Best Practices for Deploying Oracle RAC on Docker

ORACLE WHITE PAPER | APRIL 6, 2019

ORACLE

DISCLAIMER

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. This material has not been submitted to any formal Oracle test and is published as is. It has not been the subject of rigorous review. Oracle assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by Oracle for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk. Oracle may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents.

Best Practices for Deploying RAC on Docker Table of Contents

Introduction 5

Solution Overview 6

Oracle Clusterware 6

Oracle Automatic Storage Management (ASM) 6

Oracle RAC 6

Oracle Linux 6

Oracle Container Runtime for Docker 6

Deployment Best Practices 6

Step 1: Install and Configure Oracle Linux 7 6

Step 2: Install and Configure Docker Engine 6

Step 3: Oracle RAC on Docker Deployment Scenarios 7

Step 4: Download Docker Repository Files and Oracle Software 10

Step 5: Build the Docker Image 11

Step 6: Configure Docker Host Environment 12

Step 7: Deploy Oracle RAC on DOCKER 12

Step 8: Run the ADDNode Script 12

Step 9: Connect To RAC 13

RAC with Weblogic on Docker Container 13

Deployment Steps: Oracle RAC with WebLogic on Docker 14

Step 1: Install and Configure Oracle RAC on Docker 14

Step 2: Install and Configure WebLogic 14

Step 3: Download the Oracle Weblogic Server Software 15

Step 4: Build the Oracle WebLogic Image 15

Step 5: Create the Medrec Image 15

Step 6: Download WebLogic Supplemental Installer 15

Step 7: Build the Image 15

Step 8: Run the Container 15

Step 9: Access the Console 15

Conclusion 16

INTRODUCTION

Oracle Real Application Clusters (Oracle RAC) allows an Oracle Database to run any packaged or custom application across a set of clustered servers. This provides continuous database service uptime during node and instance failures, most planned maintenance activities, and during Oracle RAC expansion. If a clustered node fails, the Oracle Database service continues to run on one of the surviving nodes. When more processing power is needed, another node can be added without interrupting user access to the database or data.

Docker is an open source containerization platform that accelerates the development lifecycle and facilitates the deployment of Microservices. Docker has gained a lot of support in the IT community because of its agility and, as such, many organizations are running their services in Docker containers.

This white paper describes best practices for configuring Oracle RAC on Docker. Oracle RAC on Docker provides an elegant solution to many of the challenges faced by DevOps today.

- **Integration Challenges**: Building a database environment for applications requires solving many software dependencies. With Docker, developers or the database administrator can build an Oracle RAC Docker image with all the package dependencies, Oracle Grid Infrastructure Software, Oracle Database release updates, and application deployment scripts.

- **Environment Provisioning Challenges**: In test and development environments, users require quick environment provisioning. Without a packaged database Docker image, it will take a lot of time and skills. However, using Docker, users can download the Oracle RAC image and trigger the deployment of RAC and the application schemas. Users get a complete software stack included in the image and can quickly provision the environment.

- **Testing Challenges**: Since Oracle RAC is running in a Docker container, users can freeze the test environment and share it with other testers and developers, enhancing collaboration and productivity.

SOLUTION OVERVIEW

This section introduces Oracle Grid Infrastructure, Oracle RAC, Oracle Linux, and Docker.

Oracle Clusterware

Oracle Clusterware enables servers to coordinate with each other, so that they appear to function as a highly available unit. This combination of servers is commonly known as a cluster. Although the servers are standalone servers, each server has additional processes that communicate with other servers, making the individual servers appear as one system to applications and end users. Oracle Clusterware also monitors the components within the cluster, restarting or failing over resources to ensure high availability.

Oracle Clusterware provides the infrastructure necessary to run Oracle RAC. Oracle Clusterware also manages resources, such as virtual IP (VIP) addresses, databases, listeners, services, and many more.

Oracle Automatic Storage Management (ASM)

Oracle Automatic Storage Management (ASM) is the recommended cluster volume manager which can be used for both, Oracle RAC and Single Instance Oracle Databases. Oracle ASM simplifies storage management through the principle of “stripe and mirror everything” (SAME). Intelligent mirroring capabilities allow administrators to define 2- or 3-way mirrors to protect vital data. When a read operation identifies a corrupt block on a disk, Oracle ASM automatically relocates the valid block from the mirrored copy to an uncorrupted portion of the disk.

Oracle RAC

Oracle Real Application Clusters (Oracle RAC) enables multiple interconnected instances to cooperatively provide an Oracle Database Service. In an Oracle RAC environment, the database runs on two or more systems in a cluster, while appearing to end users and applications as a single database. The result is a single database that spans multiple hardware systems, enabling Oracle RAC to provide high availability and redundancy during failures in the cluster. Oracle RAC accommodates all system types, from read-only data warehouse systems to update-intensive online transaction processing (OLTP) systems.

Oracle Linux

Operating systems, containers, and virtualization are the fundamental building blocks of modern IT infrastructure. Oracle combines them all into one integrated offering: Oracle Linux. Operating on your choice of hardware—in your data center or in the cloud—Oracle Linux provides the reliability, scalability, security, and performance for demanding enterprise workloads.

