O-DB-DOCKER

Lab and Exercise Guide

2019 November 10, Version 0.8

Trivadis AG
Sägereistrasse 29
8152 Glattbrugg

info@trivadis.com
+41 58 459 55 55

Table of Contents

1 Preface

1.1 About O-DB-DOCKER

Oracle has long supported the use of Docker to install its products, including the latest versions of the Oracle database.

In theory, a simple “docker run” instantiates a corresponding container from a docker image. But why isn’t the database container ready in a few seconds?? Where does Oracle Database Image come from and what happens if the container is stopped again? The functional scope as well as the size of the Oracle database container presuppose that one or the other thoughts about the use and operation are made in advance. This includes topics such as data persistence, licensing and other operational aspects.

With a focus on the current versions of Oracle Database and Oracle Unified Directory the following topics will be discussed:

- Docker overview with focus on images, containers and volumes.
- Structure of Oracle Docker Images
- Operating an Oracle Docker Database Container
- Various use cases for Oracle database docker containers such as Oracle Enterprise, User Security, Test and Engineering databases, Migration and more.
- Other topics such as high availability, backup & recovery and licensing.

Within the scope of this training, basics and extended know-how in the area of Oracle database and Docker will be elaborated. The theory will be accompanied by demos and practical exercises.

1.2 Disclaim

This guide has been created with utmost care, but does not claim to be complete. It was compiled as part of the preparation for the *O-DB-DOCKER* workshop. The author assumes no responsibility for the accuracy, completeness and timeliness of the content. The use of the available content is at your own risk.

1.3 Document information

- Document: O-DB-DOCKER Workshop
- Classification: Restricted / Trivadis customer
- Status: Published
- Last changes: 2019.11.11
- Document name: O-DB-DOCKER_Workshop.pdf

Lead Authors	Contributors & Reviewers
Stefan Oehrli	Martin Berger

1.4 Revision History

Version	Date	Visa	Comment
0.1	2019.10.12	soe	Initial release O-DB-DOCKER workshop
0.2	2019.10.17		Add Lab requirements
0.3 - 0.8	2019.11.10		Add initial version of Lab exercises
1.0			First official release of O-DB-DOCKER workshop

If you have any questions, please do not hesitate to contact us via stefan.oehrli@trivadis.com.

2 Workshop Introduction

2.1 Requirements Workshop Environment

During the workshop *O-DB-DOCKER Oracle-Databases in Docker-Containers* there will be the possibility to explore different topics with practical examples. Each participant will be provided with a compute node in the Oracle Cloud for the duration of the *O-DB-DOCKER* training. Alternatively, participants can perform the exercises in their own local VM or Docker environment. Participants are free to choose which environment they want to use for the workshop. Although the setup of the local VM respectively local Docker environment is not part of the workshop itself. The following summary gives a short overview of the different requirements for the three workshop environments.

Detailed information on the workshop environment, documents, instructions etc. are available prior to the training via DOAG2019 O-DB-DOCKER.

2.2 Skills and Knowledge

The different exercises of the workshop allow a step-by-step introduction to the topic *Oracle databases in Docker containers*. The following knowledge of the participants is recommended:

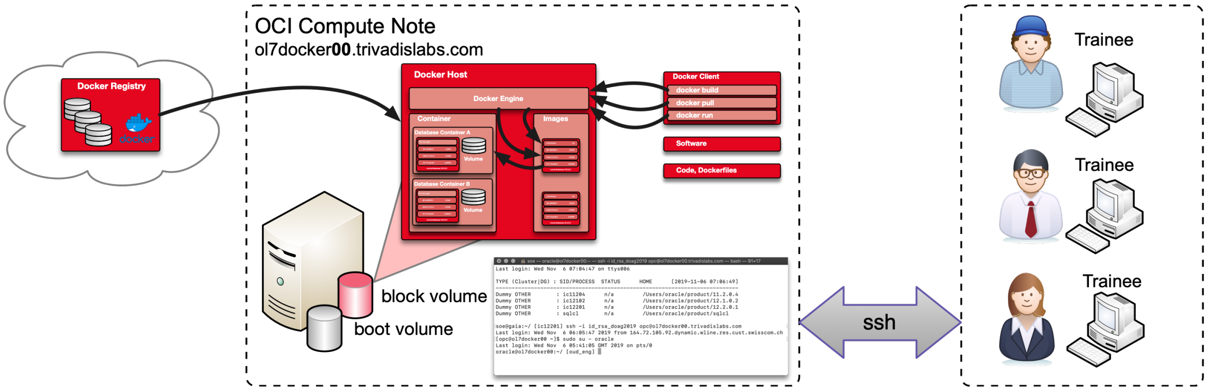
- Oracle database basics like installation, configuration and basic database administration
- Docker basics (see also Get Started, Part 1: Orientation and setup)
- Practical experience with shell scripts, SSH and the command line.

2.3 Compute Node in the Oracle Cloud

The Compute Node in the Oracle Cloud will be specially prepared for this workshop and will be available for practical work for the duration of the training. Each compute node is configured as follows:

- *Host name:* ol7dockerXX.trivadislabs.com (See host overview provided by the teacher.)
- *Internal IP address:* 10.0.0.2
- *External IP address:* See host overview provided by the teacher.
- *VM shape:* VM.Standard2.2
    - *CPU:* 2.0 GHz Intel® Xeon® Platinum 8167M (2 Cores)
    - *Memory:* 30GB
    - *Disk:* ca 256GB
- *Software:*
    - Oracle Enterprise Linux 7.7
    - Docker Engine / Community Edition
    - Predefined Docker Images
    - Miscellaneous Oracle binaries and Git client

The following figure gives an overview of the OCI compute instance and access via SSH. Detailed installation and configuration guide can be found in Appendix A.



```
Last login: Wed Nov  6 07:04:47 on ttys006

TYPE (Cluster|DG) : SID/PROCESS  STATUS      HOME      [2019-11-06 07:06:49]
------------------------------------------------------------------------------
Dummy OTHER       : ic11204      n/a        /Users/oracle/product/11.2.0.4
Dummy OTHER       : ic12102      n/a        /Users/oracle/product/12.1.0.2
Dummy OTHER       : ic12201      n/a        /Users/oracle/product/12.2.0.1
Dummy OTHER       : sqlcl        n/a        /Users/oracle/product/sqlcl

soe@gaia:~/ [ic12201] ssh -i id_rsa_doag2019 opc@ol7docker00.trivadislabs.com
Last login: Wed Nov  6 06:05:47 2019 from 164.72.105.92.dynamic.wline.res.cust.swisscom.ch
[opc@ol7docker00 ~]$ sudo su - oracle
Last login: Wed Nov  6 05:41:05 GMT 2019 on pts/0
oracle@ol7docker00:~/ [oud_eng]
```

Figure. 1: OCI Compute Instance Architecture

Access to the compute nodes is exclusively via SSH and Private Keys. Workshop participants must ensure that they meet the following requirements:

- *SSH client* for remote access, e.g. Putty, MobaXterm or similar.
- *SCP Client* to copy files remotely, e.g. WinSCP, Putty or similar.

- *Text editor* for customizing / developing docker files, scripts etc. e.g. MS Visual Studio Code, UltraEdit, Notepad++ or similar
- It must also be ensured that access to a public IP address or host name is possible via an SSH key.

The following *optional* points are recommended:

- GitHub account to access and download the source code. Simple download does not require an account.

2.4 Local Vagrant VM

As with compute nodes, all exercises can be performed directly in a Local VM. Appropriate vagrant scripts for building a VM are available in the Git Repository oehrlis/o-db-docker. The following requirements must be met in order to set up this VM with Vagrant:

- Virtualbox
- Vagrant
- Local clone of the Git repository oehrlis/o-db-docker
- Oracle Binaries for Oracle 19c and current RU.
- Sufficient hard disk space for the VM and the Docker Images approx. 50GB
- If necessary, additional tools to access and work with the VM, e.g. SSH client, text editor, etc.

Setting up a local VM is not part of the workshop. Participants who wish to work with a VM must configure it in advance. Detailed installation and configuration guide can be found in Appendix B.

2.5 Local Docker Environment

As a third option, the exercises can also be performed in a local docker environment. This is especially useful for working on Linux or MacOS notebooks. In order to perform the workshop locally, the following requirements must be met:

- Installing the Docker Community Edition. See also About Docker - Community
- Local clone of the Git repository oehrlis/o-db-docker and oracle/docker-images)
- Oracle Binaries for Oracle 19c and current RU.
- Sufficient hard disk space for the VM and the Docker Images approx. 50GB
- If necessary, additional tools to access and work with the VM, e.g. SSH client, text editor, git client etc.

Building a local Docker environment is not part of the workshop. Participants who wish to work with a local Docker installation must configure it in advance. detailed installation and configuration guide can be found in Appendix C.

3 Workshop Exercises

3.1 Overview of Exercises

The exercises are grouped in individual directories and can generally be performed independently of each other. Each exercise contains at least a short description of the tasks (`1x??en-Exercise.md`) as well as a solution (`1x??en-Solution.md`). In addition, scripts, docker files, etc. are part of every exercise. For exercises where software (Oracle Binaries, Patch) or code from other Git repositories is required, we refer to it accordingly.

