

Hands-on Lab: Build and Deploy Oracle Database and Oracle Weblogic Using Docker 17.02

Version 16

Created by Karsten Schulz Terp-Nielsen based on material from Avi Miller, Gerald Venzl and Monica Riccelli on Jan 15, 2018 10:48 PM.

Learn how to customize a Docker container image and use it to instantiate Oracle Database and Oracle Weblogic instances across different Linux servers. This article describes how to create a Dockerfile, how to allocate runtime resources to containers, and how to establish a communication channel between two containers (between Weblogic and database containers). We will also introduce some best practices for Oracle Database and Oracle Weblogic on Docker.

Introduction

This hands-on lab is designed to help you get a taste for how you can use Docker to simplify application provisioning on Linux leading to how to run The Oracle Database and The Oracle Weblogic Server in Docker. It demonstrates how to create a Docker container on Oracle Linux, modify it, and use it to instantiate multiple application instances. It also describes how to allocate system CPU and memory resources to a Docker container. When the Docker introduction is over the hands-on lab moves into the area of running the Oracle Database EE and Oracle Weblogic in Docker, creating customized Oracle Database and Oracle Weblogic Docker Images and how to utilize external volumes etc.

Getting Started (Optional)

You can download a pre-built VM template for [Oracle VM VirtualBox](#) for this lab: [Oracle Linux VM Images for Hands-On Lab](#). Choose the “Build and Deploy Portable Applications Using Docker” image.

Once you have downloaded and imported the VM template into VirtualBox, log to the VM as `holuser` with the password `oracle`.

To use Docker, the first step is to download and install the Docker Engine RPM packages. Oracle publishes Docker Engine RPMs for Oracle Linux 6 and Oracle Linux 7 on the [Oracle Linux Yum Server](#) and on the [Unbreakable Linux Network \(ULN\)](#).

Up-to-date Instructions for installing the Docker Engine are available in the [Oracle Linux 6 Docker User's Guide](#) or the [Oracle Linux 7 Docker User's Guide](#).

The initial steps are pretty simple: first enable the `ol7_addons` channel in `/etc/yum.repos.d/public-yum-ol*.repo` by using the `yum-config-manager` tool and then run `yum install docker-engine`.

In addition, the hands-on lab image has been pre-configured with a `btrfs`-based filesystem for `/var/lib/docker`. The instructions above detail how to create and format a `btrfs` filesystem. The following commands assume you are logged in as the `holuser` user and have opened a Terminal window:

1. `[holuser@docker ~]$ sudo yum-config-manager --enable ol7_addons`
2. `[holuser@docker ~]$ sudo yum install docker-engine`
3. `[holuser@docker ~]$ sudo usermod -a -G docker holuser`

You'll notice that the last command adds the `holuser` to the `docker` group. This allows the `holuser` to run the Docker tool and manipulate containers. In order for the group membership to take effect, please **log out** and then log back into the graphical user desktop.

Once Docker has been installed, we need to enable and start it using the `systemd` tool:

1. `[holuser@docker ~]$ sudo systemctl status docker`
2. `docker.service - Docker Application Container Engine`
3. `Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; vendor preset: disabled)`
4. `Drop-In: /etc/systemd/system/docker.service.d`
5. `└─docker-sysconfig.conf`
6. `Active: inactive (dead)`
7. `Docs: https://docs.docker.com`

Here we can see that the Docker Engine has not been started automatically after installation. First, we need to enable the Docker Engine so that it is started automatically whenever the system boots:

1. `[holuser@docker ~]$ sudo systemctl enable docker`
2. `Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/docker.service.`

Next, we need to start the Docker Engine and verify that it has started correctly:

1. `[holuser@docker ~]$ sudo systemctl start docker`
2. `[holuser@docker ~]$ sudo systemctl status docker`
3. `docker.service - Docker Application Container Engine`
4. `Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)`
5. `Drop-In: /etc/systemd/system/docker.service.d`
6. `└─docker-sysconfig.conf`
7. `Active: active (running) since Wed 2016-09-14 14:05:57 PDT; 1s ago`
8. `Docs: https://docs.docker.com`
9. `Main PID: 5882 (dockerd)`
10. `Memory: 12.0M`
11. `CGroup: /system.slice/docker.service`
12. `└─5882 dockerd --selinux-enabled`
13. `└─5896 docker-containerd -l unix:///var/run/docker/libcontainerd/docker-containerd.sock --shim docker-containerd-shim --metrics-interval=0 --start-timeout 2m --st...`

```

14.
15. Sep 14 14:05:57 docker.oracleworld.com docker[5882]: time="2016-09-14T14:05:57.282797074-
    07:00" level=info msg="Graph migration to content-addressability took 0.00 seconds"
16. Sep 14 14:05:57 docker.oracleworld.com docker[5882]: time="2016-09-14T14:05:57.283169910-
    07:00" level=warning msg="mountpoint for pids not found"
17. Sep 14 14:05:57 docker.oracleworld.com docker[5882]: time="2016-09-14T14:05:57.283362149-
    07:00" level=info msg="Loading containers: start."
18. Sep 14 14:05:57 docker.oracleworld.com docker[5882]: time="2016-09-14T14:05:57.294207528-
    07:00" level=info msg="Firewalld running: false"
19. Sep 14 14:05:57 docker.oracleworld.com docker[5882]: time="2016-09-14T14:05:57.361256467-
    07:00" level=info msg="Default bridge (docker0) is assigned with an IP add...P address"
20. Sep 14 14:05:57 docker.oracleworld.com docker[5882]: time="2016-09-14T14:05:57.421012757-
    07:00" level=info msg="Loading containers: done."
21. Sep 14 14:05:57 docker.oracleworld.com docker[5882]: time="2016-09-14T14:05:57.421121256-
    07:00" level=info msg="Daemon has completed initialization"
22. Sep 14 14:05:57 docker.oracleworld.com docker[5882]: time="2016-09-14T14:05:57.421121265-
    07:00" level=info msg="Docker daemon" commit=b9264d4 graphdriver=btrfs version=1.12.0
23. Sep 14 14:05:57 docker.oracleworld.com systemd[1]: Started Docker Application Container Engine.
24. Sep 14 14:05:57 docker.oracleworld.com docker[5882]: time="2016-09-14T14:05:57.445533944-
    07:00" level=info msg="API listen on /var/run/docker.sock"
25. Hint: Some lines were ellipsized, use -l to show in full.