Oracle Container Runtime for Docker

Oracle Container Runtime for Docker allows for an easy creation and distribution of applications across Oracle Linux systems and other operating systems that support Docker. Oracle Container Runtime for Docker consists of the Docker Engine, which packages and runs the applications, and integrates with the Docker Hub, Docker Store and Oracle Container Registry to share the applications in a Software-as-a-Service (SaaS) cloud.

DEPLOYMENT BEST PRACTICES

Step 1: Install and Configure Oracle Linux 7

When setting up Oracle RAC on Docker, use Oracle Linux 7.x with UEK4. For detailed installation steps and system requirements, please refer to the Installation Guide for Oracle Linux Release 7 under Oracle Linux Documentation.

Step 2: Install and Configure Docker Engine

Oracle RAC on Docker supports the non-privilege mode feature. This allows Oracle RAC to safely and securely run on a single host or multiple hosts without interference from the other Docker Containers. Configuring Oracle Container Runtime for Docker involves many steps; please execute them in the order given in the documentation. To get more details about each section described below, you can refer to Oracle Container Runtime for Docker User's Guide.

Step 3: Oracle RAC on Docker Deployment Scenarios

DEPLOYMENT ON A SINGLE HOST WITH PUBLICLY ACCESSIBLE IP ADDRESSES

Multiple Oracle RAC Clusters can be deployed on a single Docker host. Every Oracle RAC Cluster will have its own storage and assigned IPs running in a Docker container. You can connect to Oracle RAC Databases running on single hosts using Scan IPs directly from your application, which might be running on some other host. Docker MACVLAN bridge is required to expose your Docker container IPs to your physical network.

DEPLOYMENT ON A SINGLE HOST WITH CONTAINER-ONLY IP ADDRESSES

If SCAN IPs are not accessible from outside your container, you can use Oracle Connection Manager to access the Oracle RAC Cluster from your application. Oracle Connection manager acts like a Database proxy server, providing a single publicly available IP address enabling connections to multiple container-private IP addresses. Figure 1 illustrates the architecture of running RAC on Docker on a single host with Oracle Connection Manager.

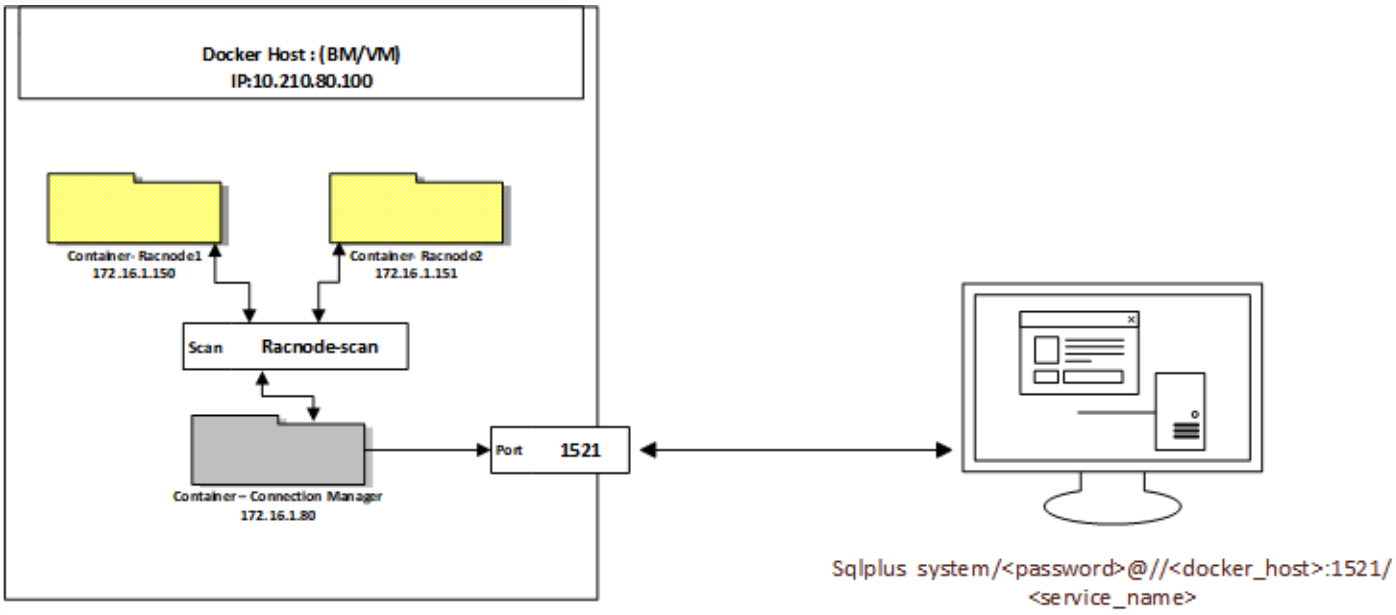

Figure 1 RAC on Docker deployment scenario with Oracle Connection Manager

DEPLOYMENT ON MULTIPLE HOSTS

Multiple Oracle RAC Clusters can be deployed on multiple hosts on Docker. Every Oracle RAC Cluster will have its own storage and assigned IP addresses within the Docker container. To run Oracle RAC on Docker on multiple hosts, you need container IPs to be on same subnet and reachable between the containers running on the different hosts. Docker MACVLAN bridge connects the containers directly to your public physical network enabling containers running on different hosts to communicate with each other.