ID	Topic	Description	Duration
01	Get known the Environment	Log in and get to know the workshop environment.	10
02	Simple docker image	Build simple Docker images.	10
03	Docker Image Size	Build different version of an Oracle Instant Client Docker images.	15
04	Local Software Repository	Setup a local Software Repository	15
05	Simples DB Docker Image	Build simple DB images. Identify challenges.	20
06	Oracle Docker Image	Build a Docker DB images using the Oracle build scripts.	20
07	OraDBA Docker Image	Build a Docker DB images using the OraDBA build scripts.	20
08	Simple DB Docker Container	Run a simple DB container.	30
09	Accessing database container	Access and work with the DB container.	10
10	Oracle Unified Audit Setup	Setup a DB Container with Unified Audit and do some tests.	15
11	Oracle RU with datapatch	Patch a Docker DB Container using datapatch.	30
12	Oracle Enterprise User Security	Setup Oracle Enterprise User Security with OUD.	30
13	Oracle PDB	Plugin an Oracle PDB.	20
14	Oracle RAC with Docker	Setup and Build a RAC Docker environment.	30
15	Container Monitoring	Setup and Configure container monitoring.	30
16	Additional Exercises	Ideas for additional exercises.	n/a

3.2 Access Workshop Hosts

The workshop hosts are running as a compute instance in the Oracle Cloud. Each system is accessible via public IP address or its hostname. you have to use a SSH client of your choice for access (e.g. Putty, MochaXterm, SSH etc.)

- Start a *Putty* session from command line. Replace NN with the number of you host.

```
putty -ssh opc@ol7dockerNN.trivadislabs.com -i keys/
   id_rsa_ol7dockerNN.ppk
```

- Start a *SSH* session from command line

```
ssh opc@ol7dockerNN.trivadislabs.com -i keys/id_rsa_ol7dockerNN
```

The instructor supplements the following table with the relevant information for the O-DB-DOCKER course. A compute instance will be assigned to each participant.

ID	Hostname	IP Address	User	Key's	Comment
00	ol7docker00.trivadislabs.com	n/a	opc	public, OpenSSH, Putty	Trainer
01	ol7docker01.trivadislabs.com	n/a	opc	public, OpenSSH, Putty	
02	ol7docker02.trivadislabs.com	n/a	opc	public, OpenSSH, Putty	
03	ol7docker03.trivadislabs.com	n/a	opc	public, OpenSSH, Putty	
04	ol7docker04.trivadislabs.com	n/a	opc	public, OpenSSH, Putty	
05	ol7docker05.trivadislabs.com	n/a	opc	public, OpenSSH, Putty	
06	ol7docker06.trivadislabs.com	n/a	opc	public, OpenSSH, Putty	
07	ol7docker07.trivadislabs.com	n/a	opc	public, OpenSSH, Putty	
08	ol7docker08.trivadislabs.com	n/a	opc	public, OpenSSH, Putty	
09	ol7docker09.trivadislabs.com	n/a	opc	public, OpenSSH, Putty	
10	ol7docker10.trivadislabs.com	n/a	opc	public, OpenSSH, Putty	

3.3 Exercise 1: Get known the Environment

3.3.1 Exercise Goals

Simple exercise to get to know the environment.

- Everybody can access his/her compute instance.

3.3.2 Tasks

- Login to your individual OCI compute instance eg. *ol7dockerXX*.
- Switch to user *oracle*.
- Check what Docker images are available.
- Check directories for workshop and software.

3.4 Solution 1: Get known the Environment

The following steps are performed in this exercise:

- login via SSH client as user opc to the individual OCI compute instance eg. *ol7dockerXX*.
- Switch to user *oracle*
- Run *docker images* to see which images are available
- Check the different directories.

3.4.1 Detailed Solution

The following steps have been performed on the *ol7docker00* host. If necessary, adjust the commands, filenames or the host name according to your environment.

- Start a Putty session from command line.

```
putty -ssh opc@ol7docker00.trivadislabs.com -i keys/
   id_rsa_ol7docker00.ppk
```

- Alternatively start a SSH session from command line

```
ssh opc@ol7docker00.trivadislabs.com -i id_rsa_ol7docker00
```

- Switch to user *oracle*
- Run *docker images* to see which images are available
- Check the different directories.

3.5 Exercise 2: Simple docker image

3.5.1 Exercise Goals

Create a simple Docker image to get warm.

- Everybody can successfully build a docker image.
- Everybody can successfully run a docker container.

3.5.2 Tasks

- Create an Dockerfile with a simple *hello world*
- Build Docker image
- Run the Docker *hello world* example.

3.6 Solution 2: Simple docker image

The following steps are performed in this exercise:

- Create an Dockerfile with a simple *hello world*
- Build Docker image
- Run the Docker *hello world* example.

3.7 Exercise 3: Docker Image Size

3.7.1 Exercise Goals

Identify the reason for different image size for the same Image.

- Everybody know why its image gets large.

3.7.2 Tasks

Build Oracle Instant Client Docker images with different Dockerfiles.

- Dockerfile with several `RUN` commands.
- Dockerfile with a combined `RUN` command.
- Using a bultistage build.
- Verify if Docker expertimental is enabled
- Build an image using squash
- Analyse the different imags using `docker history`

3.8 Solution 3: Docker Image Size

- Dockerfile with several `RUN` commands.
- Dockerfile with a combined `RUN` command.
- Using a bultistage build.
- Verify if Docker expertimental is enabled
- Build an image using squash
- Analyse the different imags using `docker history`

3.9 Exercise 4: Local Software Repository

3.9.1 Exercise Goals

Be able to use a local software repository during build.

3.9.2 Tasks

- Create docker-compose file for the local software repository
- Start the container using `docker-compose`
- Use the local software repository in a Docker build.

3.10 Solution 4: Local Software Repository

The following steps are performed in this exercise:

- login via SSH client as user opc to the individual OCI compute instance eg. *ol7dockerXX*.
- Switch to user *oracle*
- Run *docker images* to see which images are available
- Check the different directories.

3.10.1 Detailed Solution

Es muss folgendes gemacht werden

- Sicherstellen des Zugriffs auf die Docker Übungs- und Entwicklungsumgebung

3.11 Exercise 5: Simples DB Docker Image

3.11.1 Exercise Goals

- Create a simple database Docker image
- Identify challenges

3.11.2 Tasks

- Customize the Dockerfile for build a simple database image.
- Build the database Docker image.
- Think about what this image lacks.

3.12 Solution 5: Simples DB Docker Image

The following steps are performed in this exercise:

- login via SSH client as user opc to the individual OCI compute instance eg. *ol7dockerXX*.
- Switch to user *oracle*
- Run *docker images* to see which images are available
- Check the different directories.

3.12.1 Detailed Solution

The following steps have been performed on the *ol7docker00* host. If necessary, adjust the commands, filenames or the host name according to your environment.

- Start a Putty session from command line.

```
putty -ssh opc@ol7docker00.trivadislabs.com -i keys/
   id_rsa_ol7docker00.ppk
```

- Alternatively start a SSH session from command line

```
ssh opc@ol7docker00.trivadislabs.com -i id_rsa_ol7docker00
```

- Switch to user *oracle*
- Run *docker images* to see which images are available
- Check the different directories.

3.13 Exercise 6: Oracle Docker Image

3.13.1 Exercise Goals

- Create database Docker image using the Oracle build scripts.

3.13.2 Tasks

- Update local git working copy of *oracle/docker-images*
- Check whether all the prerequisites have been fulfilled.
- Build Docker image using `buildDockerImage.sh` or manually using `docker build`

3.14 Solution 6: Oracle Docker Image

The following steps are performed in this exercise:

- login via SSH client as user opc to the individual OCI compute instance eg. *ol7dockerXX*.
- Switch to user *oracle*
- Run *docker images* to see which images are available
- Check the different directories.

3.14.1 Detailed Solution

The following steps have been performed on the *ol7docker00* host. If necessary, adjust the commands, filenames or the host name according to your environment.

- Start a Putty session from command line.

```
putty -ssh opc@ol7docker00.trivadislabs.com -i keys/
   id_rsa_ol7docker00.ppk
```

- Alternatively start a SSH session from command line

```
ssh opc@ol7docker00.trivadislabs.com -i id_rsa_ol7docker00
```

- Switch to user *oracle*
- Run *docker images* to see which images are available
- Check the different directories.

3.15 Exercise 7: OraDBA Docker Image

3.15.1 Exercise Goals

- Create database Docker image using the OraDBA build scripts.

3.15.2 Tasks

- Update local git working copy of *oehrlis/docker*
- Review the Dockerfile and define Oracle version to build
- Check whether all the prerequisites have been fulfilled.
- Build Docker image using `docker build`

3.16 Solution 7: OraDBA Docker Image

The following steps are performed in this exercise:

- login via SSH client as user opc to the individual OCI compute instance eg. *ol7dockerXX*.
- Switch to user *oracle*
- Run *docker images* to see which images are available
- Check the different directories.

3.16.1 Detailed Solution

The following steps have been performed on the *ol7docker00* host. If necessary, adjust the commands, filenames or the host name according to your environment.

- Start a Putty session from command line.

```
putty -ssh opc@ol7docker00.trivadislabs.com -i keys/
   id_rsa_ol7docker00.ppk
```

- Alternatively start a SSH session from command line

```
ssh opc@ol7docker00.trivadislabs.com -i id_rsa_ol7docker00
```

- Switch to user *oracle*
- Run *docker images* to see which images are available
- Check the different directories.

3.17 Exercise 8: Simple DB Docker Container

3.17.1 Exercise Goals

- Run an Oracle database in a Docker container.
- See the difference between docker `docker run` and `docker-compose`
- See what happens if you forget to specify a volume.