```

Now that the Docker Engine has started, we can check the version of both the engine and the Docker client:

```

1. [holuser@docker ~]$ docker version
2. Client:
3.   Version:   1.12.0
4.   API version: 1.24
5.   Go version: go1.6.3
6.   Git commit: b9264d4
7.   Built:
8.   OS/Arch:   linux/amd64
9.
10. Server:
11.  Version:   1.12.0
12.  API version: 1.24
13.  Go version: go1.6.3
14.  Git commit: b9264d4
15.  Built:
16.  OS/Arch:   linux/amd64

```

We can also view information about the currently running engine, including which storage and execution drivers are active:

```

1. [holuser@docker ~]$ docker info
2. Containers: 0
3.   Running: 0
4.   Paused: 0
5.   Stopped: 0

```

```
6. Images: 0
7. Server Version: 1.12.0
8. Storage Driver: btrfs
9. Build Version: Btrfs v3.19.1
10. Library Version: 101
11. Logging Driver: json-file
12. Cgroup Driver: cgroupfs
13. Plugins:
14. Volume: local
15. Network: null overlay host bridge
16. Swarm: inactive
17. Runtimes: runc
18. Default Runtime: runc
19. Security Options: seccomp
20. Kernel Version: 4.1.12-37.5.1.el7uek.x86_64
21. Operating System: Oracle Linux Server 7.2
22. OSType: linux
23. Architecture: x86_64
24. CPUs: 4
25. Total Memory: 7.545 GiB
26. Name: docker.oracleworld.com
27. ID: U5XC:JCMK:KTY3:5NJE:SNU7:GQSV:CX2W:KSCU:P57D:W3FC:TRLQ:BI66
28. Docker Root Dir: /var/lib/docker
29. Debug Mode (client): false
30. Debug Mode (server): false
31. Registry: https://index.docker.io/v1/
32. Insecure Registries:
33. 127.0.0.0/8
```

The Docker Hub Registry is a public cloud store of images hosted by Docker, Inc. that can be used to build running containers. Repositories provide a mechanism for Docker image distribution and sharing. Docker, Inc. also hosts private repositories for a monthly fee. Alternatively, you can create a private Docker Registry behind your own corporate firewall, implementing protection mechanisms, such as SSL encryption and HTTP authentication, to restrict and protect access.

The following commands pull the Oracle Linux 6 and Oracle Linux 7 images from the public Docker Hub Registry, as well as the official MySQL image from Oracle, downloading them to the local machine:

```
1. [holuser@docker ~]$ docker pull oraclelinux
2. Using default tag: latest
3. latest: Pulling from library/oraclelinux
4.
5. e48fc69ad7bb: Pull complete
6. Digest: sha256:4dd37c90e3908fe2327754e45f7cd907691654d0d125ba1d842f2ae7d4de962d
7. Status: Downloaded newer image for oraclelinux:latest
```

```
1. [holuser@docker ~]$ docker pull oraclelinux:6
2. 6: Pulling from library/oraclelinux
3.
```

4. 9df7b97f79e6: Pull complete
5. Digest: sha256:5bbe1185549cf30a77f4e05907c386feb7518952df36aa592a24fef523e4930
6. Status: Downloaded newer image for oraclelinux:6

1. [holuser@docker ~]\$ docker pull mysql/mysql-server
2. Using default tag: latest
3. latest: Pulling from mysql/mysql-server
- 4.
5. a3ed95caeb02: Pull complete
6. 89937cfc6593: Pull complete
7. 17cdbad66360: Pull complete
8. a9c6b92f18d2: Pull complete
9. 35be0cb12451: Pull complete
10. Digest: sha256:a994152904a25d6341e11b2c8f8d28ffe1e600e3898a7466c5186dde5d9fdb42
11. Status: Downloaded newer image for mysql/mysql-server:latest

Images can include a specific tag, which indicates a particular build of that image. In the commands above, you'll notice we pulled `oraclelinux` first, which always points to the latest version of Oracle Linux on the Docker Hub (currently 7.2). The second command specified the specific tag "6", which indicates that we want the latest Oracle Linux 6 build (which is Oracle Linux 6.8). To view all the available tags, visit the [Oracle Linux official image page](#) on the Docker Hub.

During the download process, you'll notice that Docker tells us when a particular filesystem layer already exists. This is because Docker creates the final filesystem for each container by layering changes. This allows Docker to optimize both downloads and cloning operations to only operate on the layers that have changes.

Next, we can view all the available images on our local machine:

1. holuser@docker ~]\$ docker images
2.

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
oraclelinux	6	3956a77fcc51	2 weeks ago	223.1 MB
[oraclelinux	latest	5f5dae795065	2 weeks ago	278.1 MB
mysql/mysql-server	latest	18a962a188ee	5 weeks ago	366.9 MB
3. oraclelinux 6 3956a77fcc51 2 weeks ago 223.1 MB
4. [oraclelinux latest 5f5dae795065 2 weeks ago 278.1 MB
5. mysql/mysql-server latest 18a962a188ee 5 weeks ago 366.9 MB

Customizing a Container for Application Provisioning (Optional)

Suppose that I want to provision multiple, identical web servers across multiple Linux servers in my data center. Docker makes it easy to create a preconfigured, cookie-cutter environment in a container image. We can then use this prebuilt image and deploy it across one or many Linux hosts.

