z/OS 3.1 IBM Education Assistant

Solution Name: zCX NFS Support

Solution Element(s): z/OS Container Extensions

July 2023

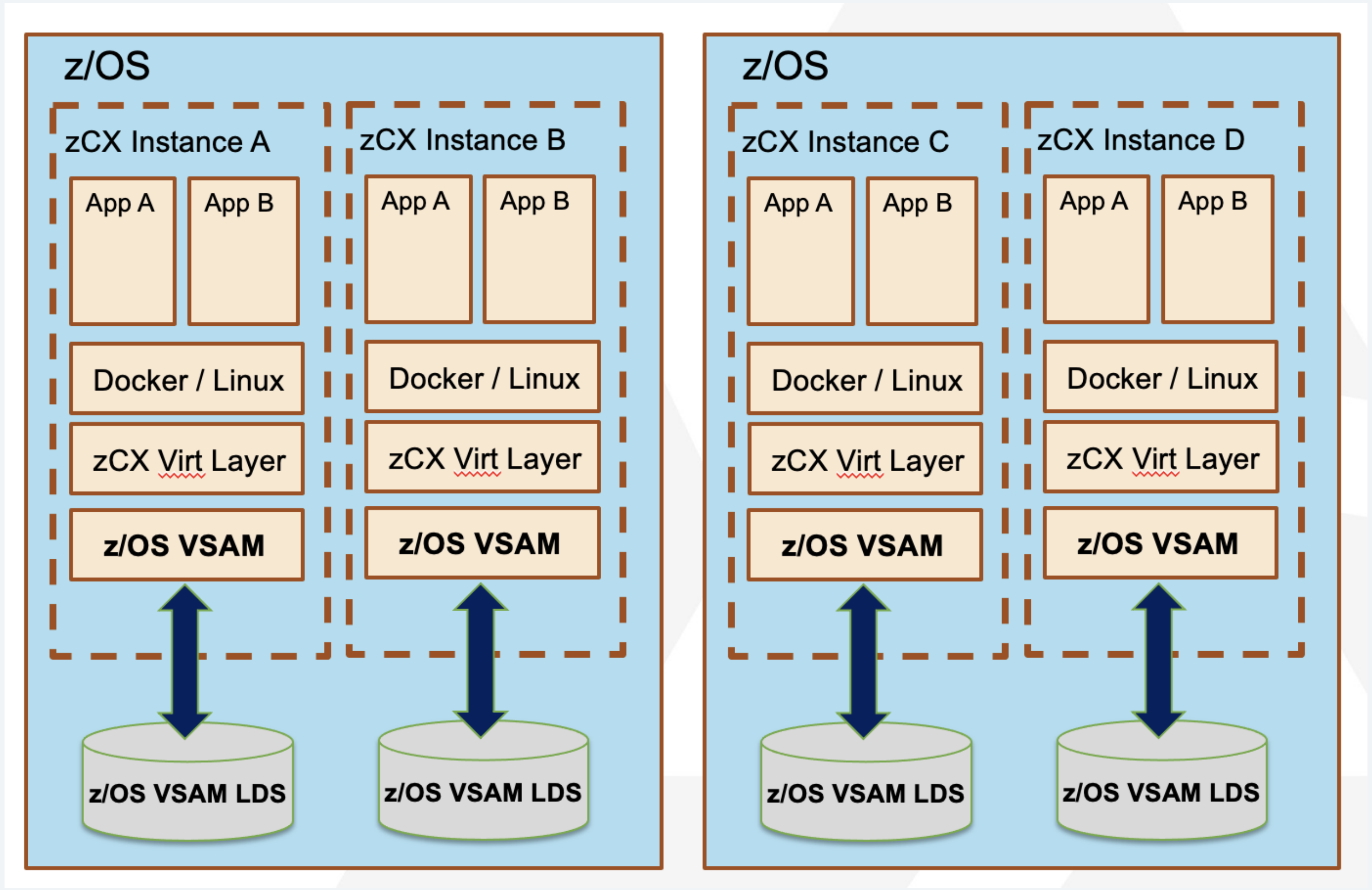
Agenda

Trademarks

Overview

- Containerized applications running inside z/OS Container Extensions (zCX) may require persistent storage option to store and share the stateful application data. NFS is one of the persistent storage option that is available for zCX.
- By leveraging **z/OS NFS server** to persist data, containerized applications deployed in zCX can take advantage of the z/OS security controls and operational benefits to securely store, access, backup and restore application data.
- z/OS NFS APAR OA62357 provides AT-TLS support for NFS client connections to z/OS NFS Server. There are no code changes for zCX.
- This solution, not only makes the authentication of multiple container userIDs to access NFS data simpler, but also provides the additional benefit of an encrypted data connection between the zCX and the NFS server host.

