Running WebLogic on Docker: Challenges & Workarounds

BOF7537

Bruno Borges - @brunoborges
Principal Product Manager
Oracle Cloud PaaS, Oracle Corp.

JavaOne™ ORACLE

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Speaker

Bruno Borges

Principal Product Manager
Oracle Cloud Platform

@brunoborges

Program Agenda

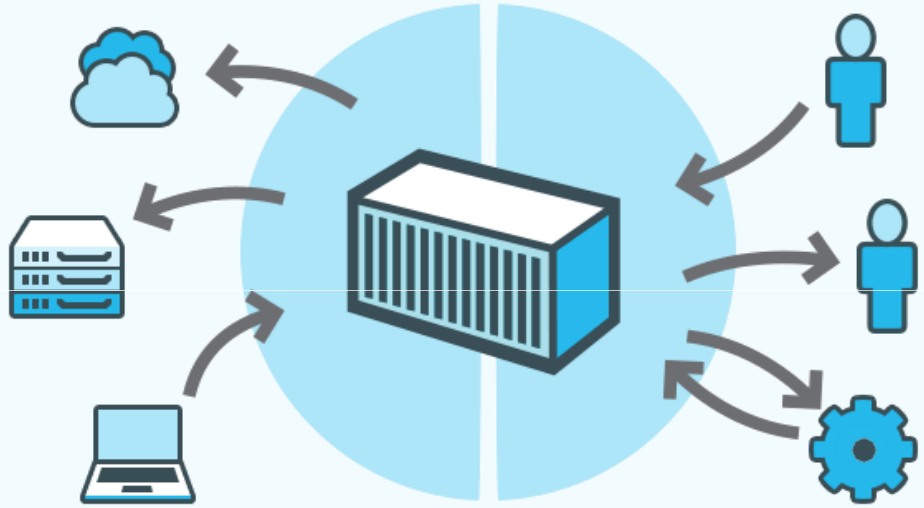
1. What is Docker?
2. Common, and Best Practices For Running Containers
3. Deploying WebLogic on Docker
4. WebLogic Server on Docker at GM
5. Demo

What is Docker?

And why should we care?

What Is Docker?

An open platform for distributed applications



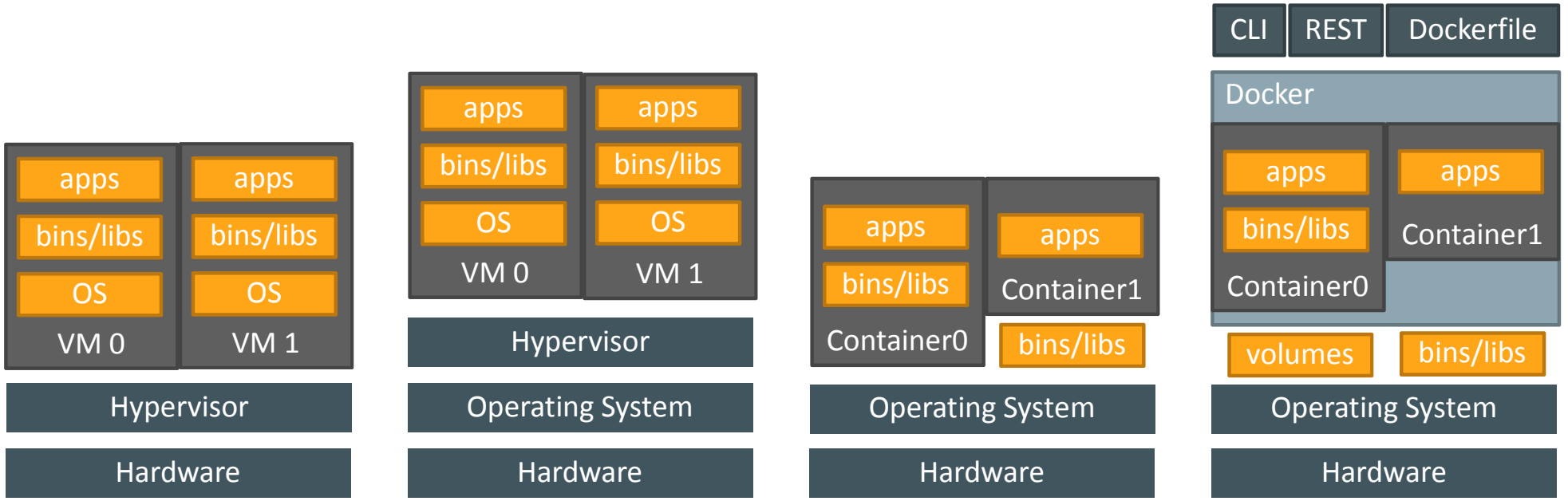
Docker Engine

A portable, lightweight application runtime and packaging tool.

