold zCX Appliance image, configuration file, etc.	>=4GB	Can be expanded by secondary extents
Total space per zCX instance (minimum)		33GB	
Total space for all zCX instances		Number of zCX instances * 33GB	

Getting Ready to Deploy zCX
Data sets and zFS requirements



Installation Directory

- High-Level Qualifier for VSAM Datasets



zCX Instance Registry

zCX Instance Directory

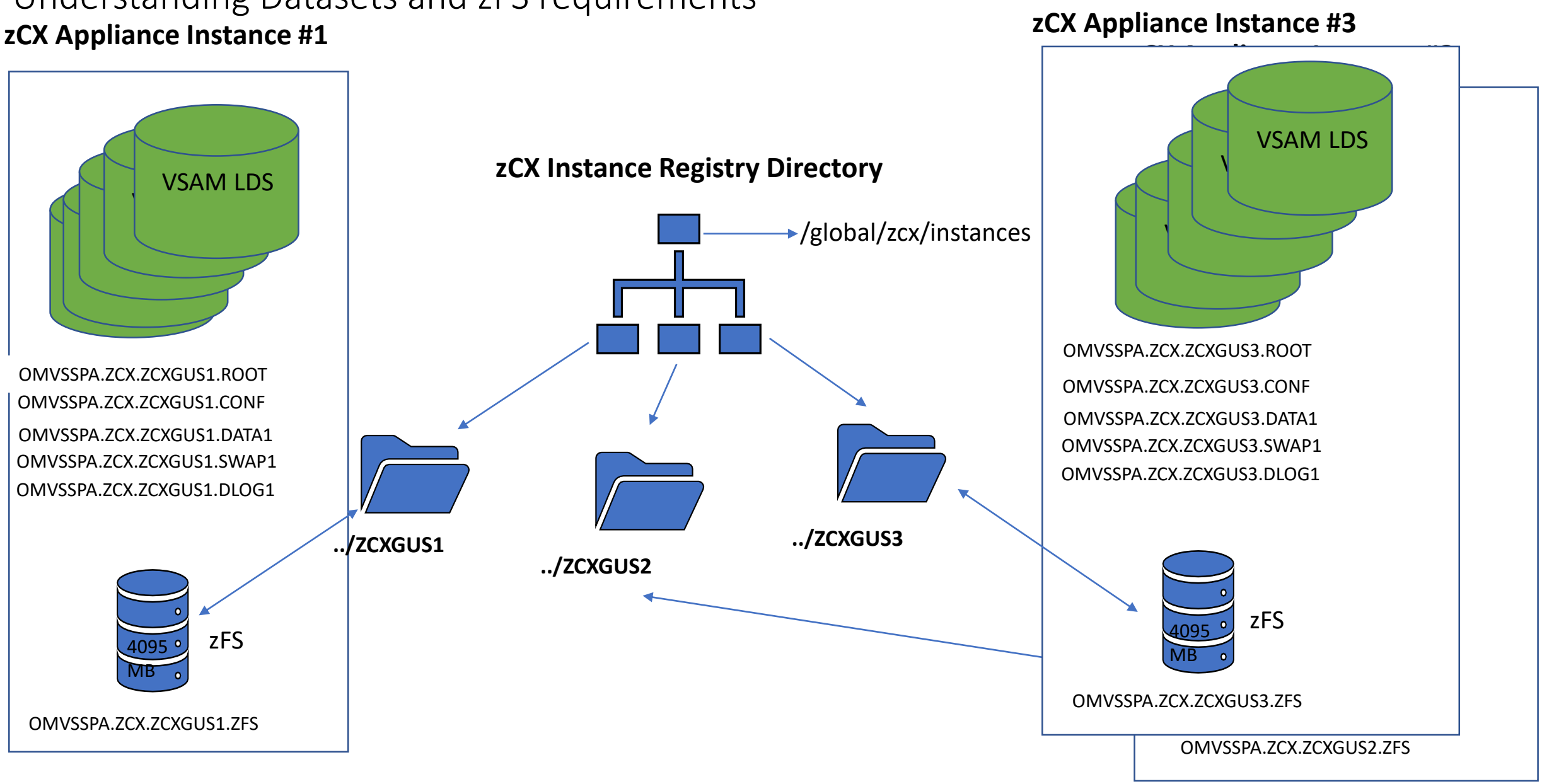
For Example:

- Where zCX binaries and workflows are installed
- /usr/lpp/zcx_zos (default)
- OMVSSPA.ZCX
- A z/OS Sysplex shared UNIX directory where zCX Appliance instances related records are kept
- Recommended: /global/zcx_zos/instances
- A unique directory will be created per zCX instance
- A new zFS filesystem will be allocated and mounted

Requirements:

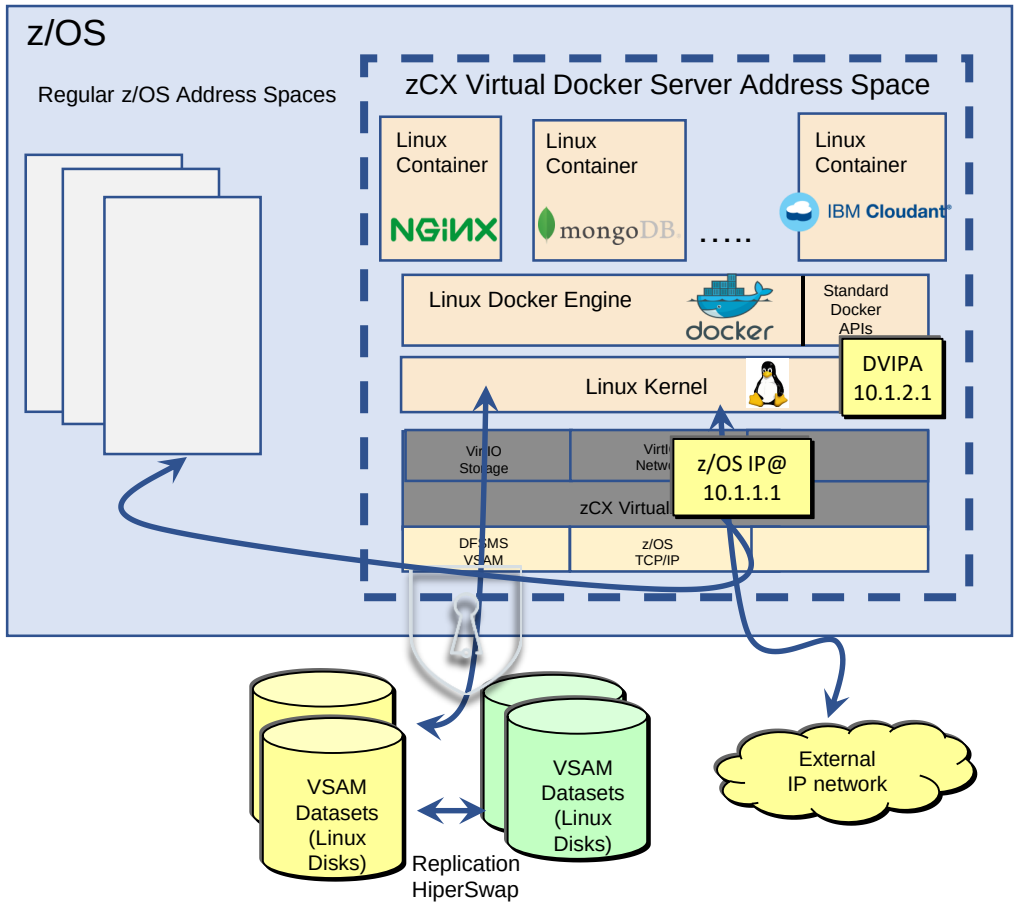
- Can specify up to 28 characters including periods
- Common location to all zCX instances in the z/OS sysplex
- Sysplex aware zFS (to support zCX instances on multiple systems or restart of a zCX instance on another system)
- Must be allocated before using workflows
- Absolute directory path can be up to 30 characters long
- Directory name is constructed using zCX provided instance name (i.e. Jobname)
- ../ZCXGUS1
- BPXPRMxx updates required to mount these ZFS instances after IPL (exploring alternative method of using automount functionality)**