Local Non-Shared Storage



- Creates affinity between containers/App to zCX instance that owns the docker volumes

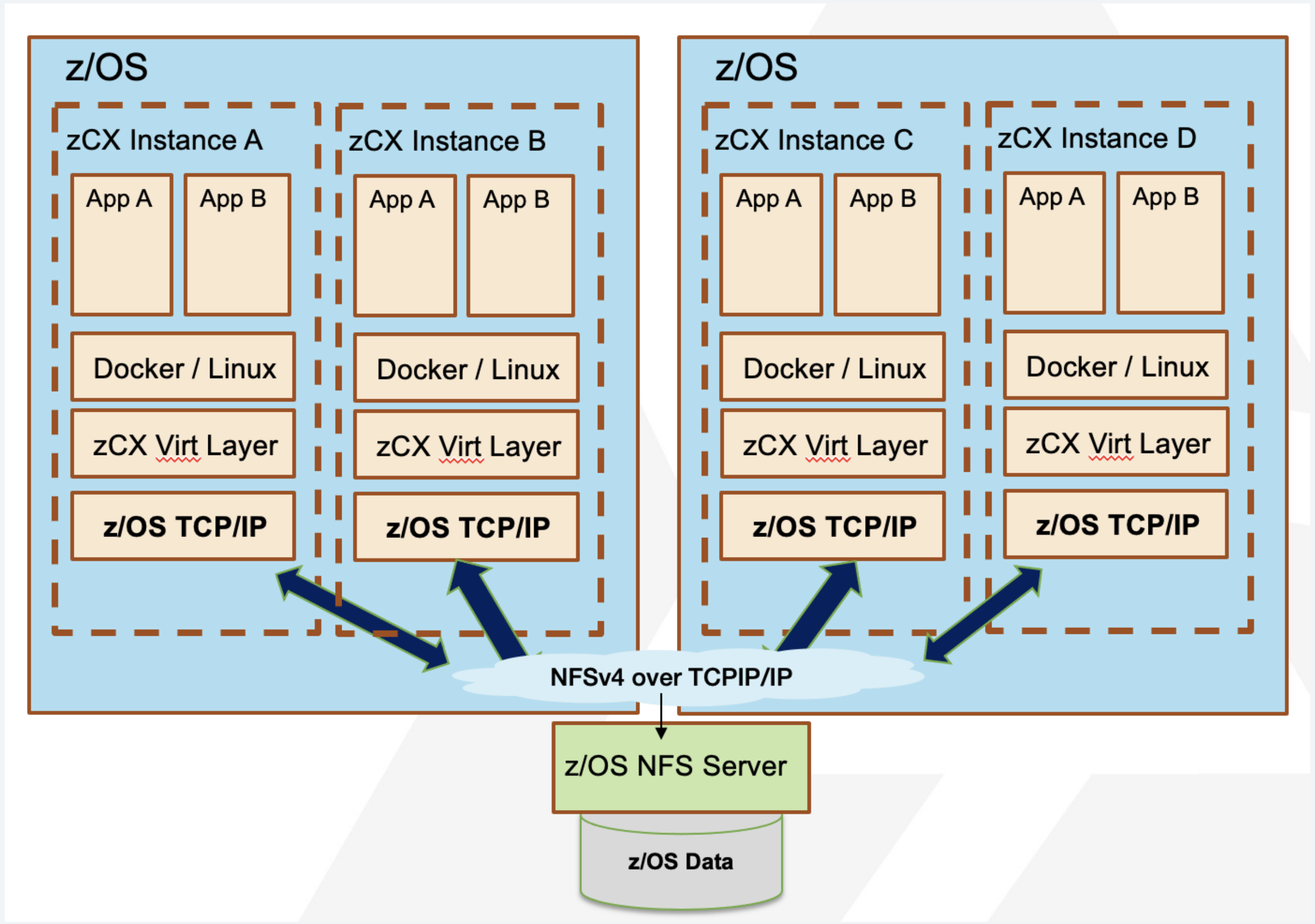
- Containers with affinity to a local volume can only be restarted in the same zCX that owns the volumes

Other Attributes:

- Very good performance through z/OS VSAM

- Can exploit z/OS VSAM strengths (pervasive encryption, HyperSwap, Replication, DR planning, etc.)