Docker Hub

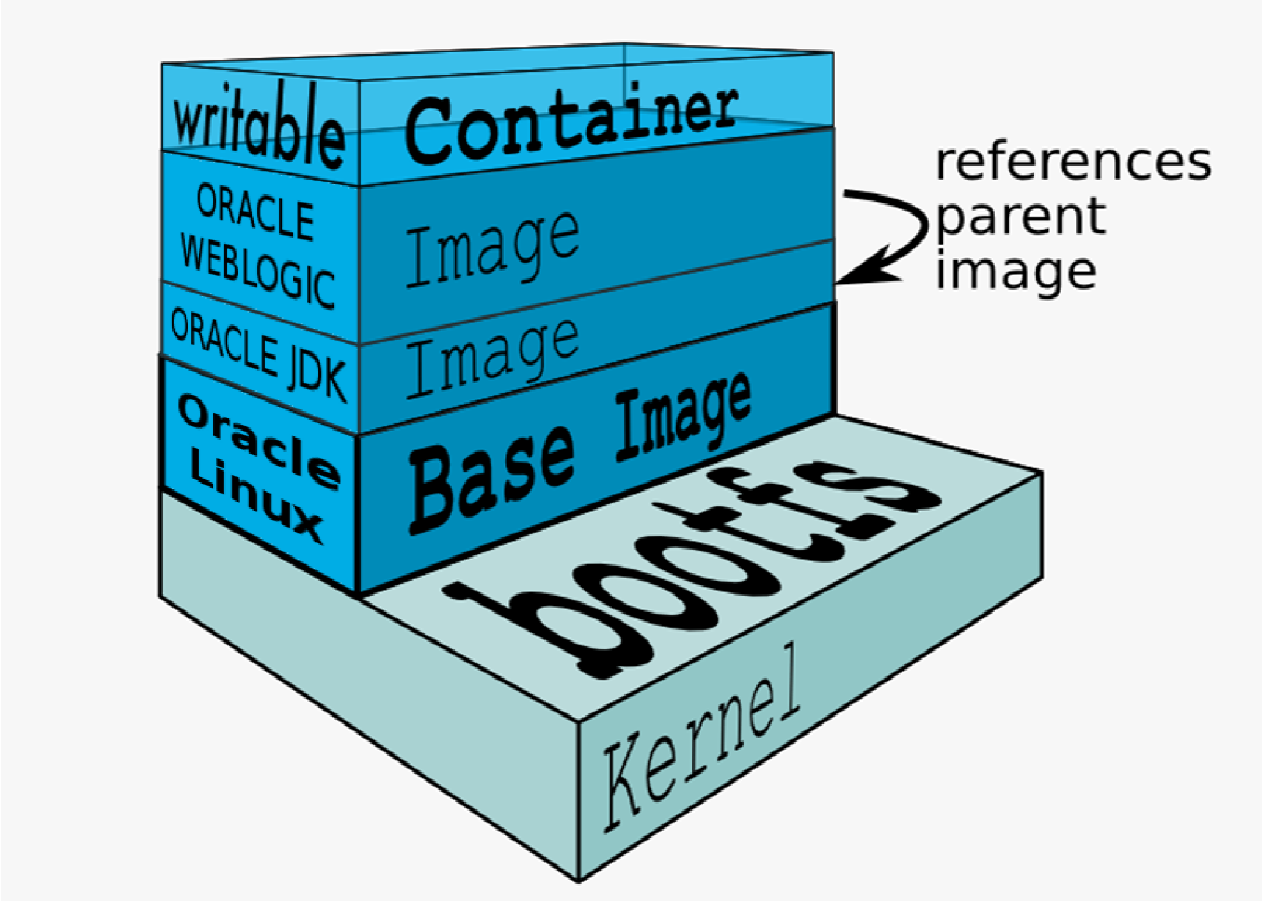
A cloud service for sharing applications and automating workflows.

Comparing Docker against LXC and Hypervisors



Docker Images

- Containers have writeable file system
- Recipes, the Dockerfiles, can be easily shared
  - Container as code
- Images have layered file systems
- Images can be extended
- Images can be shared as binary files



Example of a Dockerfile (image recipe)

```
# Pull from base image of Oracle Linux and install pkgs
FROM oraclelinux:7
RUN yum install -y unzip java-1.7.0-openjdk-headless || :
RUN yum clean all
# Download and extract GlassFish
RUN curl -O -L http://bit.ly/glassfish-4_1_zip
RUN unzip glassfish-4_1_zip
# Default CMD upon container execution
CMD /glassfish4/bin/asadmin
```

Building a container image

```
[bruno@orcl:~/gf-docker]$ ls Dockerfile*
Dockerfile
[bruno@orcl:~/gf-docker]$ sudo docker build -t glassfish:4.1 .
Sending build context to Docker daemon
Step 0 : FROM oraclelinux:7.0
 ---> 5f1be1559ccf
Step 1 : RUN yum install -y unzip java-1.7.0-openjdk-headless || :
 ---> Running in 7b1583feb2bc
Step 2 : RUN curl –O –L http://bit.ly/glassfish-4_1_zip
 ---> 4abe64ef7934
Step 3 : RUN unzip glassfish-4_1_zip
 ---> 1ac729d9809a
Step 4 : CMD /glassfish4/bin/asadmin
 ---> Running in 6aab421a83ff
Successfully built 8f1106a1d7d0
```

Creating and running a container

```
[bruno@orcl:~]$ sudo docker images
REPOSITORY      TAG     IMAGE ID        CREATED             VIRTUAL SIZE
glassfish       4.1     8f1106a1d7d0    3 minutes ago       1.081 GB
[bruno@orcl:~]$ sudo docker run -ti glassfish:4.1
Use "exit" to exit and "help" for online help.
asadmin> start-domain
Waiting for domain1 to start .....
Successfully started the domain : domain1
domain  Location: /root/glassfish4/glassfish/domains/domain1
Admin Port: 4848
Command start-domain executed successfully.
asadmin>
```