Getting Ready to Deploy zCX
Understanding Datasets and zFS requirements

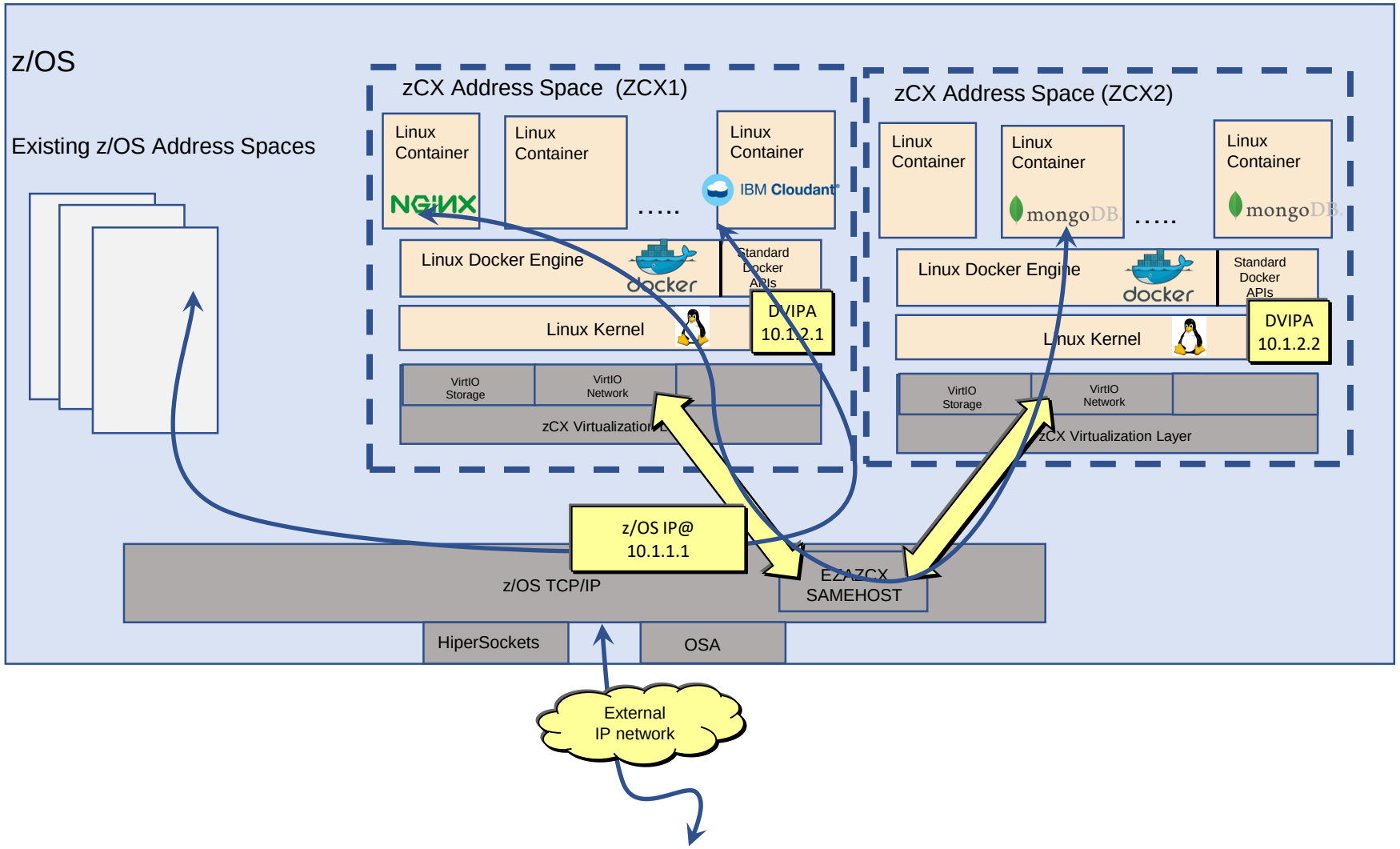


IBM zCX – z/OS Network Integration

- z/OS Linux Virtualization Layer:
  - Allows virtual access to z/OS Storage and Network
  - Using virtio Linux interfaces
    - Stable, well defined interfaces used to virtualize Linux
  - Allows us to support unmodified, open source Linux for z kernels
- Linux network access via high speed virtual *SAMEHOST* link to z/OS TCP/IP protocol stack
  - Each Linux Docker Server represented by a z/OS owned, managed and advertised Dynamic VIPA (DVIPA)
    - Allows restart of a CX instance in another system in the Sysplex
    - A new "application instance" DVIPA type "zCX" is introduced (created with the VIPARange statement)
  - Provide high performance network access across z/OS applications and Linux Docker containers – leveraging cross memory
  - External network access via z/OS TCP/IP
    - z/OS IP filters to restrict external access



Getting Started with zCX: Networking
IBM zCX – High Speed Virtual IP Network - SAMEHOST