To build a customized container image, we must first build a guest container, install the web server, and configure it to deliver web server content. We can use the `docker run` command to run an Oracle Linux 7 base container and execute a bash shell in the guest container:

1. [holuser@docker ~]\$ docker run -t -i --name guest oraclelinux /bin/bash
2. [root@722c42b3c151 /]#

The Docker Engine assigns an image ID to every running container instance. Because we used the arguments `-i` and `-t`, the bash shell runs interactively, and the prompt reflects the first 12 characters (`f85d55a6893f`) of my running container's image ID. The `--name` argument specifies a name for the running container instance. If you choose not to enter a name, the Docker Engine generates a random string that incorporates the name of a notable scientist, inventor, or developer, such as `evil_jones`; `sad_ritchie`; `sleepy_curie` and `prickly_mestorf`.

Open **another** shell and use the `docker ps` command to show information about the running guest container, including the shortened form of the image ID, the base image used to create this container, and the container name:

```
1. [holuser@docker ~]$ docker ps
2. CONTAINER ID   IMAGE      COMMAND                  CREATED          STATUS          PORTS          N
   AMES
3. 722c42b3c151   oraclelinux "/bin/bash"            About a minute ago Up About a minute          guest
```

Back in the original shell and in the newly-created Docker guest, we install the `httpd` and `perl` RPM packages using `yum`, just as you would on any other physical or virtual server. We also use `yum clean all` to remove cache files that `yum` creates during package installation, because that will ultimately save some space in the exported container image: Root password is 'oracle'.

```
1. [root@722c42b3c151 /]# yum install -y httpd perl && yum clean all
2. Loaded plugins: ulninfo
3. o17_UEKR3 | 1.2 kB 00:00:0
   0
4. o17_latest | 1.4 kB 00:00:00
5. (1/5): o17_UEKR3/x86_64/updateinfo | 64 k
   B 00:00:00
6. (2/5): o17_latest/x86_64/updateinfo | 848 kB
   00:00:00
7. (3/5): o17_latest/x86_64/group | 681 kB 00
   :00:00
8. (4/5): o17_latest/x86_64/primary | 17 MB 0
   0:00:03
9. (5/5): o17_UEKR3/x86_64/primary | 19 M
   B 00:00:05
10. o17_UEKR3 444/44
    4
11. o17_latest 14758/14758

12. Resolving Dependencies
13. --> Running transaction check
14. --> Package httpd.x86_64 0:2.4.6-40.0.1.el7_2.1 will be installed
15. --> Processing Dependency: httpd-tools = 2.4.6-40.0.1.el7_2.1 for package: httpd-2.4.6-40.0.1.el7_2.1.x86_64
16. --> Processing Dependency: /etc/mime.types for package: httpd-2.4.6-40.0.1.el7_2.1.x86_64
17. --> Processing Dependency: libaprutil-1.so.0()(64bit) for package: httpd-2.4.6-40.0.1.el7_2.1.x86_64
18. --> Processing Dependency: libapr-1.so.0()(64bit) for package: httpd-2.4.6-40.0.1.el7_2.1.x86_64
19. ...
20. --> Finished Dependency Resolution
```

```
21.
22. Dependencies Resolved
23.
24. =====
    =====
    =====
25. Package                Arch            Version          Repository
26. Size
    =====
    =====
27. Installing:
28. httpd                  x86_64          2.4.6-
   40.0.1.el7_2.1        ol7_latest      1.2 M
29. perl                   x86_64          4:5.16.3-
   286.el7               ol7_latest      8.0 M
30. ...
31. Transaction Summary
32. =====
    =====
    =====
33. Install 2 Packages (+31 Dependent packages)
34.
35. Total download size: 14 M
36. Installed size: 44 M
37. Downloading packages:
38. (1/33): apr-1.4.8-
   3.el7.x86_64.rpm                                           | 99 kB  00:00:00
39. (2/33): apr-util-1.5.2-
   6.0.1.el7.x86_64.rpm                                       | 91 kB  00:00:00
40. (3/33): groff-base-1.22.2-
   8.el7.x86_64.rpm                                           | 933 kB  00:00:00
41. ...
42. -----
    -----
43. Total                                                         7.0 MB/s | 14 MB  00:00:0
   1
44. Running transaction check
45. Running transaction test
46. Transaction test succeeded
47. Running transaction
48. Installing : apr-1.4.8-
   3.el7.x86_64                                              1/33
49. Installing : apr-util-1.5.2-
   6.0.1.el7.x86_64                                         2/33
50. Installing : httpd-tools-2.4.6-
   40.0.1.el7_2.1.x86_64                                    3/33
51. ...
52.
53. Installed:
54. httpd.x86_64 0:2.4.6-40.0.1.el7_2.1                    perl.x86_64 4:5.16.3-286.el7
55.
56. Dependency Installed:
```

```

57.  apr.x86_64 0:1.4.8-3.el7      apr-util.x86_64 0:1.5.2-6.0.1.el7      groff-base.x86_64 0:1.22.2-
    8.el7      httpd-tools.x86_64 0:2.4.6-40.0.1.el7_2.1
58.  mailcap.noarch 0:2.1.41-2.el7    perl-Carp.noarch 0:1.26-244.el7      perl-Encode.x86_64 0:2.51-
    7.el7      perl-Exporter.noarch 0:5.68-3.el7
59.  perl-File-Path.noarch 0:2.09-2.el7  perl-File-Temp.noarch 0:0.23.01-3.el7  perl-Filter.x86_64 0:1.49-
    3.el7      perl-Getopt-Long.noarch 0:2.40-2.el7
60.  perl-HTTP-Tiny.noarch 0:0.033-3.el7  perl-PathTools.x86_64 0:3.40-5.el7      perl-Pod-Escapes.noarch 1:1.04-
    286.el7    perl-Pod-Perldoc.noarch 0:3.20-4.el7
61.  perl-Pod-Simple.noarch 1:3.28-4.el7  perl-Pod-Usage.noarch 0:1.63-3.el7      perl-Scalar-List-Utils.x86_64 0:1.27-
    248.el7    perl-Socket.x86_64 0:2.010-3.el7
62.  perl-Storable.x86_64 0:2.45-3.el7  perl-Text-ParseWords.noarch 0:3.29-4.el7  perl-Time-HiRes.x86_64 4:1.9725-
    3.el7      perl-Time-Local.noarch 0:1.2300-2.el7
63.  perl-constant.noarch 0:1.27-2.el7  perl-libs.x86_64 4:5.16.3-286.el7      perl-macros.x86_64 4:5.16.3-
    286.el7    perl-parent.noarch 1:0.225-244.el7
64.  perl-podlators.noarch 0:2.5.1-3.el7  perl-threads.x86_64 0:1.87-4.el7      perl-threads-shared.x86_64 0:1.43-6.el7
65.
66.  Complete!
67.  Loaded plugins: ulninfo
68.  Cleaning repos: ol7_UEKR3 ol7_latest
69.  Cleaning up everything