Accessing a service/application running within a container

```
[bruno@orcl:~]$ sudo docker ps
CONTAINER ID    IMAGE             CREATED          STATUS
9ebf7544f8e9    glassfish:4.1     3 minutes ago    Up 3 minutes
[bruno@orcl:~]$ sudo docker inspect 9ebf7544f8e9 | grep IPAddress
           "IPAddress": "172.17.0.8",
[bruno@orcl:~]$ firefox http://172.17.0.8:4848
[bruno@orcl:~]$ ifconfig docker0
docker0   Link encap:Ethernet  HWaddr 56:84:7a:fe:97:99
          inet addr:172.17.42.1  Bcast:0.0.0.0  Mask:255.255.0.0
```

Common, and Best Practices For Running Containers

Challenges and Workarounds for Certifying a Java EE Application Server on Docker containers

Images and Containers

- Customers expect vendors to publish base images of their products pre-installed, perhaps pre-configured. Always with extensibility support.
- Customers may extend images with their own configurations.
- Images should have a repeatable state.
  - There shouldn't be requirements for modifying containers after they are created based on an image, before applications can actually work. Minimum dependency on host environment (ideally none).
- Containers are expected to be disposable.
  - Since images can be easily built and extended, and containers repeatedly created with same initial state, containers are not expected to be lively patched/modified/upgraded. These changes should go into the image.

Networking and Linking

- Containers are by default bond to host's virtual network interface bridge, **docker0**.
- Containers may be configured to run on specific bridges, thus allowing for specific groups of containers, although on same host, to be on completely isolated sub-network
- Containers can be easily linked, allowing network mapping by their hostnames
  - `$ sudo docker run -ti --link db_container_0:db myapp /bin/bash`
  - `# root@myapp> ping db`
  - Ping above will lead to "db_container_0"
- Containers from different hosts may be connected together.
  - Requires advanced networking setup not provided by Docker OOTB. Workarounds rely on NAT and some products have limitations for proper communication between peers.

Separating binaries from User-defined files and folders

How to increase repeatability and portability of Docker containers

Data inside a container

- Containers are “instances” of images. Each gets its own filesystem on a copy-on-write approach. Docker uses Union Filesystems.
- Data of a container can be “persisted” in three ways:
  - Container Union Filesystem (default)
    - copy-on-write
  - Data Volume
    - Folder mapped to a folder on host
  - Data Volume Container
    - Folder mapped to a folder on another container

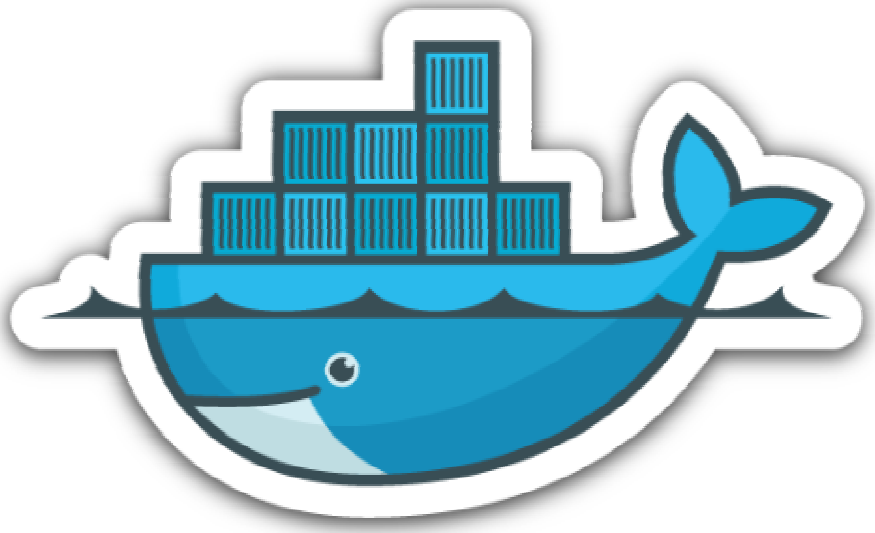
Data Volumes – Host Mapped

- User mounts a host folder as a Data Volume folder inside the container
  - Containers with running processes and binaries only
  - Modifiable files/folders are maintained and updated on host mapped volume
- **Pros**
  - Increases repeatability
  - Allows patching of binaries without losing user-defined configurations
  - Good performance for disk IO operations
- **Cons**
  - Increases dependency on hosts
  - Decreases portability

Data Volume Containers – Data-Only Containers

- User mounts a container folder as a Data Volume on another container
  - One container for holding user-defined configurations, files and folders
  - Another container for binaries and processes
- **Pros**
  - Increases repeatability
  - Increases portability
  - Decreases dependency on hosts
  - Allows patching of binaries without losing user-defined configurations
- **Cons**
  - Performance issues if constant, high disk IO is required