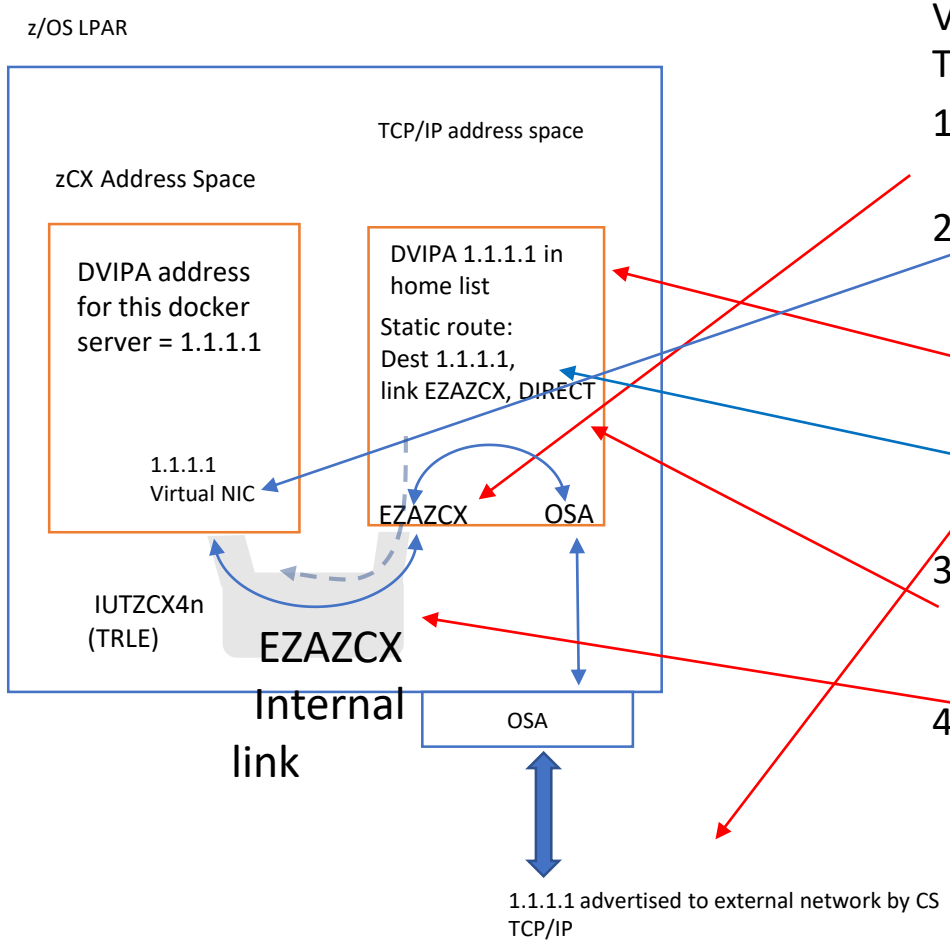
Usage and Invocation - zCX Network Configuration Steps

1. zCX Network information that will be needed for each zCX instance (inputs to z/OSMF zCX provisioning workflow):
   - zCX Server IP address – a zCX DVIPA, IPv4 (IPv6 address can also be specified if IPv6 is enabled on the z/OS system)
   - DNS Server IP Addresses (up to 2 for resiliency)
   - DNS Search Domain – example: pok.ibm.com, ibm.com
   - MTU (optional, default = 1492, suitable for most environments)
   - TCP/IP Stack name (only needed if multiple TCP/IP stacks are configured/active on the z/OS system)

2. z/OS TCP/IP profile:
   - zCX DVIPA(s) - Using VIPARANGE statements, configure zCX DVIPAs (IPv4 and optionally IPv6).
     The DVIPA must match zCX server configuration! (Must match the z/OSMF Workflow configuration, step 1 above)
   - ***Note:*** The same VIPARANGE statements should be replicated across all systems in the Sysplex that you wish to start this zCX instance on.
   - EZAZCX interface is created when the EZASAMEMVS (samehost) interface is created.
     Note. When using DynamicXCF or have enabled IUTSAMEH for Enterprise Extender both EZASAMEMVS (samehost) and EZAZCX interfaces are dynamically created and started. If you're not using Dynamic XCF or Enterprise Extender, then you must manually define (dev/link/home) IUTSAMEH (which will also create EZAZCX)

3. OMPROUTE profile:
   - Updates for zCX Dynamic VIPAs being used (Same as other DVIPAs – Use wildcarding where possible to simplify configuration)
   - And remember to propagate these to all other systems in the Sysplex that this zCX instance may be started on

zCX DVIPA and Networking Overview



Users define zCX DVIPA to TCP/IP using VIPARange ZCX (NCA or profile).
The remaining steps are automatic:

1. EZAZCX interface is automatically created by TCP/IP
2. When zCX is started:
   1. Connects to TCP/IP over EZAZCX using its DVIPA
   2. DVIPA (ZCX) is activated / added to the home list
   3. a (dynamically created) static IP route is created to the ZCX DVIPA (Docker server)
3. External network routes to the CS TCP/IP stack for ZCX address 1.1.1.1 are advertised
4. CS TCP/IP forwards packets for 1.1.1.1 …. over the zCX IP route and the EZAZCX link.

Note. Using existing z/OS TCP/IP support, IP filters can be applied during IP forwarding and IPSec tunnels can be applied to the external IP routes.

VIPARange ZCX (syntax and definition)

```
•                .-DEFINE-.    .-MOVEable NONDISRUPTive--.
• >>-VIPARange--+--------+--+-+------------------------+--address_mask--ipv4_addr------+--+----------
-+---->
•                '-DELEte-'   | '-MOVEable DISRUPTive-----'                'SAF resname'    |
•                             | .-MOVEable NONDISRUPTive--.                                 |
•                             '--+------------------------+--ipv6_intfname--ipv6_addr/prefix_len--'
•
• >------+--------+---><
•        '--ZCX---'
```

Notes:

- zCX DVIPAs are defined with VIPARANGE with a new keyword “ZCX”.
- The MOVEABLE keyword is ignored on a ZCX VIPARANGE (i.e. a zCX DVIPA can’t be activated if already active).
- An expected use case is that a zCX VIPARANGE may exist on multiple hosts so it should remain in sysplex VIPARange configuration.