Figure 2 illustrates the architecture of running Oracle RAC on Docker on multiple hosts.

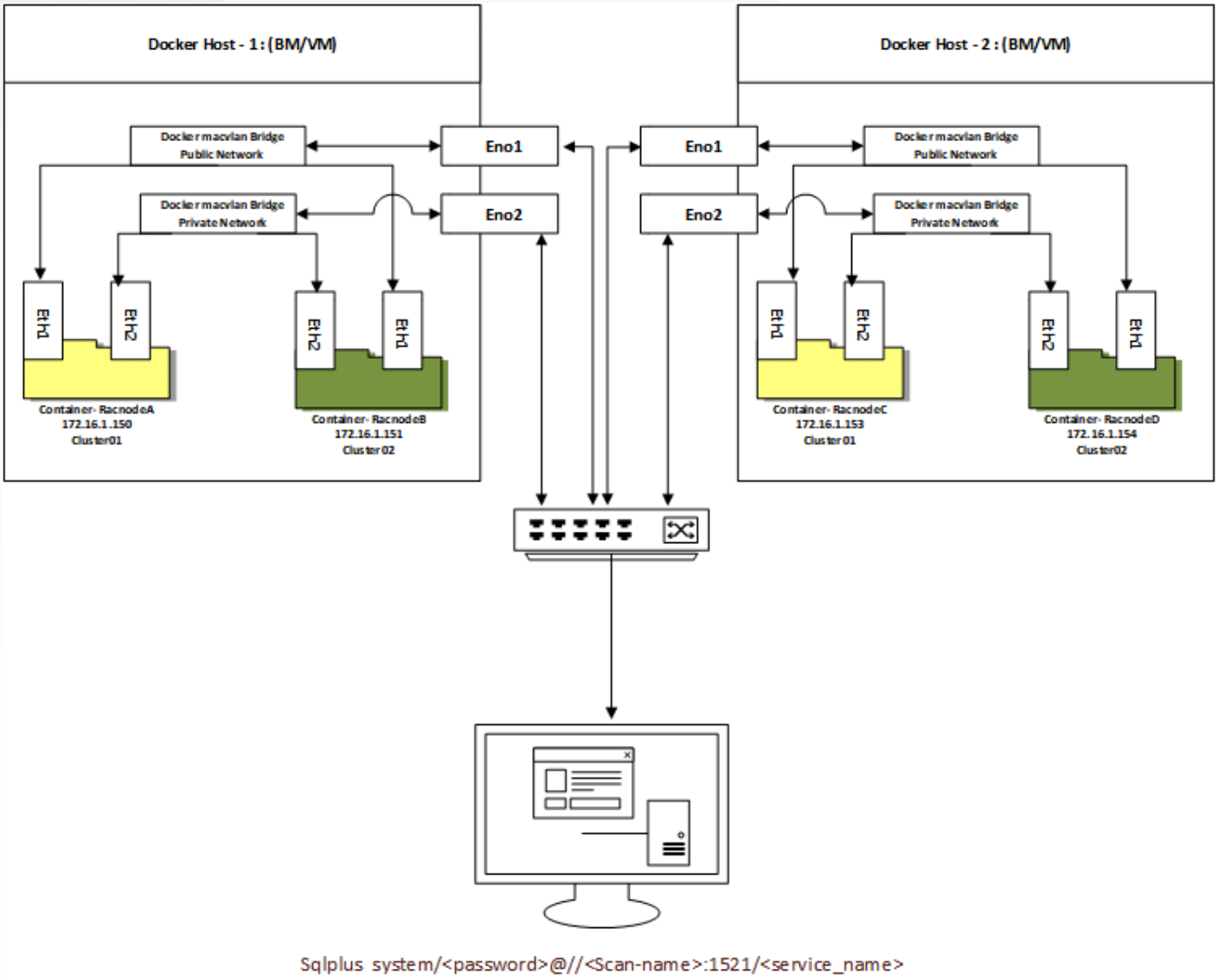

Figure 2 RAC on Docker when RAC is deployed on user network using the Docker Macvlan Driver on Multihost

DEPLOYMENT USING BLOCK STORAGE

Both Oracle Clusterware and the Oracle RAC database use files that must be available to all the nodes in the cluster. When using either direct-attached storage (DAS) or storage area network (SAN) for Oracle ASM, each disk must have a partition table. Oracle recommends creating exactly one partition for each disk that encompasses the entire disk.

For details, please refer section **Supported Storage Options for Oracle Grid Infrastructure** in the Oracle Grid Infrastructure Installation and Upgrade Guide for Linux.

If you specify the "--privileged=true" option to the commands "docker create" or "docker run", the container has access to all the devices on the host, which can present a security risk. For more precise control, you should plan your storage requirements ahead of the Oracle RAC installation and allocate specific block devices to containers based on your Oracle RAC DB storage requirement.

To allocate specific block devices on the host available to specific containers, you can use the "—device" option with the docker run

and docker create commands: --device=host_devname[:container_devname[:permissions]]

Figure 3 illustrates the architecture of deploying RAC on Docker using block devices as shared storage.