WebLogic Server on Docker

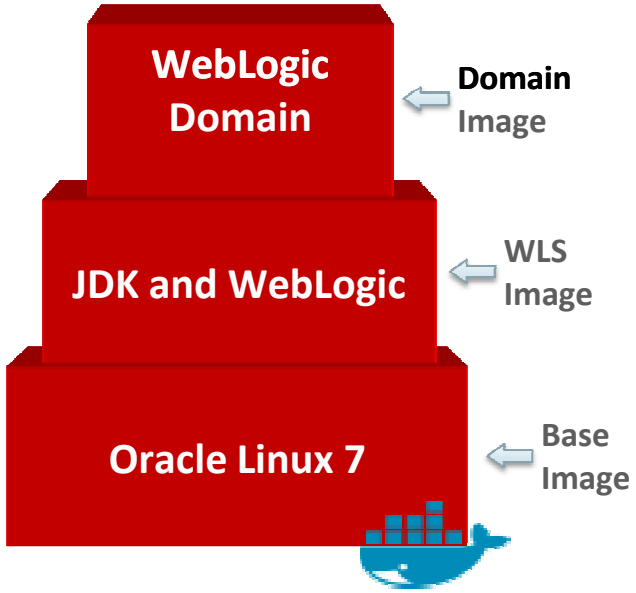


Certification of WebLogic running on Docker

WLS Version	JDK Version	HOST OS	Kernel	Docker Version
12.2.1 (EE 7)	8	Oracle Linux 6 UL 6	Unbreakable Enterprise Kernel Release 3 (3.8.13)	1.7+
12.2.1 (EE 7)	8	Oracle Linux 7	Unbreakable Enterprise Kernel Release 3 (3.8.13) or Red Hat Compatible Kernel (3.10)	1.7+
12.2.1 (EE 7)	8	Red Hat EL 7	Red Hat Compatible Kernel (3.10)	1.7+
12.1.3 (EE 6)	7/8	Oracle Linux 6 UL 5	Unbreakable Enterprise Kernel Release 3 (3.8.13)	1.3.3+
12.1.3 (EE 6)	7/8	Oracle Linux 7	Unbreakable Enterprise Kernel Release 3 (3.8.13) or Red Hat Compatible Kernel (3.10)	1.3.3+
12.1.3 (EE 6)	7/8	Red Hat EL 7	Red Hat Enterprise Linux Kernel (3.10)	1.3.3+

WebLogic Docker Images

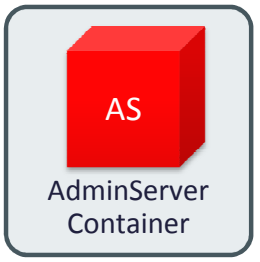
- Base Image
  - Oracle Linux 7 or RedHat 7 (already available on Docker Hub)
- Install Image
  - Download WebLogic installers and JDK
  - WebLogic Dockerfiles to extend base image with Oracle Linux 7, JDK 8, and the WebLogic Server 12.2.1 installation (Full or Developer installation)
- Domain Image
  - Edit sample Dockerfile posted on GitHub to extend WLS install image to create a domain configuration.



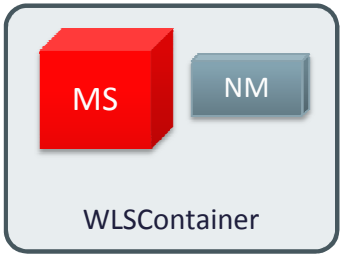
Running WebLogic Server Docker Container

- From the WebLogic Server image you can start two different container configurations
  - Run Admin Server Container
    - Default container.
  - Run Managed Server Container
    - Start NodeManager
    - Set network to connect Managed Server Containers to Admin Server Container.
    - Call WLST script to automatically add the NodeManager as Machine into the domain and configure a Managed Server in the machine.
- Using these containers you can create different topologies.
  - WebLogic Server Docker Topology
  - Containerized Oracle WebLogic Server Applications