```

At this point, We need to configure some content for the web server to display. For simplicity, let's create a basic opening page in the `/var/www/html` directory on the guest:

```

1.  [root@722c42b3c151 /]# echo "Example Web Server Content" > /var/www/html/index.html

```

The guest container is now configured with the software environment that we want. Typing `exit` stops the running guest, returning to the prompt for the Linux host:

```

1.  [root@722c42b3c151 /]# exit
2.  exit
3.  [holuser@docker ~]$

```

We now need to save this modified guest as an image that can be used to create new containers. To get the ID of the modified container, we use the `docker ps` command with the `--latest` and `--quiet` parameters. This will output just the ID of the latest guest to be created or stopped:

```

1.  [holuser@docker ~]$ docker ps --latest --quiet
2.  722c42b3c151

```

The `docker ps` command lists currently running containers by default. If we run it now, there are no results (because we have no running containers at the moment):

```

1.  [holuser@docker ~]$ docker ps

```


2.	CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	N
	AMES						

We can add the `--all` parameter to list both running and stopped containers:

1.	[holuser@docker ~]\$ docker ps --all						
2.	CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	
	NAMES						
3.	722c42b3c151	oraclelinux	"/bin/bash"	5 minutes ago	Exited (0)	About a minute ago	gu
	est						

We can combine the output from the `docker ps --latest --quiet` command with the `docker commit` command to save the last-running guest as a new image:

```
1. [holuser@docker ~]$ docker commit -m "OL7-httpd" `docker ps -l -q` doc-1002902/httpd:r1
2. sha256:c6735c08eeab980aaa8206f8ebc60cb4838788290c1e7ab92567f477963ab57c
```

Let's break down this command. To save an image, we use `docker commit` with a message (`-m`) parameter. We're then embedding the output of the `docker ps -l -q` command (using the short versions of the `--latest` and `--quiet` parameters) and finally providing a namespace, image name and tag for the image. We're saving our image into the `hol10328` namespace with the image name of `httpd` and the tag, `r1`. If you want to save an image to the Docker Hub, you will need to use the namespace you created when you signed up.

If we take a look at the images now, we'll see that our image is now available:

1.	[holuser@docker ~]\$ docker images				
2.	REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
3.	doc-1002902/httpd	r1	c6735c08eeab	26 seconds ago	332.3 MB
4.	oraclelinux	6	3956a77fcc51	2 weeks ago	223.1 MB
5.	oraclelinux	latest	5f5dae795065	2 weeks ago	278.1 MB
6.	mysql/mysql-server	latest	18a962a188ee	5 weeks ago	366.9 MB

Deploying the Configured Docker Image (Optional)

We can deploy any number of web servers now using the new Docker image as a template. The following `docker run` commands run the container image `hol10328/httpd`, creating the containers `guest1`, `guest2`, and `guest3` and executing `httpd` in each one:

```
1. [holuser@docker ~]$ docker run -d --name guest1 -p 8081:80 doc-1002902/httpd:r1 /usr/sbin/httpd -
D FOREGROUND
2. 5339dd1872151b4e23e78e4576b3f92bb0a0c50d6e3efd30daa5abc74b1f9345
3. [holuser@docker ~]$ docker run -d --name guest2 -p 8082:80 doc-1002902/httpd:r1 /usr/sbin/httpd -
D FOREGROUND
```

4. d401a2b42f5eed77cc95d3231986b62360bf01cb5899954502aaecfb646342d
5. [holuser@docker ~]\$ docker run -d --name guest3 -p 8083:80 doc-1002902/httpd:r1 /usr/sbin/httpd -D FOREGROUND
6. 603792e96faed519757511b0c5b6a829d130eb9b1050ca67ffc586ab0a21591d

The `-p` argument maps port 80 in each guest to ports 8080, 8081, or 8082 on the host. Because we didn't use the `-i` or `-t` parameters, all three containers have started in the background. We can use the `docker ps` command to show the running guests:

1. [holuser@docker ~]\$ docker ps						
2. CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	
NAMES						
3. 603792e96fae	doc-1002902/httpd:r1	"/usr/sbin/httpd -D F"	18 seconds ago	Up 17 seconds	0.0.0.0:8083->80/tcp	guest3
4. d401a2b42f5e	doc-1002902/httpd:r1	"/usr/sbin/httpd -D F"	23 seconds ago	Up 22 seconds	0.0.0.0:8082->80/tcp	guest2
5. 5339dd187215	doc-1002902/httpd:r1	"/usr/sbin/httpd -D F"	28 seconds ago	Up 27 seconds	0.0.0.0:8081->80/tcp	guest1

The default IP address value of 0.0.0.0 means that the port mapping applies to all network interfaces on the host. Using a web browser or `curl`, we can test the web server running in each guest:

1. [holuser@docker ~]\$ curl http://docker.oracleworld.com:8081
2. Example Web Server Content
3. [holuser@docker ~]\$ curl http://docker.oracleworld.com:8082
4. Example Web Server Content
5. [holuser@docker ~]\$ curl http://docker.oracleworld.com:8083
6. Example Web Server Content

Using a Dockerfile (Optional)

Now that you've seen how to create and manipulate Docker containers using the command line, the preferred way to build and customize containers is actually using Dockerfiles. A Dockerfile is a small text file that contains the instructions required to construct a container. When a Dockerfile is built, each instruction adds a layer to the container in a step-by-step process. The build creates a container, runs the next instruction in that container, and then commits the container. Docker then runs the committed image as the basis for adding the next layer. The benefit of this layered approach is that Dockerfiles with the same initial instructions reuse layers.

Dockerfiles also create an easily readable and modifiable record of the steps used to create a Docker image. Publishing a Dockerfile to the public Docker Hub or to an internal repository is generally the preferred method of creating and sharing Docker images.

To create a Dockerfile, first create a directory for it:

1. [holuser@docker ~]\$ mkdir ~/dockerfile-httpd
2. [holuser@docker ~]\$ cd ~/dockerfile-httpd/

In that directory, use a text editor to create a file called Dockerfile that contains the following contents:

```
1. # Dockerfile for creating a Docker image for OL 7 and httpd and perl
2. FROM oraclelinux
3. MAINTAINER E. Jones <ejones@email-address.com>
4. RUN yum install -y httpd perl && yum clean all
5. RUN echo "Example Web Server Content" > /var/www/html/index.html
6. EXPOSE 80
7. CMD /usr/sbin/httpd -D FOREGROUND
```

This Dockerfile reflects the same steps as the previous exercise in which we manually built the Docker image for my web server: it configures a new container from a base Oracle Linux 7 image, installs the httpd and perl RPMs, and adds placeholder content for the opening web page.

The `docker build` command constructs a new Docker image from this Dockerfile, creating and removing temporary containers as needed during its step-by-step build process:

```
1. [holuser@docker ~]$ docker build -t doc-1002902/httpd:r2 ~/dockerfile-httpd/.
2. Sending build context to Docker daemon 2.048 kB
3. Step 1 : FROM oraclelinux
4. ---> 5f5dae795065
5. Step 2 : MAINTAINER E. Jones ---> Running in 80809c13f726
6. ---> 40fc9a14038e
7. Removing intermediate container 80809c13f726
8. Step 3 : RUN yum install -y httpd perl && yum clean all
9. ---> Running in 99f88a5ce9ff
10. Loaded plugins: ulninfo
11. Resolving Dependencies
12. --> Running transaction check
13. ---> Package httpd.x86_64 0:2.4.6-40.0.1.el7_2.1 will be installed
14. ...
15. --> Finished Dependency Resolution
16.
17. Dependencies Resolved
18.
19. =====
20. Package           Arch   Version             Repository   Size
21. =====
22. Installing:
23. httpd             x86_64 2.4.6-40.0.1.el7_2.1 ol7_latest 1.2 M
24. perl              x86_64 4:5.16.3-286.el7    ol7_latest 8.0 M
25. ...
26.
27. Transaction Summary
28. =====
29. Install 2 Packages (+31 Dependent packages)
30.
31. Total download size: 14 M
32. Installed size: 44 M
```

```

33. Downloading packages:
34. -----
35. Total                      4.0 MB/s | 14 MB  00:03
36. Running transaction check
37. Running transaction test
38. Transaction test succeeded
39. Running transaction
40.   Installing : apr-1.4.8-3.el7.x86_64                1/33
41.   Installing : apr-util-1.5.2-6.0.1.el7.x86_64         2/33
42.   ...
43.
44. Installed:
45.   httpd.x86_64 0:2.4.6-40.0.1.el7_2.1      perl.x86_64 4:5.16.3-286.el7
46.
47. Dependency Installed:
48.   apr.x86_64 0:1.4.8-3.el7
49.   apr-util.x86_64 0:1.5.2-6.0.1.el7
50.   ...
51.
52. Complete!
53. Loaded plugins: ulninfo
54. Cleaning repos: o17_UEKR3 o17_latest
55. Cleaning up everything
56. ---> 4d8683e7f9b9
57. Removing intermediate container 99f88a5ce9ff
58. Step 4 : RUN echo "Example Web Server Content" > /var/www/html/index.html
59. ---> Running in 74b86752bbcf
60. ---> 09928d191d8e
61. Removing intermediate container 74b86752bbcf
62. Step 5 : EXPOSE 80
63. ---> Running in 6b6d859399e0
64. ---> 16006844f470
65. Removing intermediate container 6b6d859399e0
66. Step 6 : CMD /usr/sbin/httpd -D FOREGROUND
67. ---> Running in 1eb525d7b1e6
68. ---> b28b7dca2dd1
69. Removing intermediate container 1eb525d7b1e6
70. Successfully built b28b7dca2dd1