3.17.2 Tasks

- Create a Docker container by using `docker run` without a volume.
- Create a Docker container by using `docker run` with a volume.
- Create a Docker container using `docker-compose` with a predefined volume.

3.18 Solution 8: Simple DB Docker Container

The following steps are performed in this exercise:

- login via SSH client as user opc to the individual OCI compute instance eg. *ol7dockerXX*.
- Switch to user *oracle*
- Run *docker images* to see which images are available
- Check the different directories.

3.18.1 Detailed Solution

The following steps have been performed on the *ol7docker00* host. If necessary, adjust the commands, filenames or the host name according to your environment.

- Start a Putty session from command line.

```
putty -ssh opc@ol7docker00.trivadislabs.com -i keys/
   id_rsa_ol7docker00.ppk
```

- Alternatively start a SSH session from command line

```
ssh opc@ol7docker00.trivadislabs.com -i id_rsa_ol7docker00
```

- Switch to user *oracle*
- Run *docker images* to see which images are available
- Check the different directories.

3.19 Exercise 9: Accessing database container

3.19.1 Exercise Goals

Be able to access and use the Docker database container

3.19.2 Tasks

- Access Docker container via shell
- Access Docker container via SQLPlus / SQL Developer

3.20 Solution 9: Accessing database container

The following steps are performed in this exercise:

- login via SSH client as user opc to the individual OCI compute instance eg. *ol7dockerXX*.
- Switch to user *oracle*
- Run *docker images* to see which images are available
- Check the different directories.

3.20.1 Detailed Solution

The following steps have been performed on the *ol7docker00* host. If necessary, adjust the commands, filenames or the host name according to your environment.

- Start a Putty session from command line.

```
putty -ssh opc@ol7docker00.trivadislabs.com -i keys/
   id_rsa_ol7docker00.ppk
```

- Alternatively start a SSH session from command line

```
ssh opc@ol7docker00.trivadislabs.com -i id_rsa_ol7docker00
```

- Switch to user *oracle*
- Run *docker images* to see which images are available
- Check the different directories.

3.21 Exercise 10: Oracle Unified Audit Setup

3.21.1 Exercise Goals

Use startup / setup script folder to customize the database Docker container.

3.21.2 Tasks

- Review the setup scripts
- Refine a `docker-compose.yml` file
- Create a Docker container using `docker-compose` and review the database configuration

3.22 Solution 10: Oracle Unified Audit Setup

The following steps are performed in this exercise:

- login via SSH client as user *opc* to the individual OCI compute instance eg. *ol7dockerXX*.
- Switch to user *oracle*
- Run *docker images* to see which images are available
- Check the different directories.

3.22.1 Detailed Solution

The following steps have been performed on the *ol7docker00* host. If necessary, adjust the commands, filenames or the host name according to your environment.

- Start a Putty session from command line.

```
putty -ssh opc@ol7docker00.trivadislabs.com -i keys/
   id_rsa_ol7docker00.ppk
```

- Alternatively start a SSH session from command line

```
ssh opc@ol7docker00.trivadislabs.com -i id_rsa_ol7docker00
```

- Switch to user *oracle*
- Run *docker images* to see which images are available
- Check the different directories.

3.23 Exercise 11: Oracle RU with datapatch

3.23.1 Exercise Goals

See how to install a Oracle RU in a dockerized database.

3.23.2 Tasks

- Review scripts defined in setup / startup folder.
- Create a container based on 19.4.0.0 (Volume / database is pre-configured)
- Stop and remove the container.
- Update the `docker-compose.yml` file to match the new Docker image.
- Create a container based on 19.5.0.0.
- Review the database configuration and patch status.
- Optional: switch back to old Docker image.

3.24 Solution 11: Oracle RU with datapatch

The following steps are performed in this exercise:

- Review the configuration in `docker-compose.yml` and the scripts defined in setup / startup folder.
- Create / start a container based on 19.4.0.0
- Stop and remove the container.
- Update the `docker-compose.yml` file to match the new Docker image.
- Create a container based on 19.5.0.0.
- Review the database configuration and patch status.
- Optional: switch back to old Docker image.

3.24.1 Background Information

This example shows how to install an Oracle release update (RU) on an Oracle database in a Docker Container. The persistent data (e.g. data files, config files etc.) is stored on an external volume. This allows to stop /remove the container and create a new one based on a Docker image with additional RU's, patch etc. The startup script `00_run_datapatch.sh` will run Oracle *datapatch* to apply / rollback the patch in the database. Some prerequisites and basic principles:

- `00_run_datapatch.sh` does check if you database has java installed. If so, it will restart the database in upgrade mode to run *datapatch*.
- If database is a container database the PDB's will be open to run datapatch.
- This use case does only run within an Oracle major release eg. 19.x.0.0 or 18.x.0.0 but not as an method to upgrade from 18c to 19c.
- It is relevant it you have a basic container with an RU or any kind of one-off patch.

The following figure illustrates the patch process of an Oracle database container.

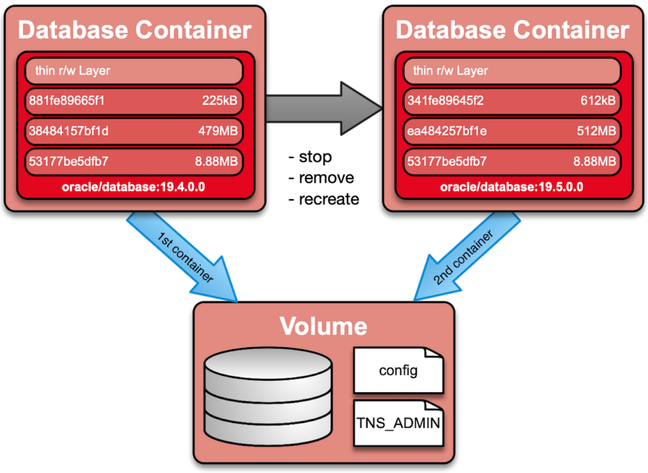


Figure 1: Patch Database Container

3.24.2 Run the Patch Update

Update the `docker-compose.yml` file and set the desired base image. Default is 19.4.0.0.

```
vi docker-compose.yml
```

Create a container tcpu01 using `docker-compose`. This will also create an initial database *TCPU01*.

```
docker-compose up -d
```

Monitor the progress of database creation using `docker-compose`.

```
docker-compose logs -f
```

The database is ready when you see the following message in your docker logs.

```
---------------------------------------------------------------
 - DATABASE TDB194S IS READY TO USE!
---------------------------------------------------------------
```

You now can shutdown and destroy the container using `docker-compose`. Database will remain since it is stored on a voluem / bind-mount folder.

```
docker-compose down
```

Update the `docker-compose.yml` file and set the desired base image. e.g 19.5.0.0.

```
vi docker-compose.yml
```

Re-create the container tcpu01 using `docker-compose`. The database *TCPU01* will be reused. The run script `50_run_database.sh` will make sure, that the scripts in the startup folder are executed. This includes `00_run_datapatch.sh`.

```
docker-compose up -d
```

Monitor the progress of database startup / datapach using `docker-compose`.

```
docker-compose logs -f
```

You may also run the use case to downgrade / remove an existing RU.

3.24.3 Customization

By default the volume will be created in the directory specified by the environment variable *DOCKER_VOLUME_BASE*. If the environment variable is not specified, it will use the default value from `*.env` which is the current path. Beside the usual changes e.g. container name, hostname, ports etc. you can configure how the DB itself will be created by specify several configuration parameter.

Feature (Bundle)	User Status	Description
CONNECTIONS	ALL DISABLE	Allow all database options
CTX_LOGGING	ALL DISABLE	Use logging in Oracle Text PL/SQL procedures as CTX_OUTPUT.START_LOG and CTX_OUTPUT.START_QUERY_LOG

Environment Variable	Values	Comment
ORACLE_SID	`TCPU01`	Default Oracle SID. Usually it will default to the variable which has been specified during build. A custom SID can / should be specified.
ORACLE_PDB	`PDB1`	Default PDB name, if *CONTAINER* is set to `TRUE`
CONTAINER	`FALSE`	Flag to create a container or single tenant database. Default set to false.
ORACLE_PWD	n/a	Custom admin password for common admin user like SYS and SYSTEM. If not specified a random password will be generated
INSTANCE_INIT	`/u01/config`	Folder for customize setup and startup. The database create script will look for a folder `setup` during initial setup or `startup` during each container startup. All bash `.sh` scripts as well sql `.sql` script will be executed. Make sure to add a sequence to keep the order of the scripts. In this use case we will set the *INSTANCE_INIT* to `/u01/config` which is mapped to the local config folder.
ORADBA_TEMPLATE_PREFIX	n/a	Prefix to use a custom dbca template or the general purpose default template. By default this variable is not set. In this case dbca will use the general purpose template with the starter database. If set to `custom_` dbca will use a custom template to create a fresh database. This will take longer since the database will be create from scratch.

Environment variable	Comment
ORACLE_SID `TCPU01`	Default Oracle SID. Usually it will default to the variable which has been specified during build. A custom SID can / should be specified.