External Shared Storage



z/OS NFS Server

- Backed by native z/OS file system and QoS, can be integrated into z/OS storage replication and disaster recovery operations
- Ability to enable z/OS pervasive encryption (transparent to Containers)
- Containers can be moved around to any zCX have access to persistent data/volume
- Containers can be scaled up across multiple zCX (different LPAR, CEC) and still could be sharing the data

Containers can access z/OS datasets

Remote non-z/OS NFS Server

- Separate planning needed for encryption, resiliency, replication, DR, etc.

zCX for OpenShift Persistent Storage

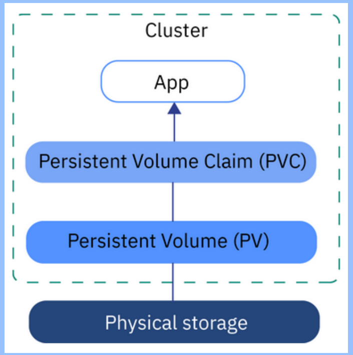
What is persistent Storage?

Containerized applications may require persistent storage, that is data storage that retains its contents beyond the life time of a single instance of execution of the application.

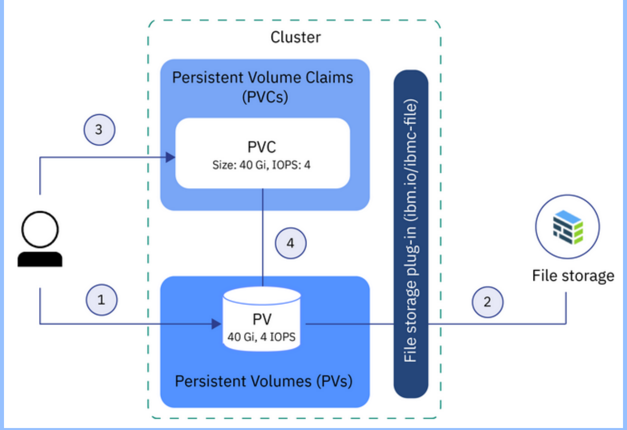
To enable this use case, Kubernetes (and OpenShift) provides a Persistent Volume API that enables access to persistent storage. This can be in the form of block, file, or object storage.

The cluster administrator defines Persistent Volume (PV) resources that map to various forms of physical (persistent) storage. An application defines Persistent Volume Claim (PVC) resources that define the storage requirements of the application. The Kubernetes infrastructure then maps requested PVCs to available PVs.

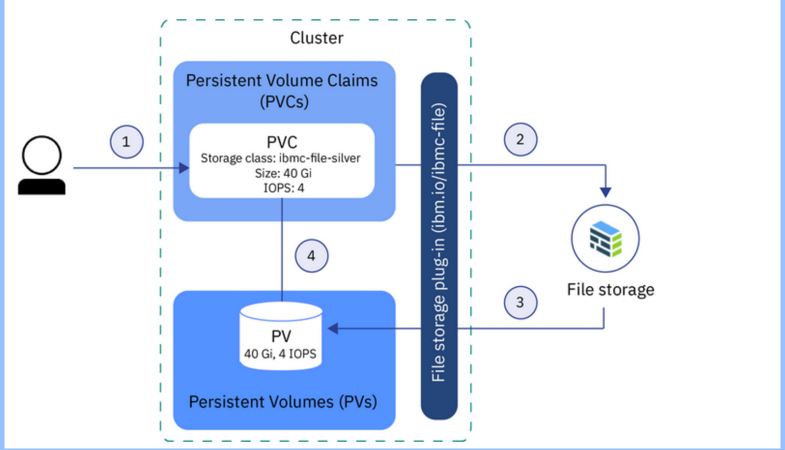
This mapping may use static provisioning (PVs defined ahead of time by the administrator) or dynamic provisioning (PVs created on demand out of pre-allocated pools - "storage classes"). The Kubernetes API allows for a variety of options to provide PVs.



Sample flow for static provisioning of file storage



Sample flow for dynamic provisioning of file storage with the pre-defined silver storage class


Cluster
Persistent Volume Claims (PVCs)
PVC
Storage class: ibmc-file-silver
Size: 40 Gi
IOPS: 4
1
2
3
4
PV
40 Gi, 4 IOPS
Persistent Volumes (PVs)
File storage plug-in (ibm.io/ibmc-file)
File storage

Persistent Storage Options for zCX for OpenShift

Local Volumes using LSO (VSAM LDS)

- OpenShift Container Platform can be provisioned with persistent storage by using local volumes
- Local persistent volumes allow you to access local storage devices, such as a disk or partition, by using the standard persistent volume claim interface
- Creates affinity between Pods (containers/App) to zCX OCP instance that owns the local volume
- Recovering the workload on a zCX instance failure implies restarting the zCX OCP instance (in same z/OS system or another system in the sysplex)
- Recovering the workload on a z/OS system failure may involve restarting the zCX OCP instance on another system in the sysplex
- Very good performance through z/OS VSAM
- Can exploit z/OS VSAM strengths (pervasive encryption, HyperSwap, Replication, DR planning, etc.)