```

The `docker images` command lists the new `doc-1002902/httpd:r2` image created from the Dockerfile:

```

1. [holuser@docker ~]$ docker images
2.  REPOSITORY          TAG          IMAGE ID          CREATED          SIZE
3.  doc-1002902/httpd   r2           b28b7dca2dd1     About a minute ago  332.3 MB
4.  doc-1002902/httpd   r1           c6735c08eeab     8 minutes ago    332.3 MB
5.  oraclelinux         6            3956a77fcc51     2 weeks ago      223.1 MB
6.  oraclelinux         latest       5f5dae795065     2 weeks ago      278.1 MB
7.  mysql/mysql-server   latest       18a962a188ee     5 weeks ago      366.9 MB

```

Limiting Runtime Resources (Optional)

A Docker container uses underlying control group (cgroup) technologies, creating a cgroup for each Docker container. Cgroups provide useful capabilities, such as collecting metrics for CPU, memory, and block I/O usage and providing resource management capabilities. For each Docker container, you can find metrics in the host's corresponding cgroup hierarchy; for example, on an Oracle Linux 7 host running the Unbreakable Enterprise Kernel Release 3 (UEKR3), a `memory.stat` file in the directory `/sys/fs/cgroup/memory/system.slice/docker-<container ID>` lists memory-related metrics for that container. (See the Docker blog article "[Gathering LXC and Docker container metrics](#).")

The `docker run` command provides options that enable runtime limits for memory and relative CPU allocations. These options provide a degree of resource control when executing container images. The `-m` or `--memory` option limits the amount of physical and swap memory available to processes within a container. For example, the following command places a limit of 256 MB for physical memory (and up to 512 MB for memory plus swap):

1. `[holuser@docker ~]$ docker run -d --memory=256m --name guest10 -p 8090:80 doc-1002902/httpd:r2`
2. `e8132ca96e46dd3c03113c5facc6331bd5dd7fee6836331ba2578348d6a5605f`

The `-c` or `--cpushares` option provides a method of assigning a relative number of CPU shares:

1. `[holuser@docker ~]$ docker run -d --cpu-shares=512 --name guest11 -p 8091:80 doc-1002902/httpd:r2`
2. `a21d4110898c2811c36b62006e17a57e6ddcc14b986d8a03cb650826f65ac273`

By default, a container gets 1024 CPU shares, so in this example, the `guest2` container gets a relative 50 percent share of the available computing resources. Note that the host kernel applies limits only when there is resource contention. For example, if `guest2` is assigned 512 shares (50 percent, as shown above) and `guest1` is assigned 1024 shares (100 percent), processes in `guest2` will still get 100 percent of the available CPU resources if all `guest1` processes are idle.

For more information about options to the `docker run` command, enter `docker run --help` or see the [Docker run reference guide](#).

Connecting a Web Server Container to a MySQL Container (Optional)

Suppose we want to connect my web server container to a container running an instance of MySQL. Docker makes it possible to link containers together, creating a secure channel for one container to access certain information from another.

Previously we pulled the Oracle MySQL image from the Docker Hub using the command `docker pull mysql/mysql-server`. To run the MySQL image as a container named `db`, we need to create a location to store the MySQL data and start a new container:

1. `[holuser@docker ~]$ cd ~`
2. `[holuser@docker ~]$ mkdir mysql-datadir`

1. [holuser@docker ~]\$ docker run --name db -d -e MYSQL_ROOT_PASSWORD=Oracle123 -e MYSQL_DATABASE=webdb -e MYSQL_USER=webuser -e MYSQL_PASSWORD=secret -v /home/holuser/mysql-datadir:/var/lib/mysql mysql/mysql-server
4. dc275a55ff987471402b8fd3e4e9147e9e785d7d19482150f59ef4ccf0d63458

Let's break down the run command: we start a container named `db` in the background using the `-d` (detached) parameter. We then set a series of environment variables using multiple `-e` parameters. The MySQL image allows you to customize the database that is created on startup by using environment variables to configure the container. In this example, we are setting the MySQL root password to `Oracle123` and creating a new database called `webdb`. We then create a new MySQL user called `webuser` with the password of `secret`. The container automatically assigns the correct permission to the user for the database it creates.

Using the `-v` or `--volume` flag is how you can associate database storage (or, in general, any files or folders) that a Docker container needs to change while it's running. Remember that Docker containers are "ephemeral"; that is, they are transient in nature and data does not persist beyond the lifespan of a container's execution. Associating a data volume with a container using the `-v` flag allows the container to access, modify, and persist data on the host.

Reminiscent of an NFS mount, the `-v` flag causes the directory `/home/holuser/mysql-datadir` on the host to be mounted as `/var/lib/mysql` (the default database location for MySQL) on the container. From an operational standpoint, the ability to mount a database directory as a container volume in this way also provides an easy way to enable database backups.

The Docker Hub `mysql/mysql-server` image includes an entrypoint script (`entrypoint.sh`) that sets up the database server automatically, initializing `mysqld`. We can view log messages generated from a container using the `docker logs` command:

1. [holuser@docker ~]\$ docker logs db
2. Initializing database
3. Database initialized
4. MySQL init process in progress...
5. Warning: Unable to load '/usr/share/zoneinfo/iso3166.tab' as time zone. Skipping it.
6. Warning: Unable to load '/usr/share/zoneinfo/zone.tab' as time zone. Skipping it.
7. mysql: [Warning] Using a password on the command line interface can be insecure.
8. mysql: [Warning] Using a password on the command line interface can be insecure.
9. mysql: [Warning] Using a password on the command line interface can be insecure.
10. mysql: [Warning] Using a password on the command line interface can be insecure.
- 11.
12. /entrypoint.sh: ignoring /docker-entrypoint-initdb.d/*
- 13.
- 14.
15. MySQL init process done. Ready for start up.