- Support will also be available in the z/OSMF Network Configuration Assistant for defining zCX DVIPAs under the Configure Sysplex Networking actions

zCX DVIPAs – Security Considerations

VIPARANGE DVIPA creation can be controlled through two SERVAUTH profiles:

- EZB.MODDVIPA.*sysname*.*tcpname*
  - Limits who can create a VIPARANGE DVIPA in general
- EZB.MODDVIPA.*sysname*.*tcpname*.*resname*
  - Limit who can create a specific VIPARANGE DVIPA:
    - VIPARANGE DEFINE 255.255.255.255 10.10.10.1 *SAF APPL1*
    - Profile: EZB.MODDVIPA.*sysname*.*tcpname.APPL1*

If either of these 2 profiles are enabled then the userid associated with the zCX Started task will require READ access to these profiles

Verifying that zCX DVIPA has been defined (VIPARange)
(Netstat VIPADCFG)

```
D TCPIP,TCPCS,N,VIPADCFG
EZD0101I NETSTAT CS V2R4 TCPCS 463
DYNAMIC VIPA INFORMATION:
  VIPA RANGE:
    IPADDR/PREFIXLEN: 9.42.101.2/32
      MOVEABLE: NONDISR                    FLG: C
1 OF 1 RECORDS DISPLAYED
END OF THE REPORT
```

DVIPA is configured with a new DVIPA FLG ‘C’ indicating that this DVIPA is a zCX Dynamic VIPA (created with VIPARANGE ZCX)

Verifying DVIPA has been activated by the zCX instance
(Netstat VIPADYN)

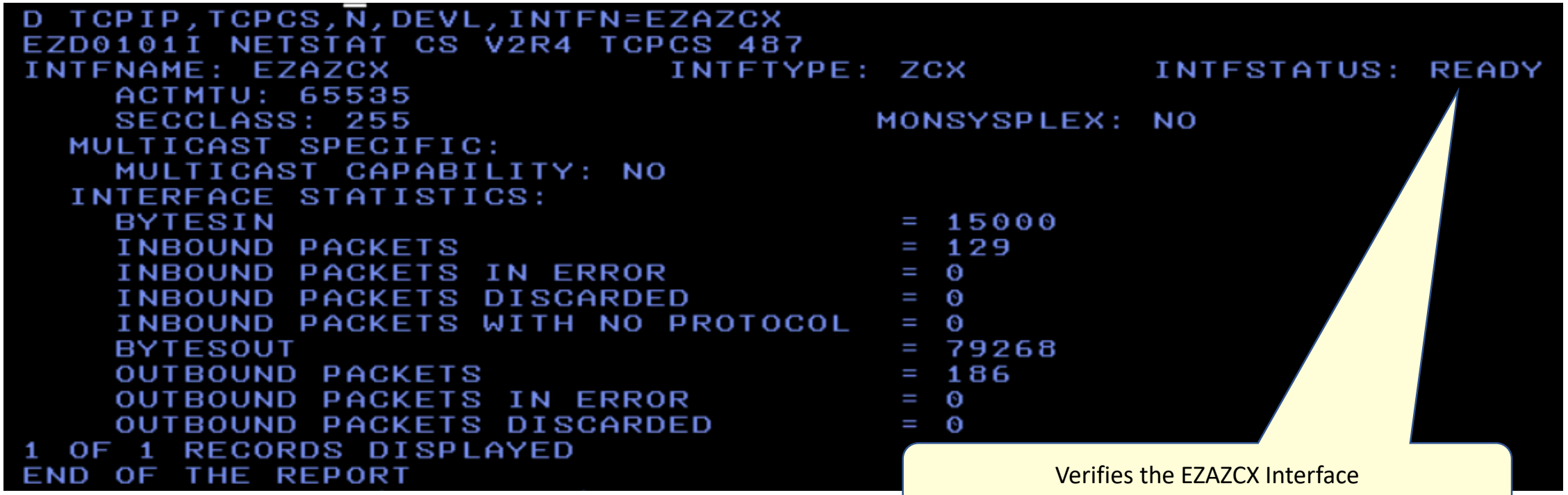
```
D TCPIP,TCPCS,N,VIPADYN
EZD0101I NETSTAT CS V2R4 TCPCS 508
DYNAMIC VIPA:
  IPADDR/PREFIXLEN: 9.42.101.2/32
    STATUS:  ACTIVE      ORIGIN: VIPARANGE IOCTL  ZCX:       YES
    ACTTIME: 02/04/2019 15:07:05                  JOBNAME: GLZ00001
1 OF 1 RECORDS DISPLAYED
END OF THE REPORT
```

zCX server instance started and activated DVIPA:

1. DVIPA activated is zCX type and matches your TCP/IP configuration
2. zCX instance (GLZ00001 = jobname)
3. TCP/IP stackname (TCPCS)
4. Time activated

Verifying that TCP/IP has activated the EZAZCX Interface

(Netstat DEVL,INTFN=EZAZCX)



Verifies the EZAZCX Interface
is created and ready

Verify the zCX Network Configuration

F *jobname*,DISPLAY,NET

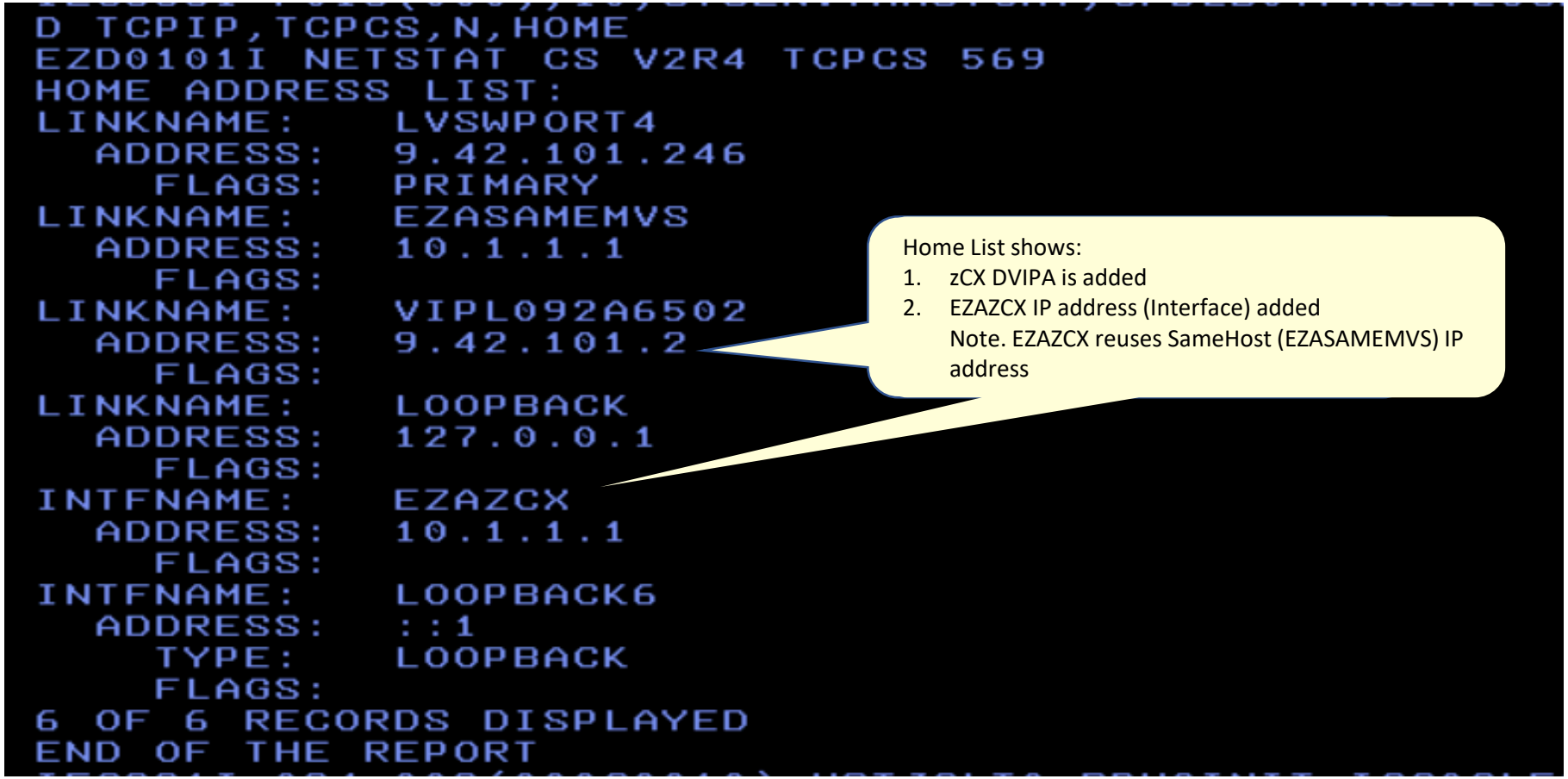
```
F GLZ00001,DISPLAY,NET
GLZC005I Network information for zCX instance GLZ00001 514
DevNo   Stack       MTU     IP Address
   0    TCPCS     65535    9.42.101.2
Total number of networks: 1
```