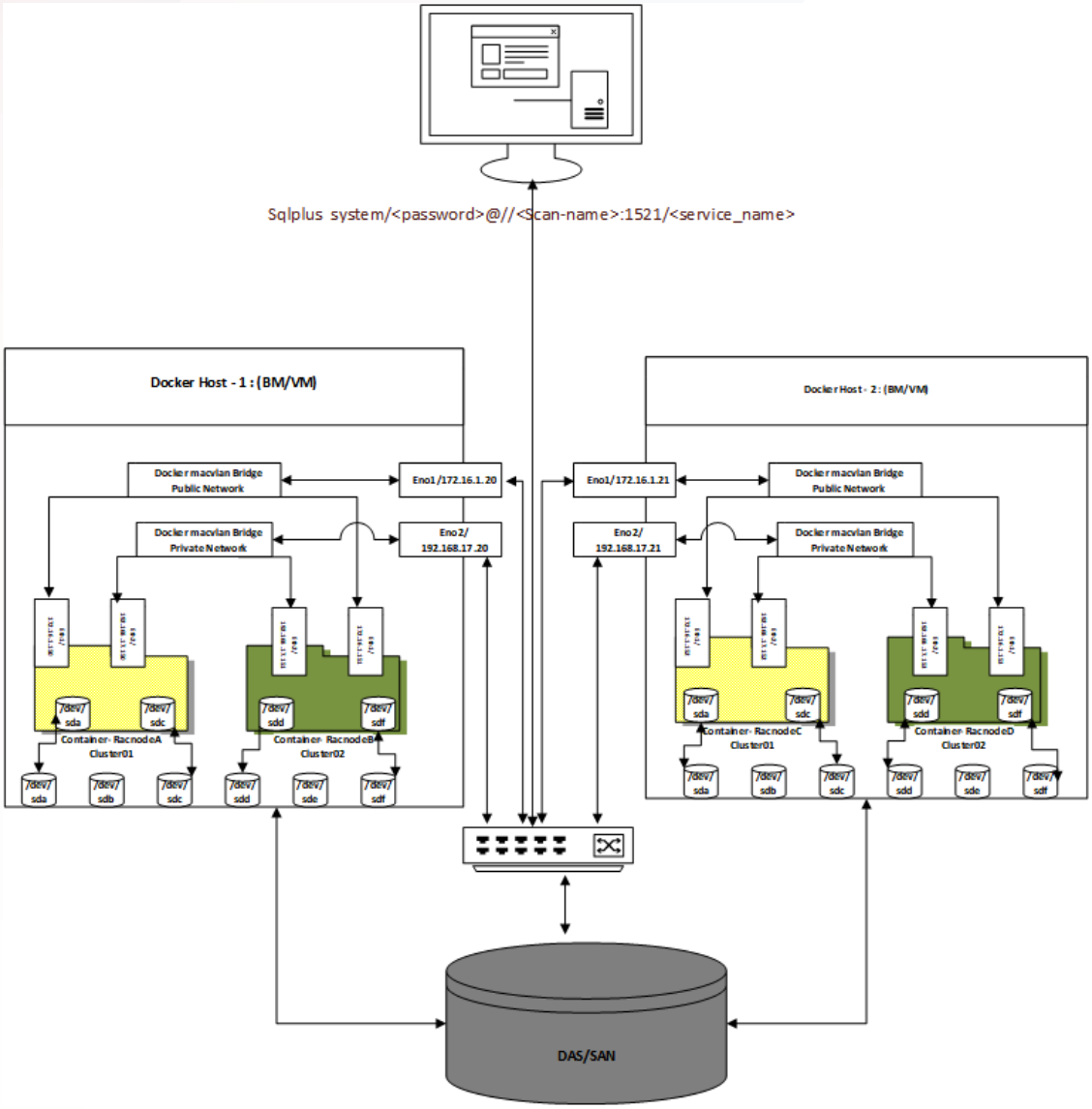

Figure 3 Oracle RAC on Docker using Block storage as shared storage

DEPLOYMENT USING NAS STORAGE

If you have a certified NAS storage device, then you can create zero-padded files in an NFS mounted directory and use those files as disk devices for the Oracle ASM disk group. In this case and to create a zero-padded file of 1GB in size, use the dd command:

Example: dd if=/dev/zero of=/oradata/asm_disk01.img bs=1M count=1000

For more details, refer to **Supported Storage Options for Oracle Grid Infrastructure** and **Creating Files on a NAS Device for Use with Oracle Automatic Storage Management** in the Oracle Grid Infrastructure Installation and Upgrade Guide for Linux.

If you do not have block or NAS devices to run Oracle RAC on Docker, you can deploy the OracleRACStorageServer image, which emulates an NFS server and an NFS volume exposed to Oracle RAC container. The following figure explains the usage of RAC on Docker with NAS devices.