Bulding Block 1

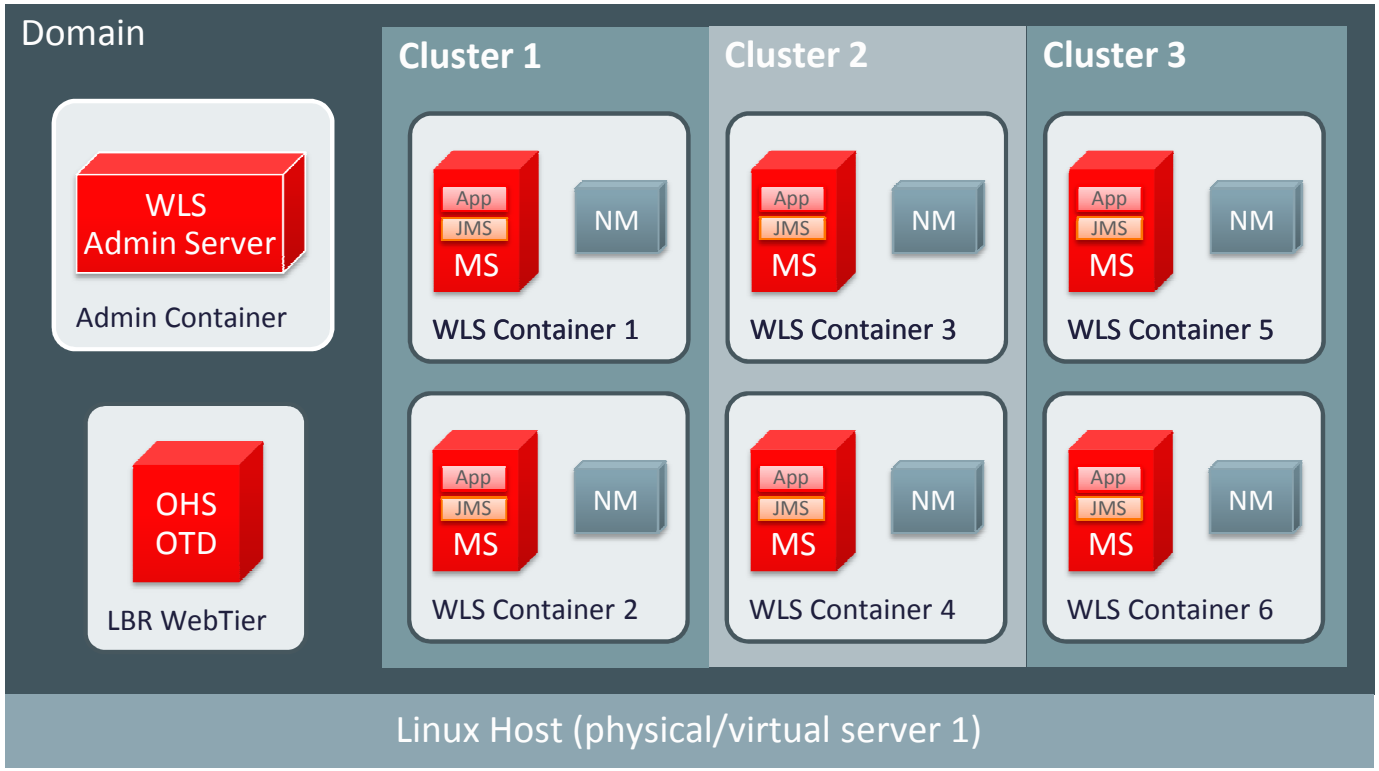


Bulding Block 2



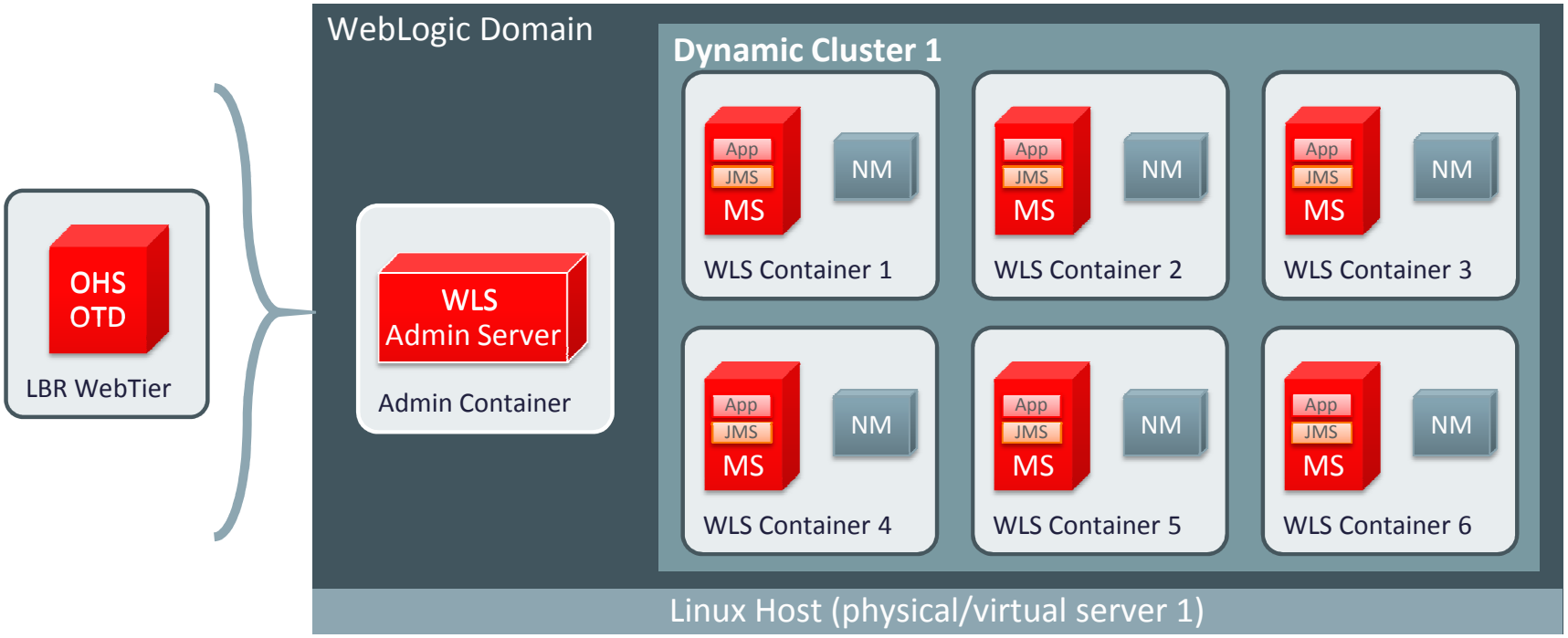
WebLogic Server Docker Topology on Single Host

- Good for traditional-like deployments
- Easy to deploy
- Easy to scale up and down
- Good for developers
- No need to install or configure anything on host except Docker binaries