zCX (workflows) configuration:

1. DVIPA matches your TCP/IP configuration
2. Configured MTU (64k)
3. TCP/IP stackname (TCPCS)

Verify the EZAZCX IP address and zCX DVIPA have been added to the Home List

(Netstat HOME)



Getting Ready to Deploy zCX
CPU and Memory - Planning

- Number of virtual CPUs for each zCX instance
  - Specified at provisioning time
  - Each virtual CPU is a z/OS dispatchable unit (i.e. task)
  - The Dispatcher will dispatch these tasks on the available zIIPs (or general purpose CPUs) on the system based on WLM policies, priorities
  - Suggestion: Do not specify more virtual CPUs than the number of zIIPs (or General Purpose Processors) available to the z/OS system
- Memory for each zCX instance
  - Specified at provisioning time
  - This is Fixed private memory above the bar
  - Amount of memory needed will depend on the type of containers/software you are deploying
  - Minimum: 1GB, recommended: Multiple GBs – especially if you are not memory constrained
    - If the containers deployed in zCX drive virtual memory usage above the amount of Fixed memory specified, Linux will begin to page to the swap datasets, significantly impacting performance and driving requirements for larger swap datasets.

Getting Ready to Deploy zCX
WLM Policies

- Ensure that your WLM policies are updated to specify a Service Class for the zCX STC instance(s)
  - Classified under Subsystem=STC, can use Jobname or Userid qualifiers to identify zCX instances
  - Unique Service Classes (and optionally Report Classes) should be created
    - A single period should be created with an execution velocity goal– may need to experiment on exact value – will depend on the priority of this workload vs other workloads on the system, zIIP/CPU utilization, etc.
    - A WLM Importance Level that describes the overall priority of the zCX workload vs other workloads in the system
- zIIPs and spillover to standard CPs
  - SRM/WLM provides options that allow you to control how work is assigned between zIIPs and standard CPs. zIIP-eligible work may also execute on standard CPs in order to achieve workload goals.
  - System Level Option: Parmlib member IEAOPT*xx* contains the **IIPHONORPRIORITY** statement which controls the workflow to zIIPs.
    - Yes (Default) allows spillover to standard processors when needed. No indicates no spillover allowed (other than to resolve resource contention)
  - Service class level: You can specify YES/NO for Honor Priority when defining a service class
    - Overrides the **IIPHONORPRIORITY** setting in IEAOPTxx

Getting Ready to Deploy zCX
WLM Policies (cont)

- zIIPs and spillover to standard CPs (cont)
  - Your WLM administrators and Capacity Planners will need to be consulted to determine what makes sense in your environment as they will have insights into available zIIP and standard CP capacity, etc.
- Your WLM and capacity planners may also need to consider whether a resource cap is needed for the amount of zIIPs and standard CP capacity one or more zCX STC instances can consume
  - This can be done using WLM Resource Groups or the new WLM Tenant Resource Groups (provide more granularity)

Docker Swarm Intro

- Docker Swarm is a clustering and scheduling tool for Docker containers. With Swarm, IT administrators and developers can establish and manage a cluster of Docker nodes as a single virtual system
- Docker Swarm provides:
  - Single point of control across a set of Docker nodes acting as a cluster
  - Application replication
  - Load balancing
  - Relatively simple extension to the Docker management approach to provide clustering
- Built into Docker