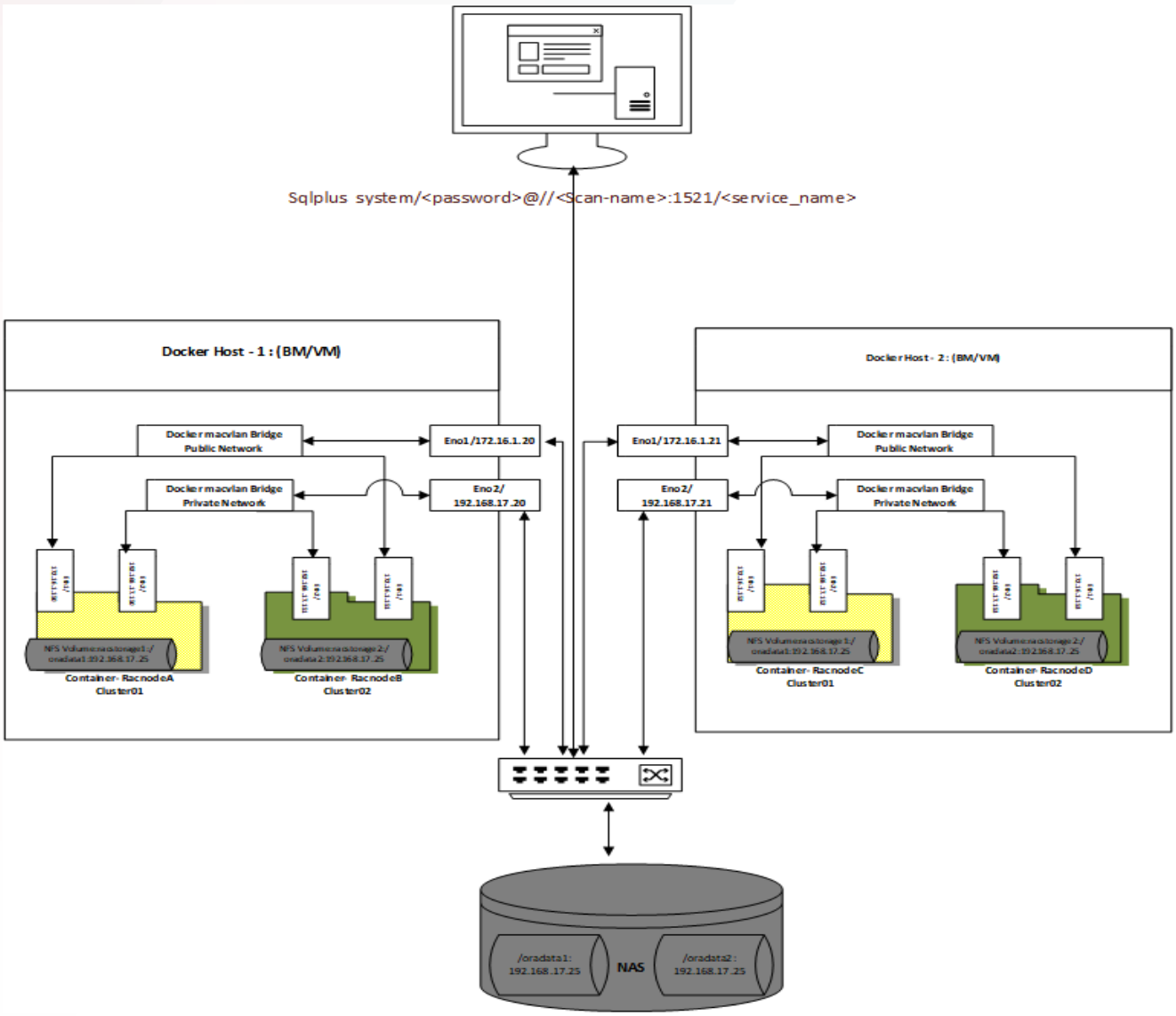

Figure 4 Deployment using NAS as shared storage device

Note: the OracleRACStorageServer storage image is supported only for test and dev deployments.

Step 4: Download Docker Repository Files and Oracle Software

DEPLOYMENT OVERVIEW

To build the Oracle RAC on Docker image, clone the Oracle/docker-images repository files from GitHub and build the images based on your environment. The Oracle RAC Docker image will be built based on the Oraclelinux:Slim-7 image. Figure 5 illustrates the complete build scenario for the Oracle RAC Docker Image, Oracle Connection Manager image, and RAC Storage image.