Environment variable		Comment
ORACLE_PDB	PDB1	Default PDB name, if *CONTAINER* is set to `TRUE`
CONTAINER	FALSE	Flag to create a container or single tenant database. Default set to false.
ORACLE_PWD	n/a	Custom admin password for common admin user like SYS and SYSTEM. If not specified a random password will be generated
INSTANCE_INIT	`${ORACLE_BASE}/admin/${ORACLE_SID}/scripts`	Folder for customize setup and startup. The database create script will look for a folder `setup` during initial setup or `startup` during each container startup. All bash `.sh` scripts as well sql `.sql` script will be executed. Make sure to add a sequence to keep the order of the scripts. In this use case we will set the *INSTANCE_INIT* to `/u01/config` which is mapped to the local config folder.
ORADBA_TEMPLATE_PREFIX	n/a	Prefix to use a custom dbca template or the general purpose default template. By default this variable is not set. In this case dbca will use the general purpose template with the starter database. If set to `custom_` dbca will use a custom template to create a fresh database. This will take longer since the database will be create from scratch.
DEFAULT_DOMAIN	`domainname`	Database default domain. If not specified the default domain will be used.
TNS_ADMIN	`${ORACLE_BASE}/network/admin`	Alternative TNS_ADMIN environment variable.

3.25 Exercise 12: Oracle Enterprise User Security

3.25.1 Exercise Goals

- See how a more complex test environment can be setup using `docker-compose`
- Test Oracle Enterprise User Security.

3.25.2 Tasks

- Review the `docker-compose.yml`
- Start the container using `docker-compose`
- Test Oracle Names Resolution within the database container.
- Test EUS Login within the database container.

3.26 Solution 12: Oracle Enterprise User Security

The following steps are performed in this exercise:

- login via SSH client as user *opc* to the individual OCI compute instance eg. *ol7dockerXX*.
- Switch to user *oracle*
- Run *docker images* to see which images are available
- Check the different directories.

3.26.1 Detailed Solution

The following steps have been performed on the *ol7docker00* host. If necessary, adjust the commands, filenames or the host name according to your environment.

- Start a Putty session from command line.

```
putty -ssh opc@ol7docker00.trivadislabs.com -i keys/
   id_rsa_ol7docker00.ppk
```

- Alternatively start a SSH session from command line

```
ssh opc@ol7docker00.trivadislabs.com -i id_rsa_ol7docker00
```

- Switch to user *oracle*
- Run *docker images* to see which images are available
- Check the different directories.

3.27 Exercise 13: Oracle PDB

3.27.1 Exercise Goals

Plugin an existing PDB into a new container.

3.27.2 Tasks

- Create the `docker-compose.yml` for a Oracle Container Database

- Prepare the scripts to look for the PDB.
- Start the container and plugin the PDB.
- Verify the status of the PDB.

3.28 Solution 13: Oracle PDB

The following steps are performed in this exercise:

- login via SSH client as user opc to the individual OCI compute instance eg. *ol7dockerXX*.
- Switch to user *oracle*
- Run *docker images* to see which images are available
- Check the different directories.

3.28.1 Detailed Solution

The following steps have been performed on the *ol7docker00* host. If necessary, adjust the commands, filenames or the host name according to your environment.

- Start a Putty session from command line.

```
putty -ssh opc@ol7docker00.trivadislabs.com -i keys/
   id_rsa_ol7docker00.ppk
```

- Alternatively start a SSH session from command line

```
ssh opc@ol7docker00.trivadislabs.com -i id_rsa_ol7docker00
```

- Switch to user *oracle*
- Run *docker images* to see which images are available
- Check the different directories.

3.29 Exercise 14: Oracle RAC with Docker

3.29.1 Exercise Goals

Setup a simple RAC based test environment on Docker

3.29.2 Tasks

- Create the Docker network.
- Create a RAC storage server Docker container.
- Create a NFS volume
- Creating the Docker GI and RAC Container

- Assign networks to RAC containers
- Adding a RAC Node using a Docker container
- Connecting to RAC Database

3.30 Solution 14: Oracle RAC with Docker

The following steps are performed in this exercise:

- login via SSH client as user opc to the individual OCI compute instance eg. *ol7dockerXX*.
- Switch to user *oracle*
- Run *docker images* to see which images are available
- Check the different directories.

3.30.1 Detailed Solution

The following steps have been performed on the *ol7docker00* host. If necessary, adjust the commands, filenames or the host name according to your environment.

- Start a Putty session from command line.

```
putty -ssh opc@ol7docker00.trivadislabs.com -i keys/
   id_rsa_ol7docker00.ppk
```

- Alternatively start a SSH session from command line

```
ssh opc@ol7docker00.trivadislabs.com -i id_rsa_ol7docker00
```

- Switch to user *oracle*
- Run *docker images* to see which images are available
- Check the different directories.

3.31 Exercise 15: Container Monitoring

3.31.1 Exercise Goals

Test the different Monitoring solutions for Oracle databases on Docker.

3.31.2 Tasks

- configure and review HEALTHCHECK
- Setup and configure Prometheus
- Setup and configure Grafana.

3.32 Solution 15: Container Monitoring

The following steps are performed in this exercise:

- login via SSH client as user opc to the individual OCI compute instance eg. *ol7dockerXX*.
- Switch to user *oracle*
- Run *docker images* to see which images are available
- Check the different directories.

3.32.1 Detailed Solution

The following steps have been performed on the *ol7docker00* host. If necessary, adjust the commands, filenames or the host name according to your environment.

- Start a Putty session from command line.

```
putty -ssh opc@ol7docker00.trivadislabs.com -i keys/
   id_rsa_ol7docker00.ppk
```

- Alternatively start a SSH session from command line

```
ssh opc@ol7docker00.trivadislabs.com -i id_rsa_ol7docker00
```

- Switch to user *oracle*
- Run *docker images* to see which images are available
- Check the different directories.

3.33 Additional Exercises

A couple of ideas for additional exercises:

- Oracle Data Guard
- Oracle Database Backup and Recovery
- Oracle APEX test DB

4 Appendix A Setup OCI Environment

4.1 Requirements and Preparations

The following appendix contains a step-by-step guide on how to set up the Workshop VM in the Oracle Cloud. The different steps are performed via command line and scripts. Alternatively the OCI configurations can be done via OCI Web Console. For the sake of simplicity, we will limit this guide to the OCI command line utility.

The following prerequisites must be fulfilled:

- Oracle Cloud Infrastructure subscription and access to the OCI console see Oracle Cloud Infrastructure Documentation
- Corresponding subscription level to create different resources in OCI (Cloud Credits, up todate billing, etc.) The Oracle Cloud Free Tier does work for basic configuration. But the free OCI compute does not have enough resources to run Oracle Database Docker containers.
- Appropriate Compartment to create the different OCI resources.
- Local installation of the OCI command line tool see OCI Command Line Interface (CLI)

For the workshop *O-DB-DOCKER* we use a separate compartment, which is also named *O-DB-DOCKER*. If you follow this guide, you either have to create a compartment with an identical name or adapt the commands accordingly. The creation of a compartment is not discussed here in detail. The workshop itself is setup in Oracle Cloud region *Germany Central (Frankfurt)* identified as *eu-frankfurt-1*.

Disclaim: This guide has been created with utmost care, but does not claim to be complete. It was compiled as part of the preparation for the *O-DB-DOCKER* workshop. The author assumes no responsibility for the accuracy, completeness and timeliness of the content. The use of the available content is at your own risk.

4.1.1 Install OCI CLI

The installation of the OCI CLI has to be done according to the OCI documentation. Under MacOS and Linux this is fairly straightforward.

- Download and install OCI. You will have to specify an installation location, update *PATH* etc.

```
bash -c "$(curl -L https://raw.githubusercontent.com/oracle/oci-
   cli/master/scripts/install/install.sh)"
```

- Configure oci using the following command. You will be asked for your tenancy OCID, user OCID as well region and configure a SSH key.

```
oci setup config
```

4.1.2 Create a Compartment

Create a compartment for *O-DB-DOCKER* withing the compartment *Compartment_trivadislabs*.

- Get the parent compartment id. For O-DB-DOCKER we search for a parent compartment containing *trivadislabs*.

```
PARENT_COMP_ID=$(oci search resource free-text-search \
--text "trivadislabs" \
--raw-output \
--query "data.items[*].identifier|[0]")
```

- Create the new compartment for *O-DB-DOCKER*.

```
oci iam compartment create \
--compartment-id $PARENT_COMP_ID \
--description "DOAG 2019 Training Day Oracle DB in Docker
   containers" \
--name "O-DB-DOCKER"
```

Create a Virtual Cloud Network (VCN) within the new compartment *O-DB-DOCKER*.

- Get the compartment id as variable *COMPARTMENT_OCID*.

```
COMPARTMENT_OCID=$(oci iam compartment list \
--compartment-id-in-subtree true --all \
--raw-output --query "data [?name == 'O-DB-DOCKER'].id|[0]")
```

- create a VCN *vcn-o-db-docker*

```
oci network vcn create \
--compartment-id $COMPARTMENT_OCID \
--cidr-block "10.0.0.0/16" \
--display-name "vcn-o-db-docker" \
--dns-label "vcnodbdocker"
```

4.2 Compute Instance

In the following section we will reference to some names eg. host name, compartment name etc. To simplify the upcoming commands we define a couple of environment variables.

```
export HOST_NAME="ol7docker00"
export COMPARTMENT_NAME="O-DB-DOCKER"
```

4.2.1 Create Compute Instance

To create a compute instance we first have to get a few OCID's for different components.