Topology Single Host – Example

Expand Dynamic Cluster: Add “Machines” Into Domain



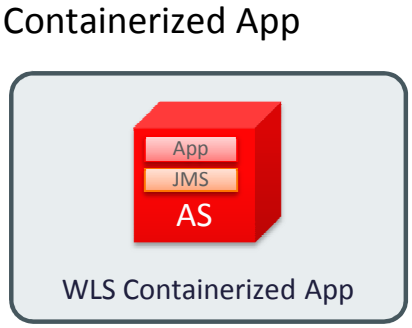

```
# docker run –-name wlsadmin –d mywlsimage startWebLogic.sh
# docker run –-link wlsadmin:wlsadmin –d mywlsimage createServer.sh
# docker run –-link wlsadmin:wlsadmin –d mywlsimage createServer.sh
```

Webogic Server on a Single Host Topology - Summary

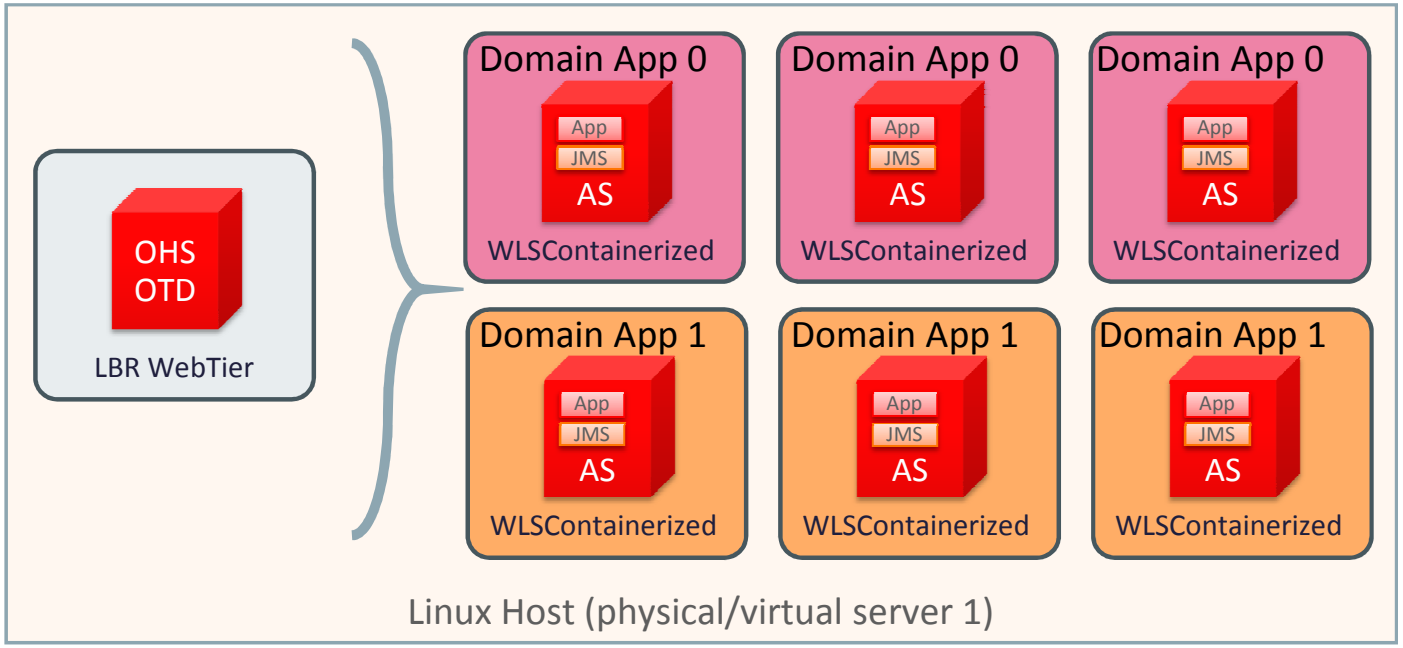
- Can simulate a production environment consuming less resources than traditional VM virtualization
- Can simulate scaling up/down environments with Dynamic Clustering
- Developers can share a complex WebLogic domains with production-like topology
- Can be used in production
  - It is certified and supported by Oracle
- Has its downsides due to Docker designs and principles

Topology - Containerized Apps – Building Blocks

- Docker Image includes
  - Domain has all resources pre-defined
  - Apps and shared libraries deployed upfront
  - No ManagedServers or Clusters defined
- Building Blocks
  - One container designed to run only AdminServer
    - All resources, shared libs, and deployments targeting the AdminServer
- Benefits
  - The “Docker-way” for containerized apps/services
  - Container is easily repeatable
  - Each container is an “instance” of the same WebLogic domain