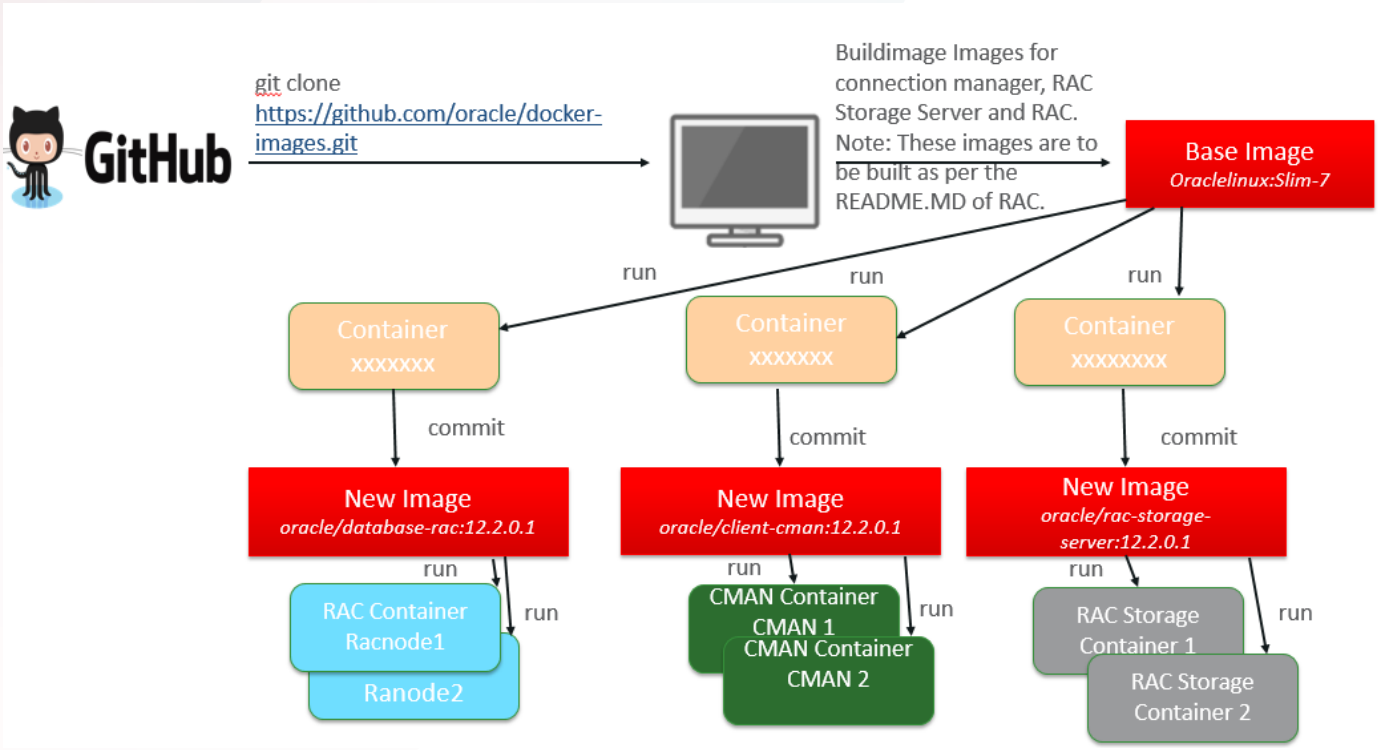

Figure 5 Steps to clone Oracle/docker repository files from GITHUB

CLONE ORACLE RAC ON DOCKER FILES FROM GITHUB

To clone the Oracle/docker-images repository files from GitHub to build the image, use the following command:

```
# git clone https://github.com/oracle/docker-images.git
```

You need to have connectivity to GitHub from your server to download the files. If GitHub is not reachable from your server, you may try setting up a proxy based on your environment to download the files from GitHub. For details, please refer Oracle Container Runtime for Docker User's Guide.

DOWNLOAD ORACLE SOFTWARE

The Oracle Docker repository files do not contain any Oracle Software binaries. Download the software from the Oracle Technology Network and stage the software under dockerfiles/<version> folder. For details, please refer to "Pre-requisites for RAC on Docker" section in the README.md of the OracleRealApplicationClusters Docker files.

Step 5: Build the Docker Image

Once you have downloaded the Oracle RAC on Docker repository files and the Oracle binaries onto your server, build the images based on your environment. You may need to build the following three images, depending on your infrastructure setup.

1. Oracle Connection Manager Image

   – Required if you want to access the Oracle RAC database but you do not have RAC IP addresses that are reachable from physical network.

2. Oracle RAC Storage Server Image

   – Required if you do not have block devices or NAS devices.

   – The OracleRACStorageServer is only for testing purpose and not a recommended way to store Oracle RAC files.

3. Oracle RAC Image

   – Required to run an Oracle RAC DB.

   – To understand the Oracle RAC on Docker provisioning steps in detail, please refer README.MD on GitHub.

Step 6: Configure Docker Host Environment

After you have built the required images, configure the Docker host environment to run Oracle RAC on Docker. You may need to configure kernel parameters, network setup, and ASM device allocation, as well as configuring real-time process settings in the container.

To understand the pre-setup steps for Oracle RAC on Docker, refer to the section “Pre-requisites for Oracle RAC on Docker” in the README.MD on GitHub.

Step 7: Deploy Oracle RAC on DOCKER

Once you have built the images and pre-setup the Docker host environment for the Oracle RAC on Docker deployment, create the following containers as applicable from the images.

1. Oracle Connection Manager Image

   – To understand the Oracle Connection Manager deployment steps in detail, refer to the “Creating the Docker GI and RAC Container” section in the README.MD on GitHub.

2. Oracle RAC Storage Server Image

   – To understand the RAC Storage Server deployment steps in detail, please refer section “Running RACStorageServer Docker container” in README.MD on GitHub.

3. Oracle RAC Image

   – To understand the Oracle RAC on Docker deployment steps in detail, refer to the README.MD on GitHub.

At this point, you have deployed a **single node RAC database running in a single container**. For test & dev environments a single node Oracle RAC environment may be sufficient. However, if you wish to run a multi-instance Oracle RAC database, you must add additional nodes to your Oracle RAC cluster. To add additional nodes, please follow step 8. Otherwise, skip to step 9.

Step 8: Run the ADDNode Script

Oracle RAC provides the addnode.sh script to scale horizontally to meet your workload requirements. If you need to extend the cluster to accommodate the workload on your server, add more nodes to your Oracle RAC cluster. Node addition on Oracle RAC on Docker is automated with parameters passed as environment variables.