We now have a MySQL container running. Next, we need to create a web server container that contains code to connect to the database and retrieve information.

Let's create a new Dockerfile that configures a web server that includes some Perl scripts:

1. [holuser@docker ~]\$ cd ~

2. [holuser@docker ~]\$ mkdir dockerfile-linked-httpd
3. [holuser@docker ~]\$ cd dockerfile-linked-httpd/
4. [holuser@docker dockerfile-linked-httpd]\$

Use your favourite text editor to create a Dockerfile with the following content:

1. FROM oraclelinux:7
2. RUN yum install -y httpd perl perl-DBI.x86_64 libdbi-dbd-mysql.x86_64 perl-DBD-MySQL.x86_64 && yum clean all
3. ADD version.pl /var/www/cgi-bin/version.pl
4. RUN chmod 755 /var/www/cgi-bin/version.pl
5. ADD initdb.pl /var/www/cgi-bin/initdb.pl
6. RUN chmod 755 /var/www/cgi-bin/initdb.pl
7. ADD doquery.pl /var/www/cgi-bin/doquery.pl
8. RUN chmod 755 /var/www/cgi-bin/doquery.pl
9. EXPOSE 80
10. ENTRYPOINT /usr/sbin/httpd -D FOREGROUND

This Dockerfile introduces the new command `ADD` which allows us to add a file from the host into the container during build. In order to successfully build this container, we need to create the three Perl scripts on the host, in the same directory as the Dockerfile itself.

version.pl:

1. #!/usr/bin/perl
2. use DBI;
3. print "Content-type: text/html\n\n";
4. my \$dbh = DBI->connect(
5. "dbi:mysql:dbname=webdb:host=db",
6. "webuser",
7. "secret",
8. { RaiseError => 1 },
9.) or die \$DBI::errstr;
10. my \$sth = \$dbh->prepare("SELECT VERSION()");
11. \$sth->execute();
12. my \$ver = \$sth->fetch();
13. print "Version = ", @\$ver, "\n";
14. \$sth->finish();
15. \$dbh->disconnect()

initdb.pl

1. #!/usr/bin/perl
2. use strict;
3. use DBI;
4. print "Content-type: text/html\n\n";
5. my \$dbh = DBI->connect(

```

6.     "dbi:mysql:dbname=webdb:host=db",
7.     "webuser",
8.     "secret",
9.     { RaiseError => 1 }
10. ) or die $DBI::errstr;
11. $dbh->do("DROP TABLE IF EXISTS People");
12. $dbh->do("CREATE TABLE People(Id INT PRIMARY KEY, Name TEXT, Age INT) ENGINE=InnoDB");
13. $dbh->do("INSERT INTO People VALUES(1,'Alice',42)");
14. $dbh->do("INSERT INTO People VALUES(2,'Bobby',27)");
15. $dbh->do("INSERT INTO People VALUES(3,'Carol',29)");
16. $dbh->do("INSERT INTO People VALUES(4,'Daisy',20)");
17. $dbh->do("INSERT INTO People VALUES(5,'Eddie',35)");
18. $dbh->do("INSERT INTO People VALUES(6,'Frank',21)");
19. my @noerr = ('Rows inserted in People table');
20. print @noerr;
21. print "\n";
22. my $sth = $dbh->prepare( "SELECT * FROM People" );
23. $sth->execute();
24.
25. for ( 1 .. $sth->rows() ) {
26.     my ($id, $name, $age) = $sth->fetchrow();
27.     print "$id $name $age\n";
28. }
29. $sth->finish();
30. $dbh->disconnect();

```

doquery.pl

```

1.  #!/usr/bin/perl
2.  use strict;
3.  use DBI;
4.  print "Content-type: text/html\n\n";
5.  my $dbh = DBI->connect(
6.      "dbi:mysql:dbname=webdb:host=db",
7.      "webuser",
8.      "secret",
9.      { RaiseError => 1 },
10. ) or die $DBI::errstr;
11. my $sth = $dbh->prepare( "SELECT * FROM People WHERE Age > $ARGV[0]" );
12. $sth->execute();
13. my $fields = $sth->{NUM_OF_FIELDS};
14. my $rows = $sth->rows();
15. print "Selected $rows row(s) with $fields field(s)\n";
16. for ( 1 .. $rows ) {
17.     my ($id, $name, $age) = $sth->fetchrow();
18.     print "$id $name $age\n";
19. }
20. $sth->finish();
21. $dbh->disconnect();

```


These three scripts will allow us to create a basic MySQL database using `initdb.pl`, then query the MySQL server using `version.pl` and finally query the MySQL database we created using `doquery.pl`.

Before we can run the web server, we need to build a new image using the `docker build` command. Using what you learned in the previous exercise, build a new Docker image for `hol10328/httpd` with the tag `R3`.

Once your new Docker image is created, you can run it:

1. [holuser@docker dockerfile-linked-httpd]\$ `docker run -d --name webdb -p 8080:80 --link db:db doc-1002902/httpd:r3`
2. `06100187714ffe4de81079c8065e57383fa97104578c1b32ac2fd2a65378647f`

By linking containers using the `--link` flag, other application containers can access the MySQL database running in the `db` container. This simplifies the separation of database, application, and web services, making it easier to isolate services in different containers. Docker automatically configures networking and the `/etc/hosts` file in each linked container.