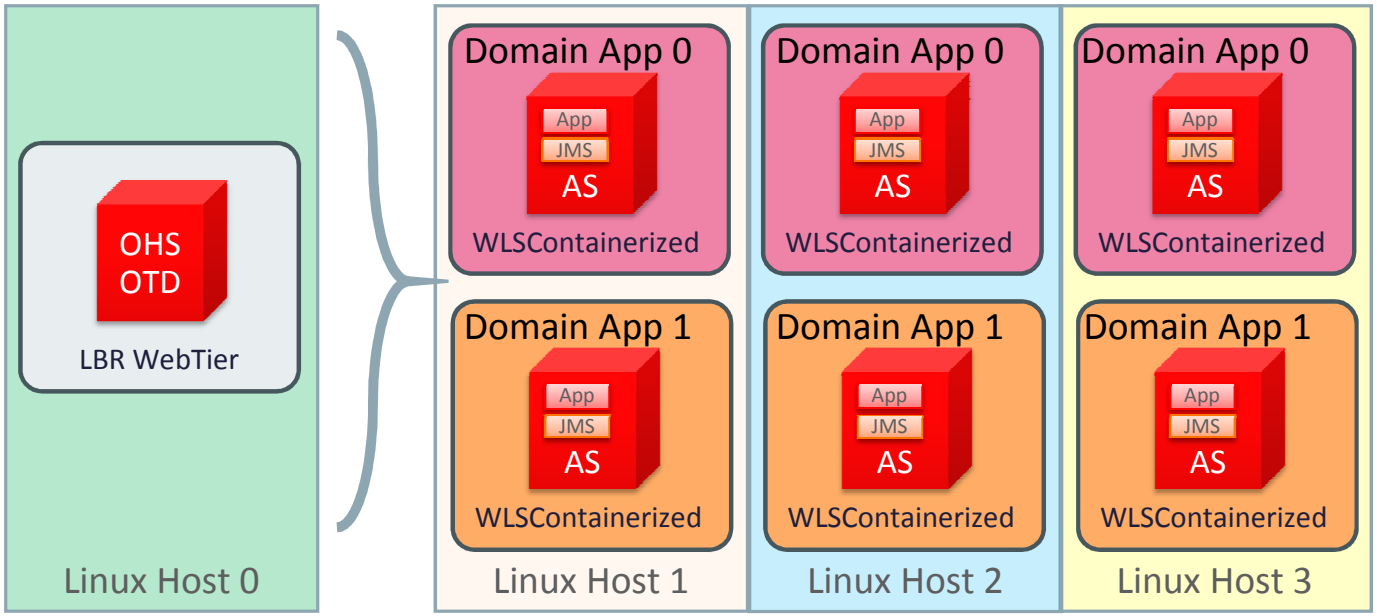
Topology - Containerized Apps – Single Host




```
root@host_1 # docker run –d mywlsapp1 startWebLogic.sh
root@host_1 # docker run –d mywlsapp1 startWebLogic.sh
root@host_1 # docker run –d mywlsapp1 startWebLogic.sh
```

Topology - Containerized Apps – Multiple Hosts

Load Balancing only. No real clustering between Servers

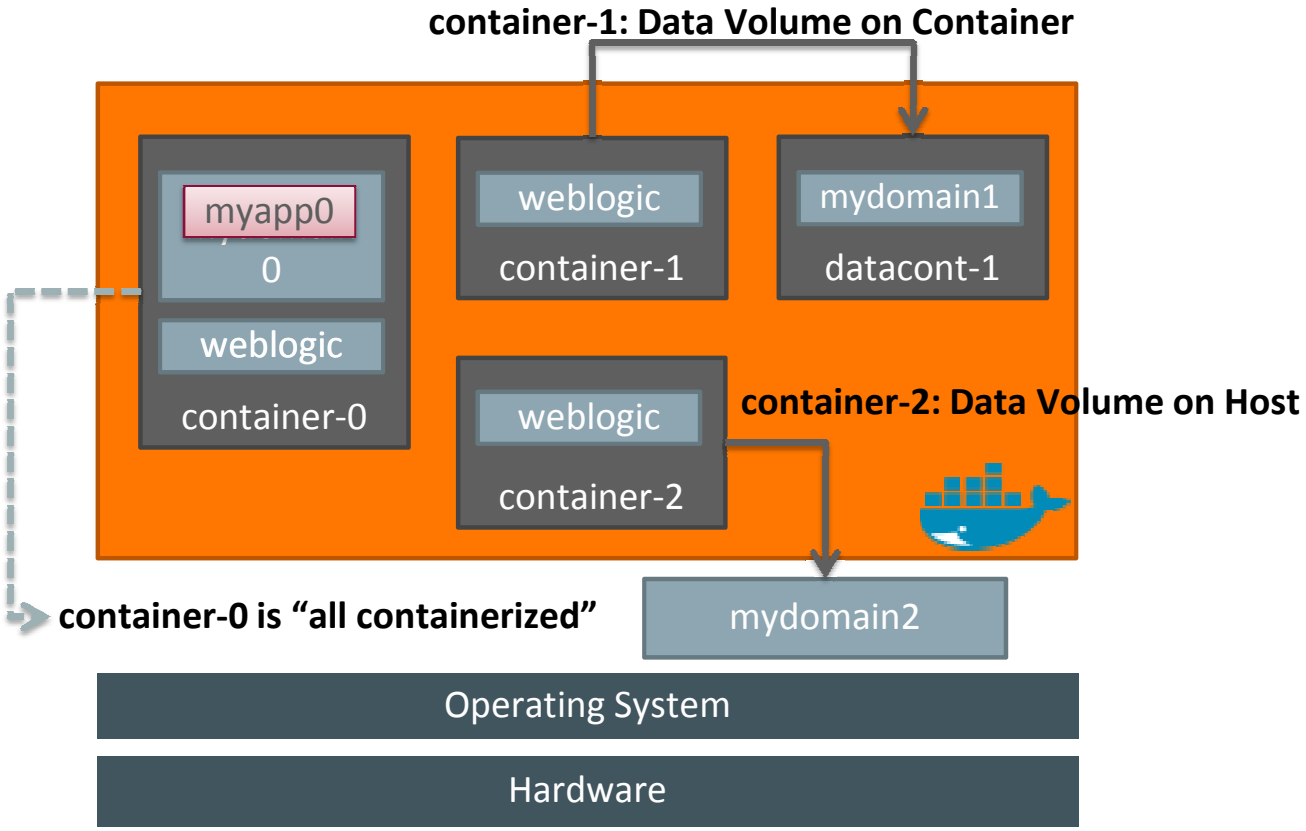



```
root@host_1 # docker run –d mywlsapp0 startWebLogic.sh
root@host_2 # docker run –d mywlsapp0 startWebLogic.sh
root@host_3 # docker run –d mywlsapp0 startWebLogic.sh
```