NFS

z/OS NFS Server

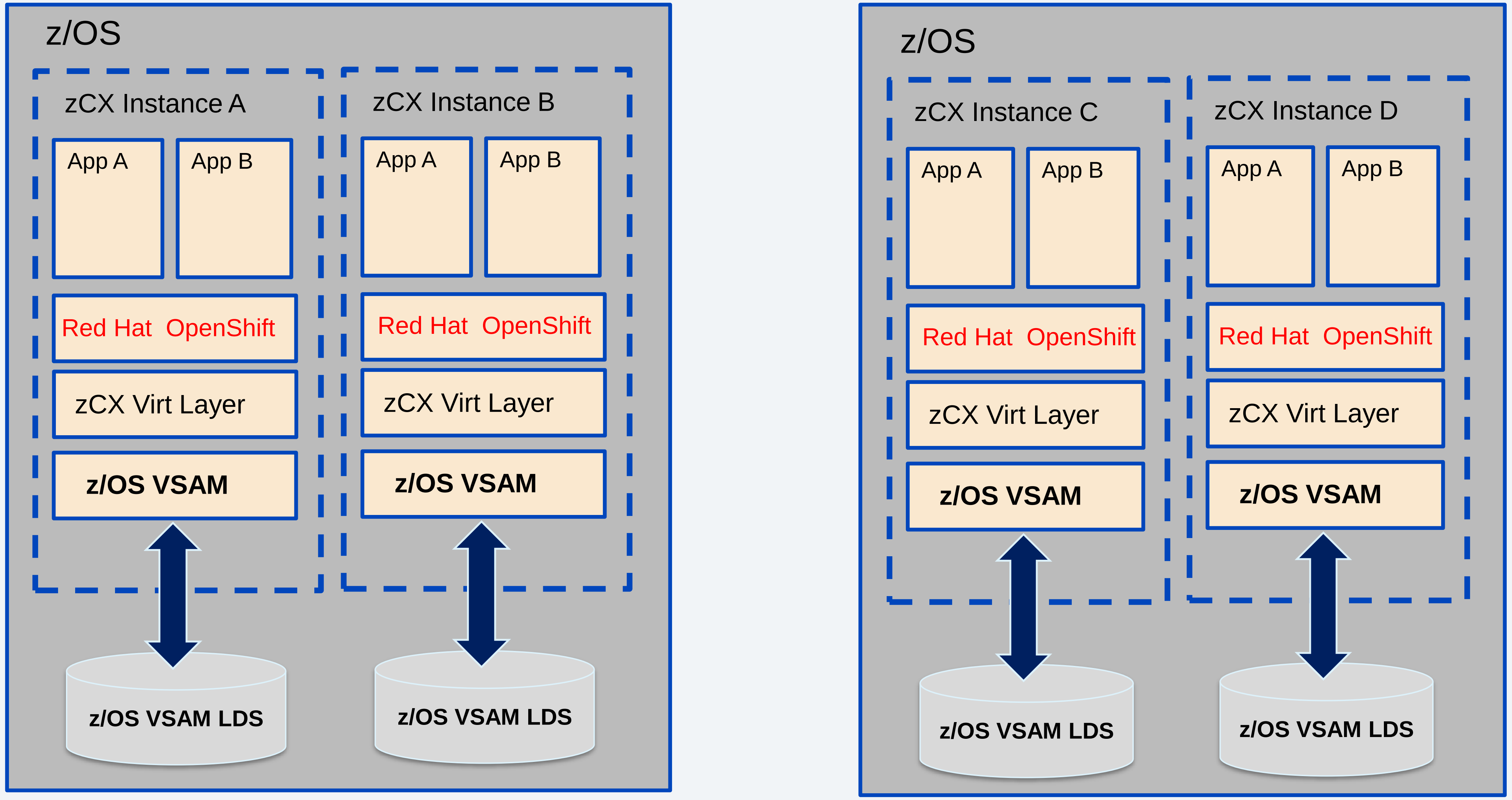
- Backed by native z/OS file system and QoS, can be integrated into z/OS storage replication and disaster recovery operations
- Ability to enable z/OS pervasive encryption (transparent to OpenShift)