- Get the compartment id as variable *COMPARTMENT_OCID*.

```
COMPARTMENT_OCID=$(oci iam compartment list \
--compartment-id-in-subtree true --all \
--raw-output --query "data [?name == '${COMPARTMENT_NAME}'].id
   |[0]")
```

- Get the *ocid* of the Oracle Linux image as variable *IMAGE_OCID*.

```
IMAGE_OCID=$(oci compute image list --compartment-id
   $COMPARTMENT_OCID \
--operating-system-version "7.7" \
--operating-system "Oracle Linux" \
--sort-by TIMECREATED \
--raw-output --query "data [*].id|[0]")
```

- Get the *ocid* of the virtual cloud network (VCN) *vcn-o-db-docker* as variable *VCN_OCID*.

```
VCN_OCID=$(oci network vcn list --compartment-id
   $COMPARTMENT_OCID \
--raw-output \
--query "data [?contains(\"display-name\",'o-db-docker')].id|[0]
   ")
```

- Get the *ocid* of the subnet as variable *SUBNET_OCID*.

```
SUBNET_OCID=$(oci network subnet list \
--compartment-id $COMPARTMENT_OCID \
--vcn-id $VCN_OCID \
--sort-by DISPLAYNAME --raw-output \
--query "data [*].id|[0]")
```

- Get the *availability domain* of the subnet as variable *AV_DOAMIN*.

```
AV_DOAMIN=$(oci network subnet list \
--compartment-id $COMPARTMENT_OCID \
--vcn-id $VCN_OCID \
--sort-by DISPLAYNAME --raw-output \
--query "data [*].\"availability-domain\"|[0]")
```

- Create a SSH key pair for the compute instance

```
cd o-db-docker/lab/oci/
ssh-keygen -b 4096 -C "DOAG 2019 Training" -f id_rsa_doag2019
```

Finally lets create the OCI compute instance using the following specification:

- ol7docker01
- VM.Standard2.2
- *COMPARTMENT_OCID*
- *AV_DOAMIN*
- *IMAGE_OCID*
- *SUBNET_OCID*
- SSH public key from file

- check the variables

```
echo "COMPARTMENT_OCID   = $COMPARTMENT_OCID" && \
echo "HOST_NAME          = $HOST_NAME" && \
echo "AV_DOAMIN          = $AV_DOAMIN" && \
echo "IMAGE_OCID         = $IMAGE_OCID" && \
echo "SUBNET_OCID        = $SUBNET_OCID" && \
echo "SSH Key            = $(cat id_rsa_doag2019.pub)"
```

- create the compute instance

```
oci compute instance launch --compartment-id $COMPARTMENT_OCID \
--availability-domain $AV_DOAMIN \
--display-name $HOST_NAME \
--image-id $IMAGE_OCID \
--subnet-id $SUBNET_OCID \
--shape VM.Standard2.2 \
--assign-public-ip true \
--metadata "{\"ssh_authorized_keys\": \"$(cat id_rsa_doag2019.
   pub)\"}"
```

- check the provisioning status

```
oci compute instance list --compartment-id $COMPARTMENT_OCID \
--output table \
--query "data [?contains(\"display-name\",$HOST_NAME)].{\"
   display-name\":\"display-name\", id:id,\"lifecycle-state\":\"
   lifecycle-state\"}"


oci compute instance list --compartment-id $COMPARTMENT_OCID \
--output table \
--query "data [?contains(\"display-name\",'$HOST_NAME')].{\"
   display-name\":\"display-name\",\"lifecycle-state\":\"
   lifecycle-state\"}"
+--------------+-----------------+
| display-name | lifecycle-state |
+--------------+-----------------+
| ol7docker00  | TERMINATED      |
| ol7docker00  | RUNNING         |
+--------------+-----------------+
```

- get the compute instance ID as variable *INSTANCE_OCID*.

```
INSTANCE_OCID=$(oci compute instance list \
--compartment-id $COMPARTMENT_OCID  \
--lifecycle-state 'RUNNING' \
--raw-output --query "data [?contains(\"display-name\",'
   $HOST_NAME')].id|[0]")
```

4.2.2 Create Block Storage

Create a new block volume to use with the compute instance o-db-docker. We use the variable *COMPARTMENT_OCID* and *AV_DOAMIN* from the previous section. First lets check the variables.

```
echo "COMPARTMENT_OCID  = $COMPARTMENT_OCID" && \
echo "AV_DOAMIN         = $AV_DOAMIN"
```

Create the block volume in the compartment *O-DB-DOCKER*.

```
oci bv volume create --compartment-id $COMPARTMENT_OCID \
--availability-domain $AV_DOAMIN \
--display-name "o-db-docker_bv00" \
--size-in-gbs 512
```

Get the block volume OCID as variable *VOLUME_OCID*.

```
VOLUME_OCID=$(oci bv volume list \
--compartment-id $COMPARTMENT_OCID \
--raw-output --query "data [?contains(\"display-name\",'o-db-
   docker_bv00')].id|[0]")
```

Assign block volume to compute instance *ol7docker00*.

```
oci compute volume-attachment attach \
--instance-id $INSTANCE_OCID \
--type "iscsi" \
--volume-id $VOLUME_OCID \
--device "/dev/oracleoci/oraclevdb"
```

Get the volume IP and volume IQN for the iscsiadm utility later on.

```
VOLUME_ATTACH_ID=$(oci compute volume-attachment list \
--compartment-id $COMPARTMENT_OCID --raw-output \
--query "data [?\"volume-id\" == '$VOLUME_OCID'].id|[0]")
VOLUME_IP=$(oci compute volume-attachment get \
--volume-attachment-id $VOLUME_ATTACH_ID \
--raw-output --query "data.ipv4")
VOLUME_IQN=$(oci compute volume-attachment get \
--volume-attachment-id $VOLUME_ATTACH_ID \
--raw-output --query "data.iqn")
```

Log into the compute instance ol7docker00 and attache the iscsi device

```
ssh opc@ol7docker00.trivadislabs.com -C "sudo iscsiadm -m node -
   o new -T $VOLUME_IQN -p $VOLUME_IP:3260"
ssh opc@ol7docker00.trivadislabs.com -C "sudo iscsiadm -m node -
   o update -T $VOLUME_IQN -n node.startup -v automatic"
ssh opc@ol7docker00.trivadislabs.com -C "sudo iscsiadm -m node -
   T $VOLUME_IQN -p $VOLUME_IP:3260 -l"
```

4.2.3 DNS Configuration

Optionally we add the IP of the compute instance to the DNS Zone *trivadislabs.com*.

- Get DNS OCID as variable *DNS_OCID*.

```
DNS_OCID=$(oci dns record zone get \
--zone-name-or-id "trivadislabs.com" \
--query "etag" --raw-output|sed 's/.*\(ocid.*\)#.*/\1/')
```

- Get IP address of compute instance as variable *IP_ADDRESS*.

```
IP_ADDRESS=$(oci compute instance list-vnics \
--instance-id $INSTANCE_OCID --raw-output \
--query "data[].\"public-ip\"|[0]")
```

- Add DNS record u

```
oci dns record domain update \
--domain "$HOST_NAME.trivadislabs.com" \
--zone-name-or-id "trivadislabs.com" \
--force \
--items "[{\"domain\": '$HOST_NAME.trivadislabs.com',\"
   isProtected\": true,\"rdata\": \"$IP_ADDRESS\",\"recordHash\"
   : null,\"rrsetVersion\": \"4\",\"rtype\": \"A\",\"ttl\": 30}]
   "
```

- remove DNS record

```
oci dns record domain delete \
--zone-name-or-id "trivadislabs.com" \
--domain "$HOST_NAME.trivadislabs.com" --force
```

4.3 Setup OS

4.3.1 Disk Partition

```
sfdisk /dev/sdb <<EOF
0,25000
,,8e
EOF
```

```
[root@ol7docker00 ~]# lsblk
NAME    MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sdb       8:16   0   512G  0 disk├─
sdb2    8:18    0 320.5G  0 part└─
sdb1    8:17    0 191.5G  0 part
sda       8:0    0  46.6G  0 disk├─
sda2    8:2     0      8G  0 part [SWAP]├─
sda3    8:3     0  38.4G  0 part /└─
```

```
sda1   8:1    0  200M  0 part /boot/efi
[root@ol7docker00 ~]# fdisk -l /dev/sdb

Disk /dev/sdb: 549.8 GB, 549755813888 bytes, 1073741824 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 1048576 bytes
Disk label type: dos
Disk identifier: 0x00000000

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               1   401624999   200812499+  83  Linux
Partition 1 does not start on physical sector boundary.
/dev/sdb2       401625000  1073736404   336055702+  8e  Linux
   LVM
```

- Create a physical volume on `/dev/sdb2` using `pvcreate`.

```
[root@ol7docker00 ~]$ pvcreate /dev/sdb2
  Physical volume "/dev/sdb2" successfully created.

[root@ol7docker00 ~]$ pvs
  PV         VG Fmt  Attr PSize    PFree
  /dev/sdb2     lvm2 ---  <320.49g <320.49g

[root@ol7docker00 ~]$ pvdisplay /dev/sdb2
  "/dev/sdb2" is a new physical volume of "<320.49 GiB"
  --- NEW Physical volume ---
  PV Name               /dev/sdb2
  VG Name
  PV Size               <320.49 GiB
  Allocatable           NO
  PE Size               0
  Total PE              0
  Free PE               0
  Allocated PE          0
  PV UUID               ONMpyt-j3RW-D4EQ-xpum-l3tR-fL7H-jgAseL
```

- Create volume group vgora using `vgcreate`

```
[root@ol7docker00 ~]$ vgcreate vgora /dev/sdb2
  Volume group "vgora" successfully created

[root@ol7docker00 ~]$ vgdisplay vgora
  --- Volume group ---
  VG Name               vgora
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  1
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                1
  Act PV                1
  VG Size               320.48 GiB
  PE Size               4.00 MiB
  Total PE              82044
  Alloc PE / Size       0 / 0
  Free  PE / Size       82044 / 320.48 GiB
  VG UUID               qDKwXo-M8ad-L0eL-SLld-Mbd9-v83g-bOA6e3
```