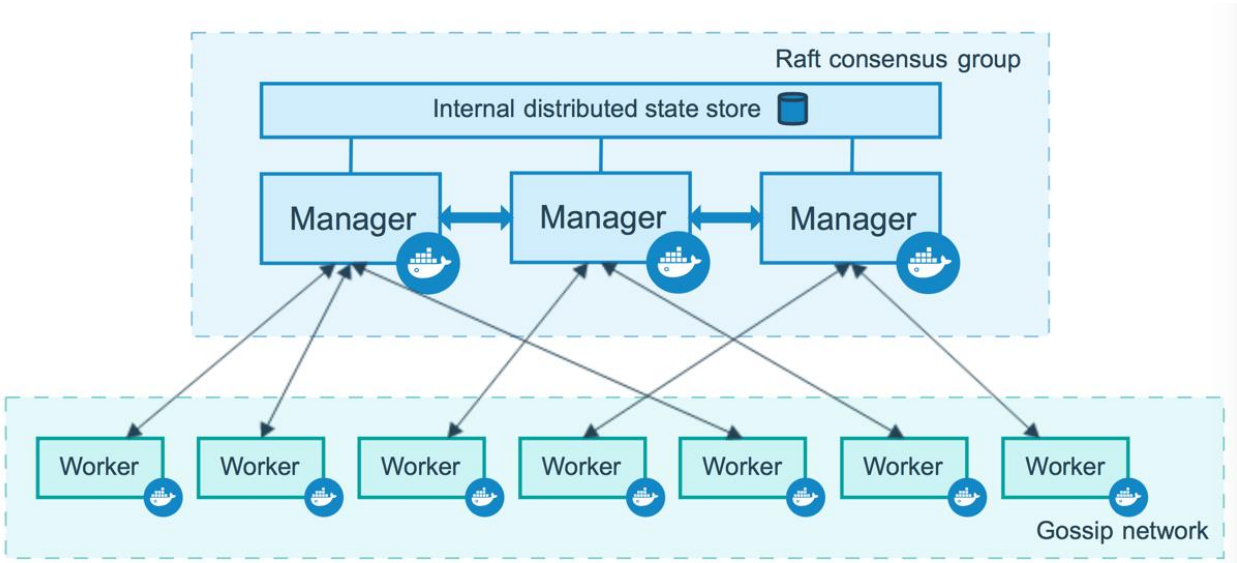
Docker Swarm Terminology

- A **swarm is** a cluster of Docker engines, or **nodes**, where you deploy **services**
- A **node is** an instance of the Docker engine participating in the swarm. You can also think of this as a Docker node. You can run one or more nodes on a single physical or virtual computer . In our case it is a Open Containers Virtual Docker Server.
- A **task** carries a Docker container and the commands to run inside the container. It is the atomic scheduling unit of swarm.
- A **service** is the definition of the tasks to execute on the worker nodes (When you create a service, you specify which container image to use and which commands to execute inside running containers)
- A **stack** is a collection of services that make up an application in a specific environment.
- A **manager** node is where you submit a service definition in order to deploy an application to a swarm. The manager node dispatches units of work called tasks to worker nodes.
- **Worker** nodes receive and execute tasks dispatched from manager nodes. By default manager nodes also run services as worker nodes

Docker Swarm Features

- **Scaling:** For each service, you can declare the number of tasks you want to run. When you scale up or down, the swarm manager automatically adapts by adding or removing tasks to maintain the desired state.
- **Desired state reconciliation:** The swarm manager node constantly monitors the cluster state and reconciles any differences between the actual state and your expressed desired state. For example, if you set up a service to run 10 replicas of a container, and a worker machine hosting two of those replicas crashes, the manager creates two new replicas to replace the replicas that crashed. The swarm manager assigns the new replicas to workers that are running and available.
- **Multi-host networking:** You can specify an overlay network for your services. The swarm manager automatically assigns addresses to the containers on the overlay network when it initializes or updates the application.
- **Service discovery:** Swarm manager nodes assign each service in the swarm a unique DNS name and load balances running containers. You can query every container running in the swarm through a DNS server embedded in the swarm.
- **Load balancing:** You can expose the ports for services to an external load balancer. Internally, the swarm lets you specify how to distribute service containers between nodes.
- **Secure by default:** Each node in the swarm enforces TLS mutual authentication and encryption to secure communications between itself and all other nodes. You have the option to use self-signed root certificates or certificates from a custom root CA.
- **Rolling updates:** At rollout time you can apply service updates to nodes incrementally. The swarm manager lets you control the delay between service deployment to different sets of nodes. If anything goes wrong, you can roll-back a task to a previous version of the service.

A swarm: architectural view

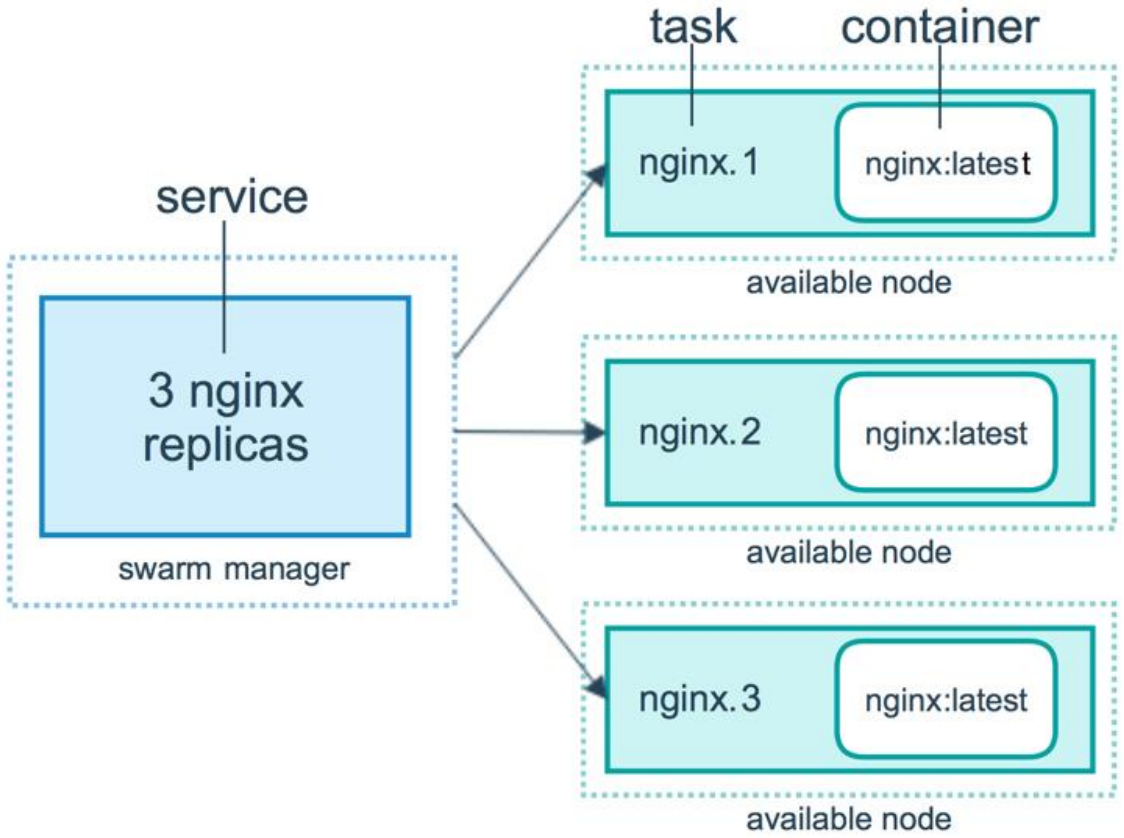


Manager nodes handle cluster management tasks:

- maintaining cluster state
- scheduling services
- serving swarm mode HTTP API endpoints

- Worker nodes receive and execute tasks dispatched from manager nodes.
- By default manager nodes also run services as worker nodes, but you can configure them to run manager tasks exclusively and be manager-only nodes.
- An agent runs on each worker node and reports on the tasks assigned to it.
- The worker node notifies the manager node of the current state of its assigned tasks so that the manager can maintain the desired state of each worker.