Make sure that you have executed pre-setup steps as mentioned in Step 6 in this white paper. For detailed steps to add a node, please refer the “Adding a RAC Node using a Docker container” section in the README.MD.

Step 9: Connect To RAC

Once the Oracle RAC database environment is up and running in the Docker container, your application can connect to the Oracle RAC database. If you have chosen to use Oracle connection manager and exposed the port 1521 on the Docker host to your public network, then you can connect from a client outside the host using “docker host name” and port 1521. If you used the Docker Macvlan configuration, you can connect directly to RAC containers using the SCAN-name.

For more details, please refer to the “Connecting to RAC Database” section in RAC on Docker README.MD on GitHub.

RAC WITH WEBLOGIC ON DOCKER CONTAINER

Oracle WebLogic (Oracle WLS) is a scalable, enterprise-ready J2EE-based application server for developing and deploying multi-tier distributed enterprise applications. WebLogic server provides enterprise-level security and administration tools for ease of managing the applications.

Tighter integration between WebLogic and Oracle RAC database provides a strong, complete, highly available infrastructure to develop and test applications with improved availability, better resource sharing, ease of configuration, and automated management facilities. As WebLogic and Oracle RAC databases are now running in containers, developers and enterprises can take advantage of Docker’s benefits of container isolation, portability, ability to automate development, and testing of these applications. To understand the deployment scenarios of WebLogic on Docker, please refer to the Oracle WebLogic Server on Docker Containers white paper published by WebLogic on OTN.

MedRec (Avitek Medical Records Application) is an end-to-end sample Java EE application shipped with WebLogic Server; it simulates an independent, centralized medical record management system. The MedRec application provides a framework for patients, doctors, and administrators to manage patient data using a variety of different clients. MedRec demonstrates WebLogic server and Java EE features, and it highlights Oracle-recommended best practices. MedRec is installed in the WebLogic server distribution. It has been used as a reliable and realistic application performance benchmark for over 10 years.

Figure 6 shows the system architecture for WebLogic deployment with Oracle RAC on Docker. This illustrates a WebLogic container running the MedRec application, which is connected to RAC Database. If any instance goes down, the application can still connect to other instances.

A WebLogic domain can be extended to a clustered domain by running multiple domains in different containers, which can be managed by the Admin server. Clustered WebLogic containers can run on same host or on multiple hosts.