Remote non-z/OS NFS Server

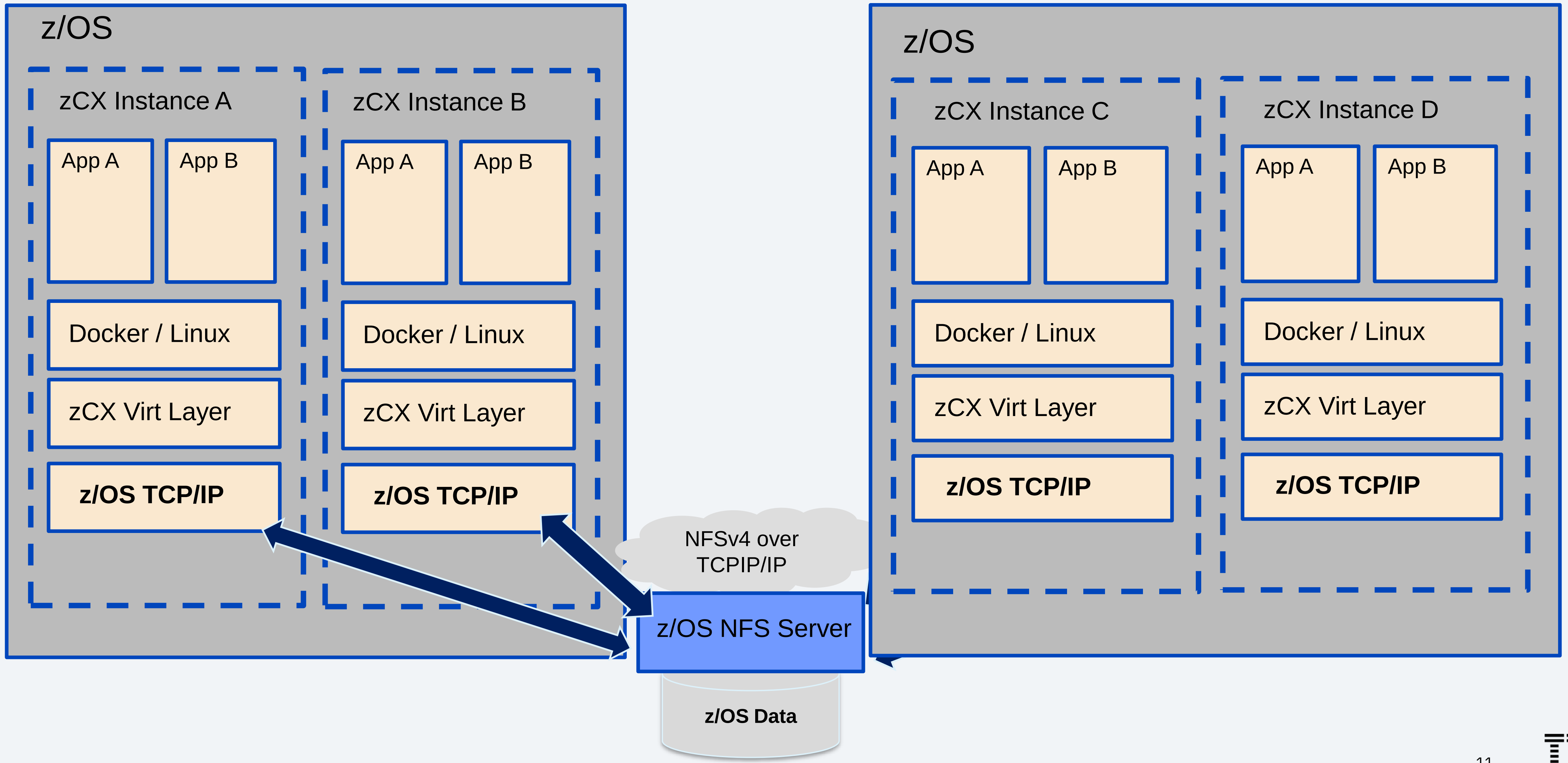
- Separate planning needed for encryption, resiliency, replication, DR, etc.