- Create a locial volumes

```
[root@ol7docker00 ~]$ lvcreate -n vol_u00 -L 100G vgora
  Logical volume "vol_u00" created.
[root@ol7docker00 ~]$ lvcreate -n vol_u01 -L 100G vgora
  Logical volume "vol_u01" created.

[root@ol7docker00 ~]$ lvdisplay
  --- Logical volume ---
  LV Path                /dev/vgora/vol_u00
  LV Name                vol_u00
  VG Name                vgora
  LV UUID                BJ3T5W-xzgy-jpwr-u1vS-Jl3y-tHKJ-bAnXED
  LV Write Access        read/write
  LV Creation host, time ol7docker00, 2019-10-23 15:14:52 +0000
  LV Status              available
  # open                 0
  LV Size                100.00 GiB
  Current LE             25600
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           252:0

  --- Logical volume ---
  LV Path                /dev/vgora/vol_u01
  LV Name                vol_u01
  VG Name                vgora
  LV UUID                c2zepi-PnVA-6Cs1-EQgB-yJmz-Wtm5-Q6eL4b
  LV Write Access        read/write
  LV Creation host, time ol7docker00, 2019-10-23 15:15:06 +0000
  LV Status              available
  # open                 0
  LV Size                100.00 GiB
  Current LE             25600
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           252:1
```

Create the filesystem on the new volumes

```
[root@ol7docker00 ~]# mkfs.ext4 /dev/vgora/vol_u00
```

```
mke2fs 1.42.9 (28-Dec-2013)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=256 blocks
6553600 inodes, 26214400 blocks
1310720 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=2174746624
800 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736,
       1605632, 2654208,
    4096000, 7962624, 11239424, 20480000, 23887872

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

[root@ol7docker00 ~]# mkfs.ext4 /dev/vgora/vol_u01
mke2fs 1.42.9 (28-Dec-2013)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=256 blocks
6553600 inodes, 26214400 blocks
1310720 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=2174746624
800 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736,
       1605632, 2654208,
    4096000, 7962624, 11239424, 20480000, 23887872

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
```

- get the block device information

```
[root@ol7docker00 ~]# blkid /dev/vgora/vol_u00 /dev/vgora/
   vol_u01
/dev/vgora/vol_u00: UUID="2d8a938f-5960-4664-9091-eb9bd6132f91"
   TYPE="ext4"
/dev/vgora/vol_u01: UUID="2c74d466-221e-49d0-a644-8e1e299cabf4"
   TYPE="ext4"
```

- create mount points

```
mkdir -p /u00 /u01
```

- update fstab and mount the filesystems

```
echo "$(blkid /dev/vgora/vol_u00|cut -d' ' -f2|tr -d '"')    /u00
       ext4    defaults,noatime,_netdev      0    0" >>/etc/fstab
echo "$(blkid /dev/vgora/vol_u01|cut -d' ' -f2|tr -d '"')    /u01
       ext4    defaults,noatime,_netdev      0    0" >>/etc/fstab

mount /u00
mount /u01
```

4.3.2 Setup ORAbase_init Scripts

Download the oradba_init script

```
DOWNLOAD="/tmp/download"
SETUP_INIT="00_setup_oradba_init.sh"
GITHUB_URL="https://github.com/oehrlis/oradba_init/raw/master/
   bin"
```

```
mkdir -p ${DOWNLOAD}
curl -Lsf ${GITHUB_URL}/${SETUP_INIT} -o ${DOWNLOAD}/${
   SETUP_INIT}
```

```
chmod 755 ${DOWNLOAD}/${SETUP_INIT}
${DOWNLOAD}/${SETUP_INIT}
```

sudo -u oracle /opt/oradba/bin/20_setup_oudbase.sh

echo “oud_eng:1389:1636:4444:8989:OUD:Y” »${ETC_BASE}/oudtab oracle@ol7docker00:~/ [oud_eng] . oudenv.sh

sed -i “s|. ${OUD_BASE}/bin/oudenv.sh|. ${OUD_BASE}/bin/oudenv.sh SILENT|” $HOME/.bash_profile

mkdir -p /u00/app/oracle/software

sudo usermod -a -G oinstall opc # chmod 775 /u00/app/ [root@ol7docker00 u00]# chmod 775 /u00/app/oracle/ [root@ol7docker00 u00]# chmod 775 /u00/app/oracle/software/ ### Setup OS Oracle DB

nohup /opt/oradba/bin/01_setup_os_db.sh > /tmp/01_setup_os_db.sh 2>&1 & nohup /opt/oradba/bin/01_setup_os_docker.sh > /tmp/01_setup_os_docker.log 2>&1 & yum install git systemctl stop docker rm -rf /var/lib/docker docker-storage-config -s btrfs -d /dev/sdb1 systemctl start docker systemctl enable docker

git clone https://github.com/oehrlis/docker.git docker git clone https://github.com/oehrlis/o-db-docker.git o-db-docker git clone https://github.com/oracle/docker-images.git docker-images

Generate download url file from the `*.download` files which are part of the oradba/docker repository.

- change to the oci working directory and remove the old download url files.

```
cd $cdl/o-db-docker/lab/oci

rm download*.url
```

- build a new download url file

```
find $cdl/docker -name *.download \
-exec grep -i "Direct Download" {} \;| \
sed "s/# Direct Download    : //" |\
grep -iv '^#'|grep -iv 'n/a'|sort -u>> download.url
```

- Separate OTN from MOS downloads

```
grep -i "download.oracle.com" download.url >download_otn.url
grep -iv "download.oracle.com" download.url >download_mos.url
```

Start to download the patch from MOS using `curl`.

- Temporary create a `.netrc` file with MOS credentials. Replace *MOS_USER* and *MOS_PASSWORD* with corresponding values.

```
cd $cdl/o-db-docker/lab/oci
echo 'machine login.oracle.com login MOS_USER password
   MOS_PASSWORD' >.netrc
```

- Download the files from MOS using `download_mos.url`.

```
cd $cdl/o-db-docker/lab/oci
sw="/u00/app/oracle/software"
for url in $(cat download_mos.url); do
  file=$(echo $url| cut -d= -f3)
  log=$(basename $file .zip).log
  echo "Initiate download job for file : $file"
  nohup curl --netrc-file .netrc --cookie-jar cookie-jar.txt \
    --location-trusted "${url}" -o ${sw}/${file} > ${sw}/$log
       2>&1 &
done
```

- Wait until all curl background jobs are done:

```
ps -ef|grep curl
ps -ef|grep curl|wc -l
```

4.4 Configure Environment

- Disk partition

- Docker Volume
- Docker images
- Git stuff
- Download Oracle Binaries

4.4.1 Create Custom Image

Stop the compute instance

- define my variables

```
export HOST_NAME="ol7docker00"
export COMPARTMENT_NAME="O-DB-DOCKER"
```

- Get the compartment id as variable *COMPARTMENT_OCID*.

```
COMPARTMENT_OCID=$(oci iam compartment list \
--compartment-id-in-subtree true --all \
--raw-output --query "data [?name == '${COMPARTMENT_NAME}'].id
   |[0]")
```

- get the compute instance ID as variable *INSTANCE_OCID*.

```
INSTANCE_OCID=$(oci compute instance list \
--compartment-id $COMPARTMENT_OCID  \
--lifecycle-state 'RUNNING' \
--raw-output --query "data [?contains(\"display-name\",'${
   HOST_NAME}')].id|[0]")
```

- stopping compute instance

```
oci compute instance action \
--action SOFTSTOP \
--instance-id ${INSTANCE_OCID}
```

- check if stoppend

```
oci compute instance list --compartment-id $COMPARTMENT_OCID \
--output table \
--query "data [?contains(\"display-name\",'${HOST_NAME}')].{\"
   display-name\":\"display-name\",\"lifecycle-state\":\"
   lifecycle-state\"}"
```

yum install htop Loaded plugins: langpacks, ulninfo ol7_UEKR5 | 2.8 kB 00:00:00 ol7_addons

Create a custom image

```
oci compute image create \
--compartment-id $COMPARTMENT_OCID \
--display-name "${COMPARTMENT_NAME}_master" \
--instance-id ${INSTANCE_OCID}
```

http://www.nazmulhuda.info/download-from-the-otn-using-wget https://blog.pythian.com/how-to-download-oracle-software-using-wget-or-curl/ ## Clone Trainee Instances

4.4.2 Create Compute Instance

To create a compute instance we first have to get a few OCID's for different components.

```
export HOST_NAME="ol7docker01"
export COMPARTMENT_NAME="O-DB-DOCKER"
```

- Get the compartment id as variable *COMPARTMENT_OCID*.

```
COMPARTMENT_OCID=$(oci iam compartment list \
--compartment-id-in-subtree true --all \
--raw-output --query "data [?name == '${COMPARTMENT_NAME}'].id
   |[0]")
```

- Get the *ocid* of the virtual cloud network (VCN) *vcn-o-db-docker* as variable *VCN_OCID*.

```
VCN_OCID=$(oci network vcn list --compartment-id
   $COMPARTMENT_OCID \
--raw-output \
--query "data [?contains(\"display-name\",'o-db-docker')].id|[0]
   ")
```