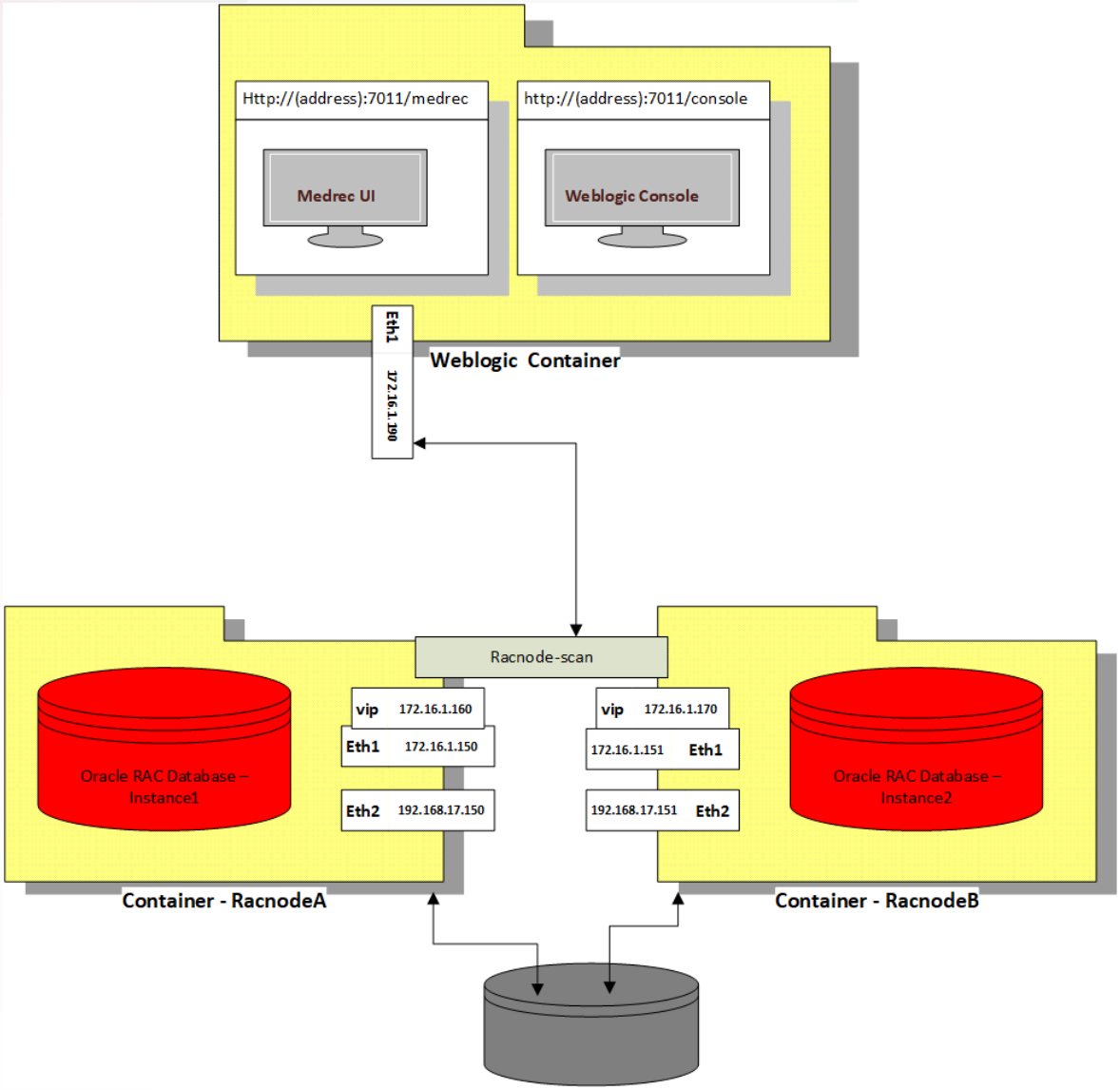

Figure 6 RAC with WebLogic Running on Docker/Container

DEPLOYMENT STEPS: ORACLE RAC WITH WEBLOGIC ON DOCKER

Step 1: Install and Configure Oracle RAC on Docker

To install WebLogic on Docker, first install Oracle RAC on Docker.

Step 2: Install and Configure WebLogic

Download Oracle WebLogic images from GitHub if not already downloaded. Change directory to <DOCKER_IMAGES>/OracleWebLogic/dockerfiles/12.2.1

Step 3: Download the Oracle Weblogic Server Software

Download Oracle WebLogic Server 12.2.1.2 quick installer and stage it under <DOCKER_IMAGES>/OracleWebLogic/dockerfiles/12.2.1. Refer to the README.md to get the exact details of the software along with a download link.

Step 4: Build the Oracle WebLogic Image

Build the Oracle WebLogic developer image by running the following command:

```
# docker build –t oracle/weblogic:12.2.1.2-developer .
```

Step 5: Create the Medrec Image

Change he directory to <DOCKER_IMAGES/OracleWebLogic/samples/12212-oradb-medrec and edit container-scripts/oradatasource.properties. Modify parameters based on your environment as needed. If you have created a RAC database with the default settings, set the following parameters:

```
dsurl=jdbc:oracle:thin:@//172.16.1.70:1521/ORCLCDB

dsusername=system

dspassword=Oracle_12c
```

Step 6: Download WebLogic Supplemental Installer

Download Oracle WebLogic 12.2.1.2.0 supplemental quick disk installer and stage it under the current directory.
For more details, please refer to the README.MD placed under the current directory.

Step 7: Build the Image

Once the software is staged, build the image. Execute the following command to build the image:

```
# docker build –t 12212-oradb-medrec .
```

For more details, please refer to the README.MD placed in the current directory.

Step 8: Run the Container

Once the image is built, execute the following command to create and run the container:

```
docker run –d –p 7011:7011 –network rac_pub1_nw 12212-oradb-medrec
```

Note: Pass –network docker parameter and create the WebLogic container on a public network that you have used for RAC.

Step 9: Access the Console

You can access WebLogic console from a browser, by using the following URL.

```
http://<Docker_Host>:7011/medrec
```

CONCLUSION

In traditional deployment workflows various steps are required, which contribute to an overall deployment pain felt by various teams. Each step added to the deployment process for an application or database, such as applying a patch, providing application scripts, or setting up environment for users, increases the overall, inhering risk that comes with deploying dev/test or production systems. Docker provides a simple tool set that is aimed to combine all steps in a single workflow to address these concerns.

Oracle RAC represents an enterprise-class database product, which is best used with the latest patches and releases. Combined with Docker, it can provide fast provisioning and ready-made images using those latest patches and releases. These images can easily be consumed in dev and test environments and discarded when the dev or test jobs are done, which can expedite the dev and test cycle in any organization.

Concluding, one of the greatest challenges that today's IT faces is the efficient management of test, dev, QA and production environments necessary to support the complete application development lifecycle. Economic realities dictate that these environments be hosted on consolidated systems to maintain cost efficiency. With the introduction of Oracle RAC and Docker repository files on GitHub, customers now have a simple solution to all these challenges. It provides simple and rapid deployment, storage and portability of environments as well as the isolation required for consolidated environments.

ORACLE CORPORATION

Worldwide Headquarters
500 Oracle Parkway, Redwood Shores, CA 94065 USA

Worldwide Inquiries
TELE + 1.650.506.7000 + 1.800.ORACLE1
FAX + 1.650.506.7200
oracle.com

CONNECT WITH US

Call +1.800.ORACLE1 or visit oracle.com. Outside North America, find your local office at oracle.com/contact.

 blogs.oracle.com/oracle  facebook.com/oracle  twitter.com/oracle

Integrated Cloud Applications & Platform Services

Best Practices for Deploying RAC on Docker
January 2019

Author: Sanjay Singh, Paramdeep Saini, Racpack team, Bob Thome and Markus Michalewicz,
and Contributing Authors: Avi Miller, Gerald Venzl and Monica Riccelli

 Oracle is committed to developing practices and products that help protect the environment