Pods can be scheduled anywhere in the cluster and can still have access to Persistent Volumes

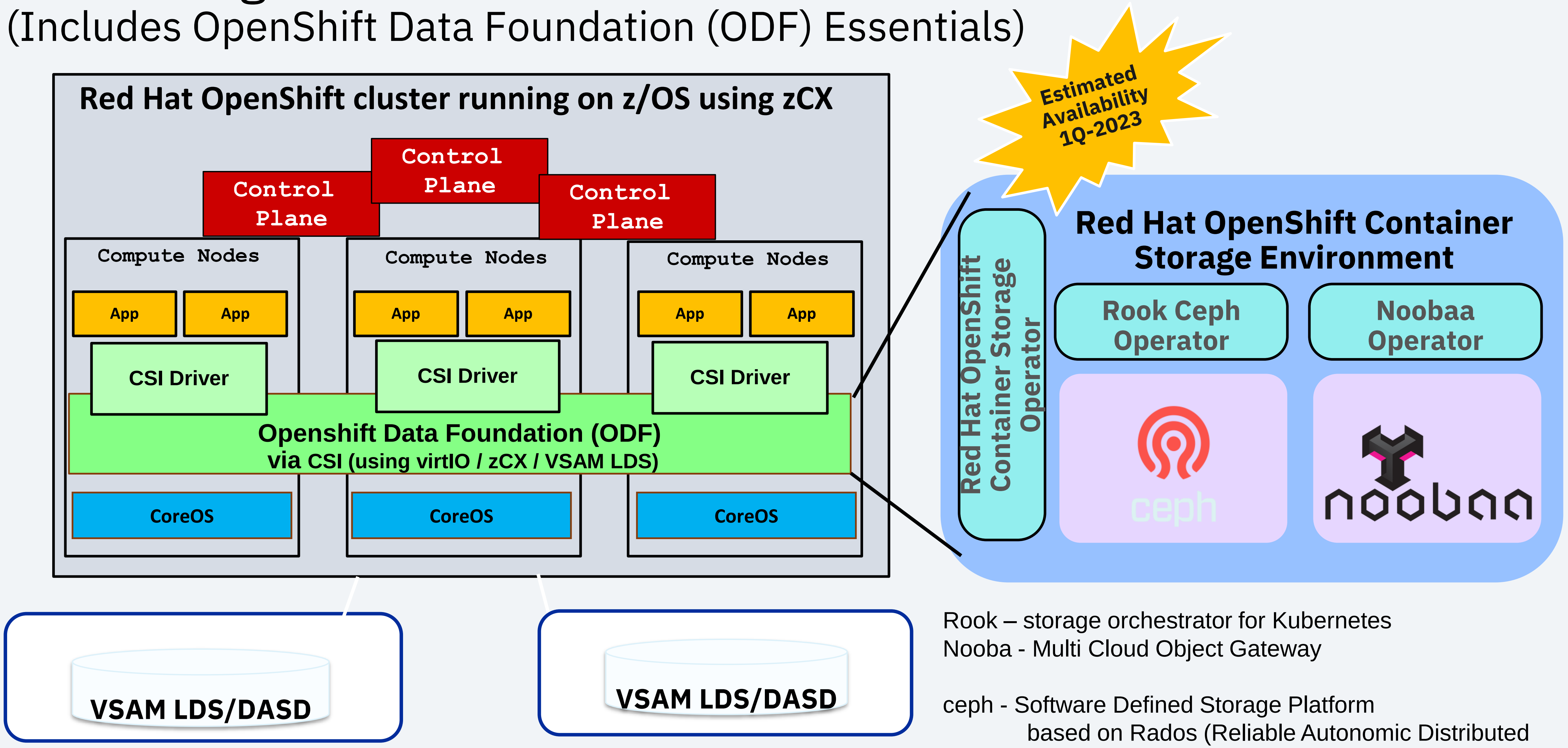
zCX Persistent Storage – Local Volume using LSO



zCX Persistence Storage - Shared NFS Volumes



IBM Storage Fusion -
(Includes OpenShift Data Foundation (ODF) Essentials)


Red Hat OpenShift cluster running on z/OS using zCX
Control Plane
Control Plane
Control Plane
Compute Nodes
Compute Nodes
Compute Nodes
App
App
App
App
App
App
CSI Driver
CSI Driver
CSI Driver
Openshift Data Foundation (ODF)
via CSI (using virtIO / zCX / VSAM LDS)
CoreOS
CoreOS
CoreOS
VSAM LDS/DASD
VSAM LDS/DASD
Estimated Availability 1Q-2023
Red Hat OpenShift Container Storage Environment
Red Hat OpenShift Container Storage Operator
Rook Ceph Operator
Noobaa Operator
ceph
noobaa