Containerized Oracle WebLogic Server Applications Topology - Summary

- WebLogic, application(s), and required resources are containerized as one single isolated process
- Highly scalable horizontally; portable
- Easy to run
  - No dependencies on other containers for orchestrated, complex domain
  - No network challenges (containers don't communicate with each other)
- Downsides
  - Not capable of doing HA for transactions, JMS, web sessions, EJBs, etc.

WebLogic on Docker – Domain Location Examples



- Keep the file system on a separate container.
  - Easy to move containers
- Keep the file system on the host.
  - Better I/O performance
- Containers with containerized applications have no negative side effects of loosing their file system.

Upgrading /Patching WLS Domains

- Upgrade or apply patch by extending WebLogic Docker image

- Back up your domain folder
  - Use the **docker cp** command to copy your domain folder to a host, or on a data-only container

- Start new containers from the patched image

Known Issues with Docker (latest versions)

- IP addresses change after a container restart
  - Container restart will leave servers not being able to communicate to each other
  - DNS server configured in the container
  - Rebuild configuration
- No multicast support
  - Unicast clustering is recommended if on Single Host

Docker 1.8+ three important enhancements

- **Multi Host support (experimental)**- SDN functionality through its recent SocketPlane acquisition together with 2 important features:
  - DNS Server, apps resilient to change in ip-address
  - VXLAN (virtual extensible LAN), allows application's microservices to reside on any member of a Swarm, a native Docker cluster.
- **Data Volumes (experimental)**- Docker Plugin Architecture**,** third-party container data management solutions to provide data volumes for containers.
  - Available plugins for container data volume management Flocker
- **Security –** Docker Image Signing and Verification using The Update Framework (TUF)

Configuring Containers for the Environment

- Containers can run in any environment.
- Docker applications tend to use environment variables to pass configuration information into the container and its application.
- More complex applications will need to use a Configuration Service,
  - To allow to retrieve configuration parameter values by name
  - To group configuration parameter into named sets for retrieval by applications
  - To create and managed configuration data across multiple environments
  - To define the “schema” of its configuration data
- Application can use third-party providers of distributed configuration management systems. Examples of these include Zookeeper , Consul, …

Oracle WebLogic Server and Docker Containers [CON8629]

2016 Chevrolet Camaro

Oracle OpenWorld

Daniel Parsons, Core Engineering Lab Manager & Business Continuity
General Motors Information Technology
October 27, 2015

Docker Advantage Exploration

- Rapid Application Deployment
  - Use Docker technology for small set of applications for our core standard release sets
  - Many server templates down to three or 4 depending on technology tower
  - Free to be creative, Debug at will, try again with supported image
- Portable across machines
  - Speeds and feeds through development cycle reducing complexity with additional OS release levels
  - Develop team members have full access to their VM's to work within their own single host. No need for additional access requests during Dev/Test in Enterprise Proving Grounds
  - Containers provide application isolation so no need for full operating system, better abstraction then small VM
- Package apps
  - Many homegrown apps that need special attention
  - App can control its own runtime dependencies
  - Pathway towards supportability, feature enhancements for Ops
- Not just apps to run in containers
- Keep ROI in check

Future Docker work

Docker Future for Oracle WebLogic and Beyond

- Integration with other DevOps technologies (e.g. Maven)
- Publish WLS Docker images on a Docker Hub repository
- Optimizing Docker Image size to make easy to move across the network
  - Optimize size of each layer that makes the image
- Broader product line adoption, with recommended and certified topologies
  - SOA, Enterprise Manager, OHS/OTD, Database, etc.

Main Links and Contacts

- Oracle Docker Projects
  - Dockerfiles on GitHub - github.com/oracle/docker
  - Oracle Linux 6 Docker docs and Oracle Linux 7 Docker docs
  - Repository of Oracle Linux images: public-yum.oracle.com/docker-images/
    - Also available on Docker Hub Repository (preferred)
- WebLogic on Docker Containers
  - Public Whitepaper PDF
  - Official Public Announcement with Certification Details
- WebLogic Product Management Team
  - Monica Riccelli monica.riccelli@oracle.com
  - Will Lyons will.lyons@oracle.com
- WebLogic Docker on GitHub
  - Bruno Borges bruno.borges@oracle.com

Blogs and Articles

- Oracle Linux
  - Oracle Linux Containers and Docker
- Oracle WebLogic
  - Docker, Java EE 7, and Maven with WebLogic 12.1.3
  - WebLogic with Docker in the Cloud
- Docker
  - Getting to know Docker – A better way to do virtualization?
  - Data-Only Containers for Docker
  - Docker Documentation

JavaOne™

ORACLE®