- Get the *ocid* of the subnet as variable *SUBNET_OCID*.

```
SUBNET_OCID=$(oci network subnet list \
--compartment-id $COMPARTMENT_OCID \
--vcn-id $VCN_OCID \
--sort-by DISPLAYNAME --raw-output \
--query "data [*].id|[0]")
```

- Get the *availability domain* of the subnet as variable *AV_DOAMIN*.

```
AV_DOAMIN=$(oci network subnet list \
--compartment-id $COMPARTMENT_OCID \
--vcn-id $VCN_OCID \
--sort-by DISPLAYNAME --raw-output \
--query "data [*].\"availability-domain\"|[0]")
```

- Get the *ocid* of the Oracle Linux image as variable *IMAGE_OCID*.

```
IMAGE_OCID=$(oci compute image list --compartment-id
   $COMPARTMENT_OCID \
--display-name "O-DB-DOCKER_master_v00" \
--raw-output --query "data [*].id|[0]")
```

- Create a SSH key pair for the compute instance

```
cd o-db-docker/lab/oci/
ssh-keygen -b 4096 -C "DOAG 2019 Training ${HOST_NAME}" -f
   id_rsa_${HOST_NAME} -N ""
```

- check the variables

```
echo "COMPARTMENT_OCID  = $COMPARTMENT_OCID" && \
echo "HOST_NAME         = $HOST_NAME" && \
echo "AV_DOAMIN         = $AV_DOAMIN" && \
echo "IMAGE_OCID        = $IMAGE_OCID" && \
echo "SUBNET_OCID       = $SUBNET_OCID" && \
echo "SSH Key           = $(cat id_rsa_${HOST_NAME})"
```

- create the compute instance

```
oci compute instance launch --compartment-id $COMPARTMENT_OCID \
--availability-domain $AV_DOAMIN \
--display-name $HOST_NAME \
--image-id $IMAGE_OCID \
--subnet-id $SUBNET_OCID \
--shape VM.Standard2.2 \
--assign-public-ip true \
--ssh-authorized-keys-file id_rsa_${HOST_NAME}.pub
```

- check the provisioning status

```
oci compute instance list --compartment-id $COMPARTMENT_OCID \
--output table \
--query "data [?contains(\"display-name\",'$HOST_NAME')].{\"
   display-name\":\"display-name\",\"lifecycle-state\":\"
   lifecycle-state\"}"
```

```
oci compute instance list --compartment-id $COMPARTMENT_OCID \
--query "data [?contains(\"display-name\",'$HOST_NAME')].\"
   lifecycle-state\"|[0]" --raw-output
```

- get the compute instance ID as variable *INSTANCE_OCID*.

```
INSTANCE_OCID=$(oci compute instance list \
--compartment-id $COMPARTMENT_OCID  \
--lifecycle-state 'RUNNING' \
--raw-output --query "data [?contains(\"display-name\",'
   $HOST_NAME')].id|[0]")
```

- Get the block volume OCID from the master block volume as variable *VOLUME_OCID*.

```
VOLUME_OCID=$(oci bv volume list \
--compartment-id $COMPARTMENT_OCID \
--raw-output --query "data [?contains(\"display-name\",'o-db-
   docker_bv00_master_v00')].id|[0]")
```

- clone block volume

```
oci bv volume create --compartment-id $COMPARTMENT_OCID \
--availability-domain $AV_DOAMIN \
--display-name "o-db-docker_bv00_$HOST_NAME" \
--source-volume-id $VOLUME_OCID
```

- clone block volume

```
oci bv volume create --compartment-id $COMPARTMENT_OCID \
--availability-domain $AV_DOAMIN \
--display-name "o-db-docker_bv00_$HOST_NAME" \
--source-volume-id $VOLUME_OCID
```

- check the provisioning status

```
oci bv volume list --compartment-id $COMPARTMENT_OCID \
--output table \
--query "data [?contains(\"display-name\",'o-db-
   docker_bv00_$HOST_NAME')].{\"display-name\":\"display-name\"
   ,\"lifecycle-state\":\"lifecycle-state\"}"

oci bv volume list --compartment-id $COMPARTMENT_OCID \
--output table \
--query "data [?contains(\"display-name\",'o-db-
   docker_bv00_$HOST_NAME')].\"lifecycle-state\"|[0]" --raw-
   output
```

- Get the block volume OCID from the master block volume as variable *VOLUME_OCID*.

```
VOLUME_OCID=$(oci bv volume list \
--compartment-id $COMPARTMENT_OCID \
--raw-output --query "data [?contains(\"display-name\",'o-db-
   docker_bv00_$HOST_NAME')].id|[0]")
```

- check the variables

```
echo "INSTANCE_OCID  = $INSTANCE_OCID" && \
echo "VOLUME_OCID    = $VOLUME_OCID"
```

- Assign block volume to compute instance *ol7docker00*.

```
oci compute volume-attachment attach \
--instance-id $INSTANCE_OCID \
--type iscsi \
--volume-id $VOLUME_OCID \
--device "/dev/oracleoci/oraclevdb"
```

- Get the volume IP and volume IQN for the iscsiadm utility later on.

```
VOLUME_ATTACH_ID=$(oci compute volume-attachment list \
--compartment-id $COMPARTMENT_OCID --raw-output \
--query "data [?\"volume-id\" == '$VOLUME_OCID'].id|[0]")
VOLUME_IP=$(oci compute volume-attachment get \
--volume-attachment-id $VOLUME_ATTACH_ID \
--raw-output --query "data.ipv4")
VOLUME_IQN=$(oci compute volume-attachment get \
--volume-attachment-id $VOLUME_ATTACH_ID \
--raw-output --query "data.iqn")
```

Optionally we add the IP of the compute instance to the DNS Zone *trivadislabs.com*.

- Get DNS OCID as variable *DNS_OCID*.

```
DNS_OCID=$(oci dns record zone get \
--zone-name-or-id "trivadislabs.com" \
--query "etag" --raw-output|sed 's/.*\(ocid.*\)#.*/\1/')
```

- Get IP address of compute instance as variable *IP_ADDRESS*.

```
IP_ADDRESS=$(oci compute instance list-vnics \
--instance-id $INSTANCE_OCID --raw-output \
--query "data[].\"public-ip\"|[0]")
```

- Add DNS record u

```
oci dns record domain update \
--domain "$HOST_NAME.trivadislabs.com" \
--zone-name-or-id "trivadislabs.com" \
--force \
--items "[{\"domain\": '$HOST_NAME.trivadislabs.com',\"
   isProtected\": true,\"rdata\": \"$IP_ADDRESS\",\"recordHash\"
   : null,\"rrsetVersion\": \"4\",\"rtype\": \"A\",\"ttl\": 30}]
   "
```

```
[{domain: \"$HOST_NAME.trivadislabs.com\",isProtected: true,
   rdata: \"$IP_ADDRESS\",recordHash: null,rrsetVersion: 4,rtype
   : \"A\",ttl: 30},
  {
    "domain": "string",
    "isProtected": true,
    "rdata": "string",
    "recordHash": "string",
    "rrsetVersion": "string",
    "rtype": "string",
    "ttl": 0
  }
]
echo oci dns record domain update \
--domain "$HOST_NAME.trivadislabs.com" \
--zone-name-or-id "trivadislabs.com" \
--force \
--items "[{domain: \"$HOST_NAME.trivadislabs.com\",isProtected:
   true,rdata: \"$IP_ADDRESS\",recordHash: null,rrsetVersion: 4,
   rtype: \"A\",ttl: 30},{domain: \"$HOST_NAME.trivadislabs.com\
   ",isProtected: true,rdata: \"$IP_ADDRESS\",recordHash: null,
   rrsetVersion: 4,rtype: \"A\",ttl: 30}]"
```

ssh opc@$HOST_NAME.trivadislabs.com

Log into the compute instance ol7docker00 and attache the iscsi device

```
ssh opc@$IP_ADDRESS -C "sudo iscsiadm -m node -o new -T
   $VOLUME_IQN -p $VOLUME_IP:3260"
ssh opc@$IP_ADDRESS -C "sudo iscsiadm -m node -o update -T
   $VOLUME_IQN -n node.startup -v automatic"
ssh opc@$IP_ADDRESS -C "sudo iscsiadm -m node -T $VOLUME_IQN -p
   $VOLUME_IP:3260 -l"
```

5 Appendix B Setup Vagrant Environment

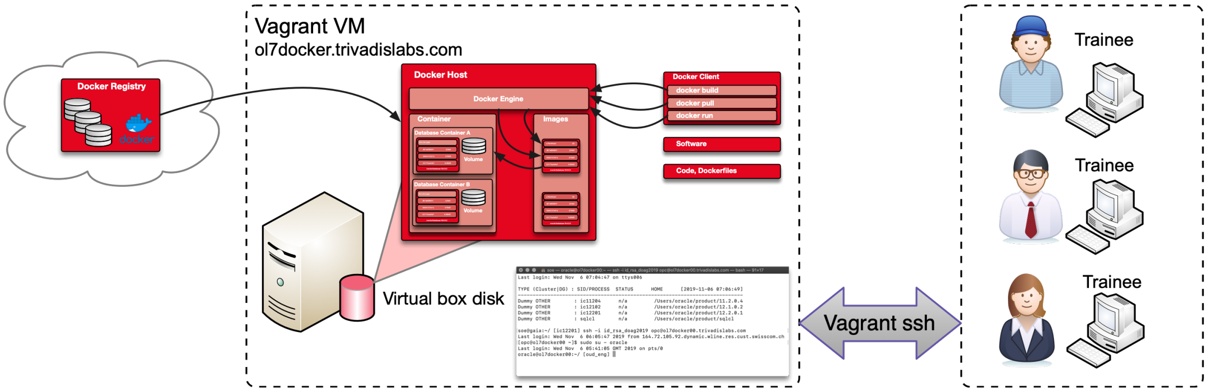
5.1 Requirements and Preparations

The following appendix contains a step-by-step guide on how to set up the workshop environment with local Virtualbox installation. The different steps are performed via command line and scripts.