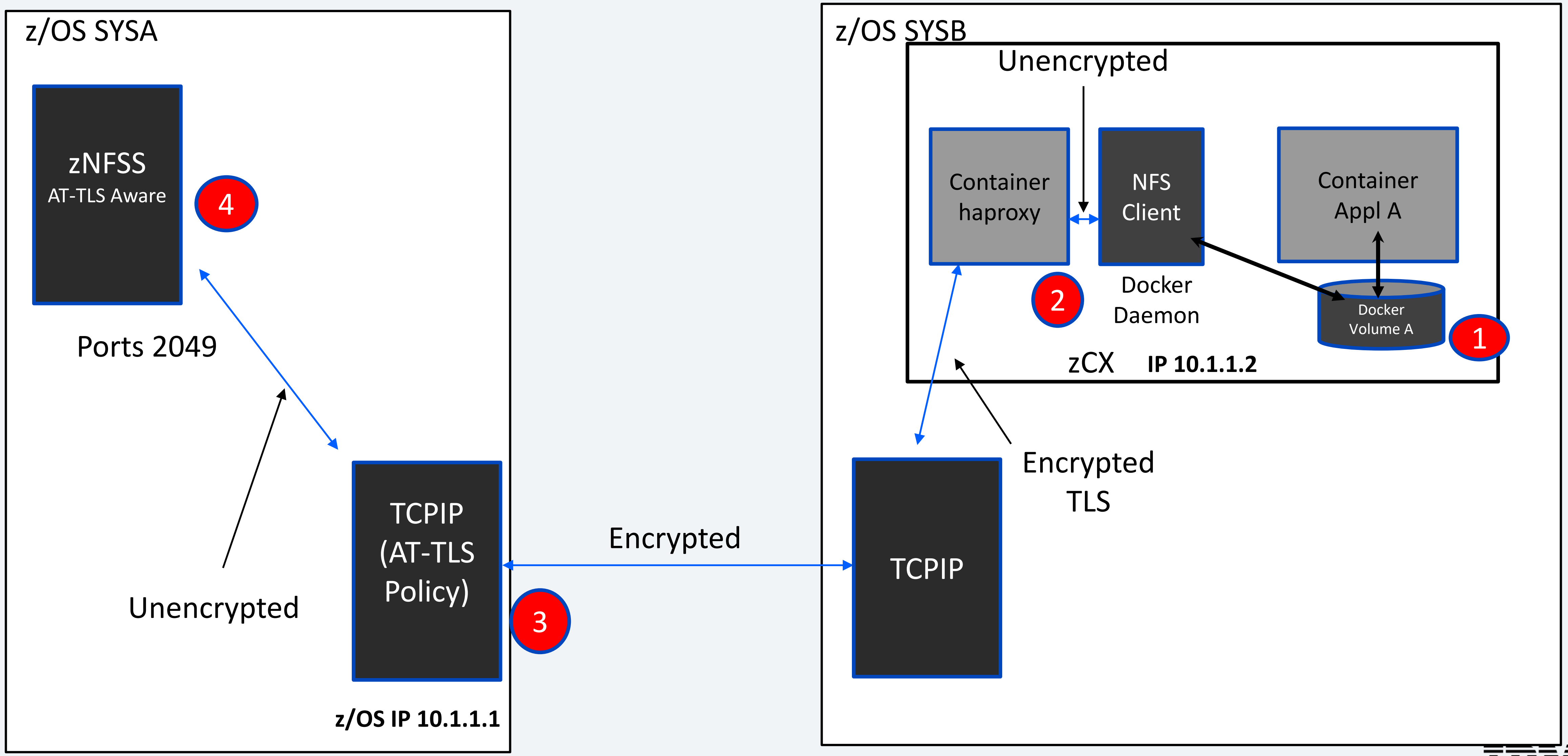
Shared Persistent Container Storage via ODF

Rook – storage orchestrator for Kubernetes
Nooba - Multi Cloud Object Gateway

ceph - Software Defined Storage Platform based on Rados (Reliable Autonomic Distributed Object Store)

Installation & Configuration

zCX NFS Docker Volume, HAPROXY and z/OS AT-TLS NFSv4 Server



High-Level Steps - Overview

1. Containers running on zCX which try to mount NFS datasets submit NFS mount requests that are routed via HAPROXY to NFS server. For mount requests we specify the zCX DVIPA.

2. zCX is setup with TLS server such as HAPROXY running as a container. It routes these mount request with zCX DVIPA to NFS server IP. HAProxy is built with cert-auth and client certificate which has the z/OS id associated with it.

3. AT-TLS policy setup on NFS server hosts intercepts the NFS request based on port range/ zCX DVIPA. It verifies the client certificate and that it has associated z/OS useid defined on RACF.