Services in depth



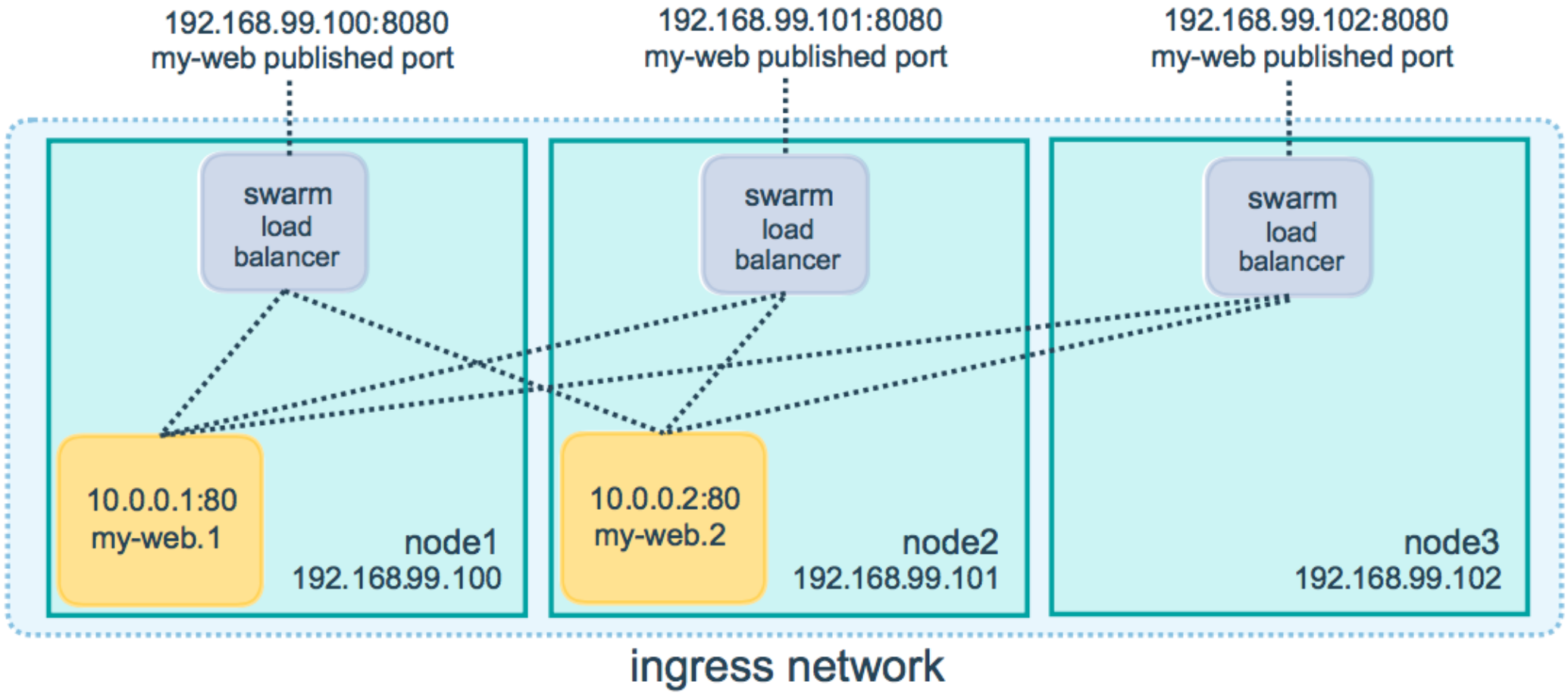
- A service is the definition of the tasks to execute on the worker nodes.
- Frequently a service will be the image for a microservice within the context of some larger application. Examples of services might include an HTTP server, a database, or any other type of executable program that you wish to run in a distributed environment.
- When you create a service, you specify which container image to use and which commands to execute inside running containers.
- Once can also define options for the service including:
  - The port where the swarm will make the service available outside the swarm
  - CPU and memory limits and reservations
  - The number of replicas of the image to run in the swarm

Scaling: For each service, you can declare the number of tasks you want to run. When you scale up or down, the swarm manager automatically adapts by adding or removing tasks to maintain the desired state.

Stack

- A stack is a collection of services that make up an application in a specific environment.
- Stacks are a convenient way to automatically deploy multiple services that are linked to each other, without needing to define each one separately.
- Stack files define environment variables, deployment tags, the number of containers, and related environment-specific configuration.

Swarm Load Balancing



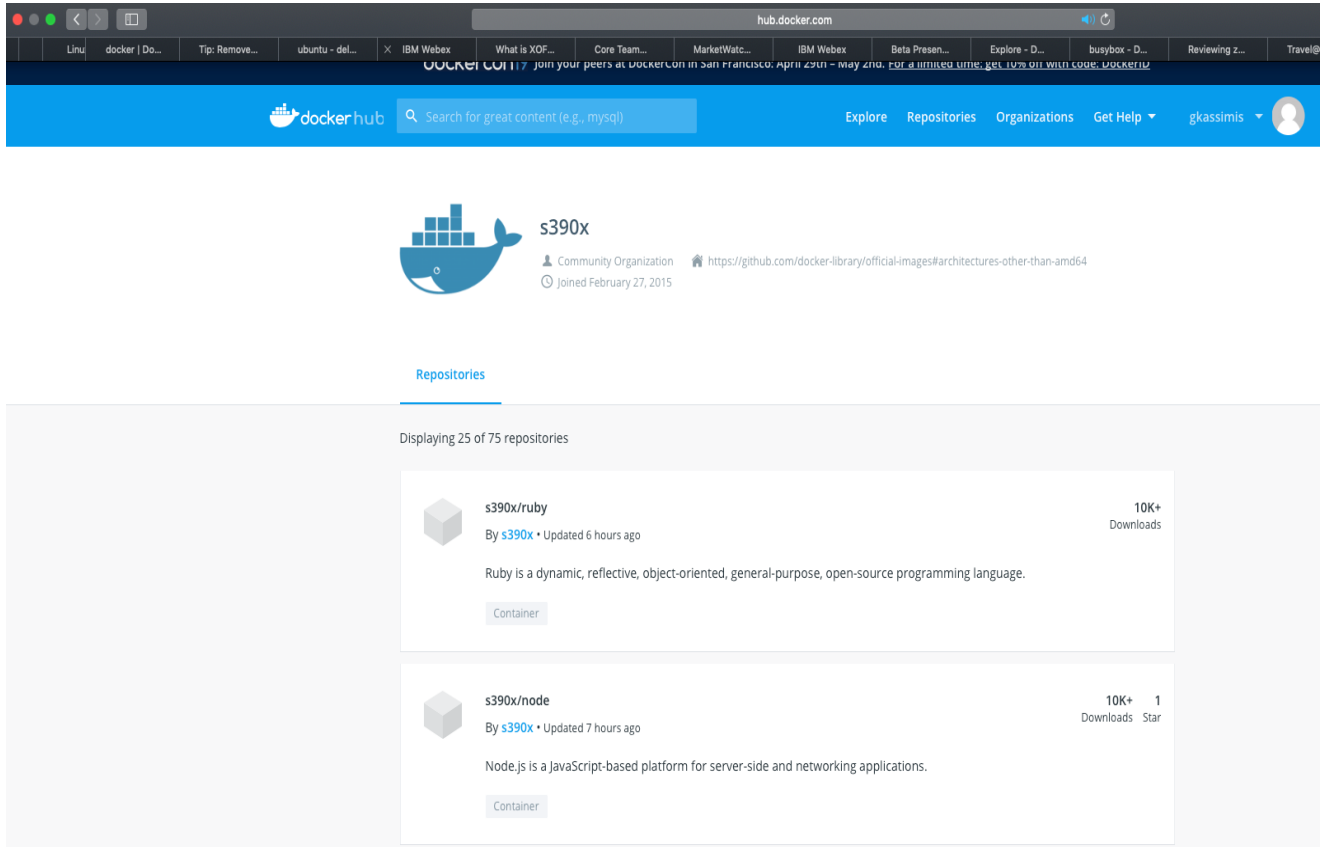
When you access port 8080 on any node, the swarm load balancer routes your request to an active container.