The following prerequisites must be fulfilled:

- Oracle Virtual Box https://www.virtualbox.org/
- Oracle Vagrant Box https://yum.oracle.com/boxes
- Vagrant https://www.vagrantup.com/
- optional local git client.
- And binaries required to build the Docker images e.g. Oracle database softwar

The following figure shows the schematic layout of the Docker environment based on Vagrant / Virtualbox. The provided vagrant environment does only have a minimal disk setup. It may be necessary to add an additional hard disk or extend the existing one.



```
Last login: Wed Nov  6 07:04:47 on ttys006

TYPE (Cluster|DG) : SID/PROCESS  STATUS      HOME      [2019-11-06 07:06:49]
---------------------------------------------------------------------------
Dummy OTHER       : ic11204      n/a         /Users/oracle/product/11.2.0.4
Dummy OTHER       : ic12102      n/a         /Users/oracle/product/12.1.0.2
Dummy OTHER       : ic12201      n/a         /Users/oracle/product/12.2.0.1
Dummy OTHER       : sqlcl        n/a         /Users/oracle/product/sqlcl

soe@gaia:~/ [ic12201] ssh -i id_rsa_doag2019 opc@ol7docker00.trivadislabs.com
Last login: Wed Nov  6 06:05:47 2019 from 164.72.105.92.dynamic.wline.res.cust.swisscom.ch
[opc@ol7docker00 ~]$ sudo su - oracle
Last login: Wed Nov  6 05:41:05 GMT 2019 on pts/0
oracle@ol7docker00:~/ [oud_eng]
```

Figure. 2: Vagrant VM Workshop Architecture

5.2 Setup Using Vagrant

The installation and configuration of the VM is fairly straightforward using vagrant.

- Add the latest Oracle Linux 7 vagrant box

```
vagrant box add --name ol7-latest https://yum.oracle.com/boxes/
   oraclelinux/latest/ol7-latest.box
```

- Clone the O-DB-DOCKER git Repository to you local working or development directory

```
git clone https://github.com/oehrlis/o-db-docker.git o-db-docker
```

- Change to the LAB Vagrant folder.

```
cd o-db-docker/lab/ol7docker
```

- Install Vagrant Plugin's.

```
vagrant plugin install vagrant-reload
vagrant plugin install vagrant-vbguest
```

- Setup virtualbox using vagrant

```
vagrant up
```

- as soon as the VM provisioning is finished you can login.

```
vagrant ssh
```

- Within the VM you now can switch to user Oracle and run docker commands.

```
sudo su - oracle
docker images
```

The VM is now basically ready to work with Docker. The Git repositories for O-DB-DOCKER as well as oracle/docker-images and oehrlis/docker are available under /xx. Nevertheless there are a few things to consider.

- Oracle binaries and patches must be downloaded separately. Ideally, the software is made available via a shared folder in order not to occupy the disk space in the VM.
- Available disk space is at best too small for many Oracle Docker images. Either create bind mount's on a VM shared folder or increase disk space.
- The O-DB-DOCKER workshop does provide a couple of pre-build docker images, which are not available within a local VM. This is not an issue, since all images, volumes etc can be build. It just takes some time.

5.3 Manual Setup

The O-DB-DOCKER environment can also be installed manually on an existing Oracle VM. The following steps have to be performed.

- Update the OEL 7 OS using `yum upgrade`
- Install the latest `docker-engine` and `docker-compose` via yum.
- Configure a dedicated docker volume / disk
- Configure a user which is used for Docker Engineering.

5.3.1 Manual Setup Using the Vagrant Scripts

The scripts used to setup the Vagrant VM can be used to setup / configure an existing VM. So that not all paths have to be adapted, the scripts are simply copied to a directory /vagrant on the VM.

- Copy all vagrant files from VM host to the VM.

```
scp -r o-db-docker/lab/ol7docker VMNAME:/vagrant
```

- Manual run the vagrant provisioning script to setup the OS (update yum packages, install oracle packages and create an oracle user)

```
cd /vagrant
scripts/01_setup_os_db.sh
```

- Manual install oudbase environment

```
scripts/10_configure_oudbase.sh
```

- Clone git repositories using the script

```
scripts/11_clone_git_repos.sh
```

- install some base images

```
scripts/12_pull_base_images.sh
```

5.3.2 Manual Setup Using the oradba_init Scripts

The vagrant setup scripts are all based on the oradba_init scripts. VM can therefor also configured using these scripts.

- Install the OraDBA init scripts

```
mkdir -p /tmp/download

curl -Lsf https://raw.githubusercontent.com/oehrlis/oradba_init/
   master/bin/00_setup_oradba_init.sh \
    -o /tmp/download/00_setup_oradba_init.sh

chmod 755 /tmp/download/00_setup_oradba_init.sh
/tmp/download/00_setup_oradba_init.sh
rm -rf /opt/oradba/oradba_init-master
```

- Run the OS install script. This script does install oracle preinstall packages and creates an oracle user

```
/opt/oradba/01_setup_os_db.sh
```

- Run the Docker install script. Just install `docker-engine` and `docker-compose`.

```
/opt/oradba/01_setup_os_docker.sh
```

- install git client

```
yum install git -y
```

- Install OUD base a simple Oracle environment similar to Trivadis BasEnv.

```
/opt/oradba/20_setup_oudbase.sh
```

- Clone the git repo either as user root, oracle or any other user you plan to use.

```
export ORACLE_BASE=${ORACLE_BASE:-/u00/app/oracle}
su -l oracle -c "cd ${ORACLE_BASE}/local;git clone https://
   github.com/oehrlis/o-db-docker.git"
su -l oracle -c "cd ${ORACLE_BASE}/local;git clone https://
   github.com/oehrlis/docker.git"
su -l oracle -c "cd ${ORACLE_BASE}/local;git clone https://
   github.com/oracle/docker-images.git"
```

- Pull a couple of docker base images.

```
docker pull busybox
docker pull alpine
docker pull oraclelinux:7-slim
```

Disclaimer: This guide has been created with utmost care, but does not claim to be complete. It was compiled as part of the preparation for the *O-DB-DOCKER* workshop. The author assumes no responsibility for the accuracy, completeness and timeliness of the content. The use of the available content is at your own risk. # Appendix C Setup local Docker Environment

5.4 Requirements and Preparations

The following appendix contains a step-by-step guide on how to set up the workshop environment with local Docker installation. The different steps are performed via command line and scripts.

The following prerequisites must be fulfilled:

- Docker Community Edition https://docs.docker.com/install

Disclaimer: This guide has been created with utmost care, but does not claim to be complete. It was compiled as part of the preparation for the *O-DB-DOCKER* workshop. The author assumes no responsibility for the accuracy, completeness and timeliness of the content. The use of the available content is at your own risk.

To be documented. Will be updated in an upcoming release.

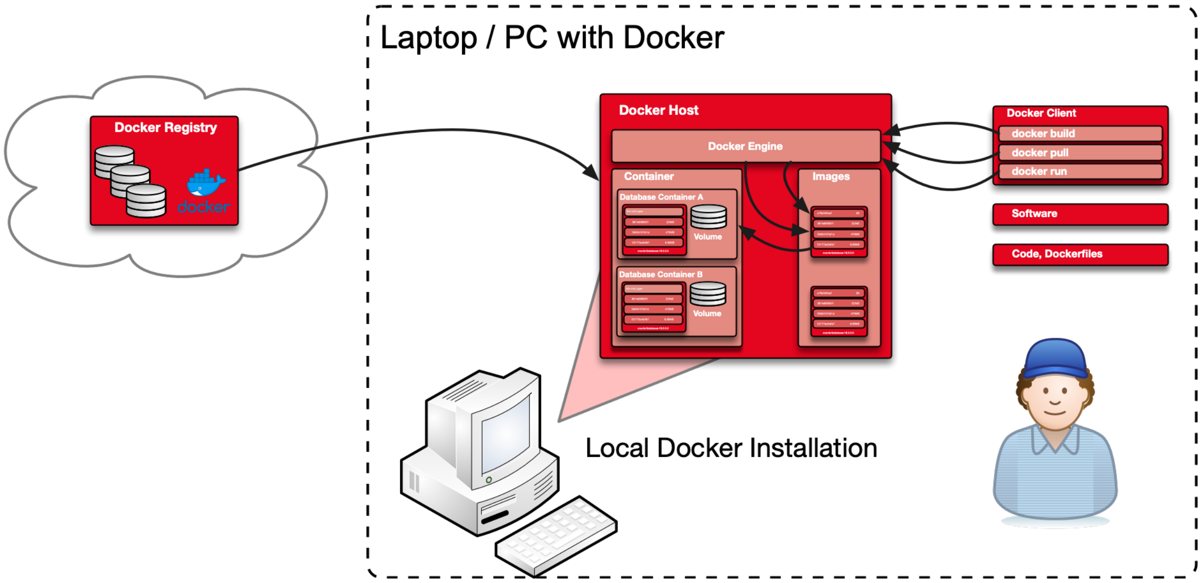


Figure. 3: Local Docker Workshop Architecture