4. With ClientAuthType = SAFCheck, NFS server setup with security(safexp), it submits the request to a security product such as RACF to create ACEE using this z/OS userid in the client cert. NFS requests from any container userid from this zCX will use this ACEE.

z/OS NFS Server Setup 4

- Use z/OS NFS Server v4
- Apply NFS APAR OA62357 to make NFS server AT-TLS aware
- Ensure z/OS NFS server has a security attribute setting of ‘safexp’
- Export all directories to be mounted inside the zCX application container
  - Mount points
- Note: The TCPIP stack on the z/OS Server host system must have AT-TLS enabled via “TTLS” in the TCPCONFIG statement in TCPIP Profile

AT-TLS Policy configuration 3

- The TTLS Rule must specify the NFS server job name, the port range 2043:2049 for NFS requests, and the zCX DVIPA for all zCX instances planning to access NFS data
- TTLS Environment action must specify HandShakeRole of ServerWithClientAuth with ClientAuthType: SAFCheck
- TTLS KeyRing parameter must specify the keyring. This keyring is owned by NFS server id. It has the cert-auth certificate, and the NFS server id personal certificate signed by cert-auth certificate
- With ClientAuthType SAFCheck, Certificate is validated against keyring and must be associated with a user ID in the security product
- AT-TLS policy can be dynamically updated such as for adding or removing zCX DVIPA. For this stop NFS server, refresh PAGENT, start NFS server
- If your z/OS NFS server is configured to run on multiple LPARs for high availability, be sure to have the same policies applied to any LPAR on which the z/OS NFS server might be running

Generate Certificates to enable AT-TLS and Authentication

- Server and Client Certificates must be generated on the z/OS NFS server system
- A certificate authority certificate is needed
  - Create or obtain a CERTAUTH certificate with KEYUSAGE(CERTSIGN)
- Create a server certificate for z/OS NFS server ID with KEYUSAGE (HANDSHAKE DATAENCRYPT)
  - Sign it with CERTAUTH certificate and connect to a keyring owned by z/OS NFS server ID
- Create a client certificate for z/OS UserID that will be associated with the zCX instance, and will be used to authenticate and authorized to access z/OS NFS exported data sets
  - Note: This ID should only have access to the z/OS data sets intended to be accessed by the zCX instance
- Export client certificate and the CERTAUTH certificates in CERTB64 and PKCS12DER formats
- Transfer exported client certificate and CERTAUTH certificate to the zCX instance to use with proxy server

Set up TLS Proxy Server using HAPROXY Container on zCX 2

- Get the base image from IBM Cloud Registry (ICR) for s390x architecture:
  - icr.io/ibmz/haproxy:[version]
- Configure and deploy the HAPROXY container on zCX instance where you plan to access z/OS NFS server exported data sets
  - Copy certauth, and client certificates, and proxy configuration
  - Build the container including the certauth and client certificates
    - Proxies traffic to NFS server.
- ACLs should be setup with zCX's DVIPA, so that no requests from other IPs are routed to the NFS server via this HAPPROXY instance
- Deploy the container to listen on port range 2043-2049
- **Note:** All traffic from a single zCX appliance to the z/OS NFS server shares a single client certificate, and therefore the same SAF privileges to NFS-mounted data sets and directories

zCX NFS Docker Volume Mount – Application Container 1

- Create docker volume to use NFS as shared persistent volume:
  - docker volume create --driver local --opt type=nfs4 --opt o=addr=10.0.1.2,rw --opt device=:/hfs/home/user1/zcxtest1/zosdata,mvsmnt volhfs
  - Where 10.0.1.2 is the DVIPA IP address of the same zCX instance
- Start the application container in zCX by specifying a mount point for remote NFS mount:
  - docker run --restart=unless-stopped-d -it -v volhfs:/targetdir --name conthfs ubuntu bash

Interactions & Dependencies

Software Dependencies

– APAR OA62357

Hardware Dependencies

– None

Exploiters

– None

Upgrade & Coexistence Considerations

None

Reference

- Documentation:
  - Using NFS as persistent storage for zCX:
    - https://www.ibm.com/docs/en/zos/2.5.0?topic=pz-using-zos-nfs-server-as-persistent-storage-zcx
  - Setting up the z/OS NFS Server:
    - https://www.ibm.com/docs/en/zos/2.5.0?topic=zcx-setting-up-zos-nfs-server
  - Getting started with AT-TLS:
    - https://www.ibm.com/docs/en/zos/2.5.0?topic=protection-getting-started-tls
  - ICR Registry for HAPROXY container image:
    - https://ibm.github.io/ibm-z-oss-hub/main/main.html