Other Notes about a Swarm Cluster

- The scope of a Swarm has no technical relationship to a z/OS sysplex
  - The nodes of a Swarm only require a network connection
  - From a z/OS management scope it may make sense to keep a Swarm cluster contained with in a z/OS sysplex
- Swarm has scheduling parameters to control on which nodes a given service is deployed
  - Each node can be labeled and a service can be defined to only be deployed to nodes that have a label within a given set.
- A Swarm can be heterogenous from an architecture point of view (e.g. x86 and z)
  - If multi-architectures images are available from a registry, each node will pull the proper image for its architecture
  - Node labels can be used to direct services to only a specific architecture
  - Are there use cases for a multi architecture Swarm?
  - IBM will only test and support Swarm clusters within a z/OS Sysplex environment

Docker Registries

- The Docker Registry is a stateless server side application that stores and distributes Docker images.
- Users must connect to this registry in order to load a docker image onto a zCX Appliance.
  - Both secure and insecure configurations for connecting to the registry are supported.
  - External Docker hub (look for s390x architecture images – this reflects images for IBM Z)
    - https://hub.docker.com/u/s390x/
    - This is built into Docker and cannot be disabled
  - But you can direct Docker commands to an internal registry (secure or unsecure)
    - Must be specified during zCX provisioning workflow (or reconfiguration workflow)
    - Helps alleviate connectivity issues with accessing external DockerHub from your zCX systems



Docker Registries (cont)

- Insecure Docker Registry
  - A Docker daemon running in the IBM zCX Appliance can connect to an insecure private Docker registry
  - Communication to insecure docker registry is not protected. This is very insecure and is generated not recommended (unless in test environments or environments that provide isolated networks/security)
  - The user must provide the domain name or IP address and the Port of the secure docker registry.
- Secure Docker Registry
  - Secure Docker Registry allows you to connect to your own private docker registry with TLS authentication
  - The user must provide the domain name or IP address and the Port of the secure docker registry.
  - You will also need the necessary TLS certificates and key as well during the provisioning workflow process
  - Contact your Docker admin for more information on internal Docker registries you may have deployed in your enterprise
- You can also host an internal registry inside zCX
  - Instructions on how to deploy a registry inside zCX will be provided in the IBM z/OS Container Extensions Guide

Reconfiguration and Upgrade of a zCX Appliance Instance

- Additional Workflows:
  - Reconfigure a zCX Appliance
    - Modify CPU, memory, Network
    - Add additional disks for Data, Swap and Dlog pools
    - Modify Docker Configuration Parameters
    - A restart of the zCX appliance will be required for all configuration changes
  - Upgrade a zCX Appliance to the latest code level
    - Updates/Maintenance to zCX Appliance will be a complete replacement of the appliance
    - The workflow will allow you to refresh the appliance code level while maintaining the existing configuration and deployed containers
    - A restart of the zCX appliance will be required to pick up the new code level
  - More details will be provided in the future in the IBM z/OS Container Extensions Guide

Additional zCX Operator Commands

- modify zcxgus1,display,config (display configuration information)

```
RESPONSE=NP8
 GLZC003I Configuration information for zCX instance ZCXGUS1
 File Path:      /oc4z/shared/zcx_instances/ZCXGUS1/start.json
 Dump Path: /oc4z/shared/zcx_instances/ZCXGUS1/FFDC/zcx-guest.dmp
 Memory size:              4GB
 Number of CPUs:           2
 Number of Disks:          5
 Number of Networks:    1
 CTRACE Parmlib Member: CTIGLZ00
```

Additional zCX Operator Commands (cont)

- modify zcxgus1,display,diskver (Display Disk Version Information)

```
RESPONSE=NP8
 GLZC008I Disk Version information for zCX instance ZCXGUS1
 DevNo    Data Set Name                                    Version
   1      OMVSSPA.ZCX.ZCXGUS1.ROOT                         20190312T212339Z
                  3.3.3       1.3.0       HZDC7C0              v1.3.0
   2      OMVSSPA.ZCX.ZCXGUS1.CONF                         20190318T195354Z
   3      OMVSSPA.ZCX.ZCXGUS1.SWAP1                        20190318T195332Z
   4      OMVSSPA.ZCX.ZCXGUS1.DATA1                        20190318T195335Z
   5      OMVSSPA.ZCX.ZCXGUS1.DLOG1                        20190318T195339Z
 Total number of disks: 5
```

Additional zCX Operator Commands (cont)

- modify zcxgus1,display,disk (Display disks)

```
RESPONSE=NP8
 GLZC004I Disk information for zCX instance ZCXGUS1
 DevNo        Size          Encrypted?        Data set Name
     1         4GB            No              OMVSSPA.ZCX.ZCXGUS1.ROOT
     2         3MB            No              OMVSSPA.ZCX.ZCXGUS1.CONF
     3      1001MB            No              OMVSSPA.ZCX.ZCXGUS1.SWAP1
     4         5GB            No              OMVSSPA.ZCX.ZCXGUS1.DATA1
     5      1001MB            No              OMVSSPA.ZCX.ZCXGUS1.DLOG1
 Total number of disks: 5
```

Session Summary

- zCX brings an exciting new capability to z/OS: the ability to deploy s390x Docker Linux containers directly under z/OS
- This new runtime is planned to be a part of the Operating System
- It should improve the platform
  - For application development
  - For application deployment
  - For overall management of a customers end to end enterprise applications
- And can be managed from a z/OS Operational perspective

Appendix

- IBM z/OS Container Extensions Guide
  - A PDF version of this document will be made available
  - It will contain more details on many of the topics discussed today
- For questions on Use Cases and How to
  - Joe Bostian – jbostian@us.ibm.com
  - Gus Kassimis – kassimis@us.ibm.com
- z/OS Container Extensions Content Solution page:
  - https://izswebpage.mybluemix.net/zcx/containerextensions.html
- *Google Docker:* Lots of excellent resources, documentation and self-paced learning courses are available