Finally, we can run the Perl scripts in the web server container to create and query the database running in the database container

1. [holuser@docker dockerfile-linked-httpd]\$ `curl http://docker.oracleworld.com:8080/cgi-bin/version.pl`
2. `Version = 5.7.13`

1. [holuser@docker dockerfile-linked-httpd]\$ `curl http://docker.oracleworld.com:8080/cgi-bin/initdb.pl`
2. `Rows inserted in People table`
3. `1 Alice 42`
4. `2 Bobby 27`
5. `3 Carol 29`
6. `4 Daisy 20`
7. `5 Eddie 35`
8. `6 Frank 21`

1. [holuser@docker dockerfile-linked-httpd]\$ `curl http://docker.oracleworld.com:8080/cgi-bin/doquery.pl?30`
2. `Selected 2 row(s) with 3 field(s)`
3. `1 Alice 42`
4. `5 Eddie 35`

1. [holuser@docker dockerfile-linked-httpd]\$ `curl http://docker.oracleworld.com:8080/cgi-bin/doquery.pl?21`
2. `Selected 4 row(s) with 3 field(s)`
3. `1 Alice 42`
4. `2 Bobby 27`
5. `3 Carol 29`
6. `5 Eddie 35`

The Oracle Linux documentation includes a [detailed example of how to link a web container with a MySQL database](#), and it provides sample scripts that configure httpd.conf, create a database, and perform queries. Additional background on linking containers and using data volumes is available there and also in the [Docker user guide](#).

Before proceeding to the Oracle Database and Oracle Weblogic Server labs make sure to stop and remove all running containers by usage of ***docker container rm -force <container_id> <container_id>.....***

You can get <container_id> from ***docker ps --all***

Using the the official docker-certified Oracle Database Enterprise image

Oracle Database Server 12c R2 is an industry leading relational database server. The Oracle Database Server Docker Image contains the Oracle Database Server 12.2.0.1 Enterprise Edition running on Oracle Linux 7. This image contains a default database in a multitenant configuration with one pdb.

Let's get started. From the store.docker.com website accept Terms of Service for Oracle Database Enterprise Edition. If you don't have a Docker Store Account go to <https://store.docker.com/> and register and you will be ready.

Login to Docker Store with your credentials

1. \$ docker login

Starting an Oracle database server instance is as simple as executing

1. \$ [holuser@docker ~]\$ docker run -d -it --name MyDB1 store/oracle/database-enterprise:12.2.0.1
2. 7888c44d91da112c741c2afc9745e37a0f41fbbc2f55816356bb2a79fb775d39
3. [holuser@docker ~]\$

where "MyDB1" is the name of the container and 12.2.0.1 is the Docker image tag. The database server is ready to use when the STATUS field shows (healthy) in the output of docker ps.

1. [holuser@docker ~]\$ docker ps
2. CONTAINER ID IMAGE COMMAND CREATED STATUS
3. 7888c44d91da store/oracle/database-enterprise:12.2.0.1 "/bin/sh -c '/bin/..." 2 minutes ago Up 2 minutes
4. [holuser@docker ~]\$

Connecting to the Database Server Container

The default password to connect to the database with sys user is Oradoc_db1.

Connecting from within the container

The database server can be connected to by executing SQL*Plus,

```

5. [holuser@docker ~]$ docker exec -it MyDB1 bash -c "source /home/oracle/.bashrc; sqlplus /nolog"
6.
7. SQL*Plus: Release 12.2.0.1.0 Production on Tue Jan 16 09:43:53 2018
8.
9. Copyright (c) 1982, 2016, Oracle. All rights reserved.
10.
11. SQL>connect sys/Oradoc_db1 as sysdba
12. Connected.
13. SQL>

```

Connecting from outside the container

The database server exposes port 1521 for Oracle client connections over *SQLNet protocol* and port 5500 for *Oracle XML DB*. *SQLPlus* or any JDBC client can be used to connect to the database server from outside the container.

In the following we will use *sqlplus* to connect to the database instance running in a Docker container. To do that we first need to install *sqlplus* instant client on our vbox image.

Open Firefox – go to <http://www.oracle.com/technetwork/topics/linuxx86-64soft-092277.html>, login in to Oracle OTN, accept license agreement and download the following version 12.2.0.1.0:

- Instant Client Package - Basic: All files required to run OCI, OCCI, and JDBC-OCI applications
- Instant Client Package - SQL*Plus: Additional libraries and executable for running SQL*Plus with Instant Client

Embedded

BI & Data Warehousing

.NET

New to Java

Cloud Computing

Big Data

Security

Enterprise Architecture

Digital Experience

Service-Oriented Architecture

Virtualization

Mobile Computing

Instant Client Downloads for Linux x86-64

Thank you for accepting the OTN License Agreement; you may now download this software.

See the [Instant Client Home Page](#) for more information about Instant Client.

The [installation instructions](#) are at the foot of the page.

Version 12.2.0.1.0

Instant Client Package - Basic: All files required to run OCI, OCCI, and JDBC-OCI applications

[instantclient-basic-linux.x64-12.2.0.1.0.zip](#) (68,965,195 bytes) (cksum - 3923339140)
[oracle-instantclient12.2-basic-12.2.0.1.0-1.x86_64.rpm](#) (52,826,628 bytes) (cksum - 888077889)

Instant Client Package - Basic Light: Smaller version of the Basic package, with only English error messages and Unicode, ASCII, and Western European character set support

[instantclient-basiclite-linux.x64-12.2.0.1.0.zip](#) (32,917,466 bytes) (cksum - 2936437456)
[oracle-instantclient12.2-basiclite-12.2.0.1.0-1.x86_64.rpm](#) (26,731,248 bytes) (cksum - 1705759050)

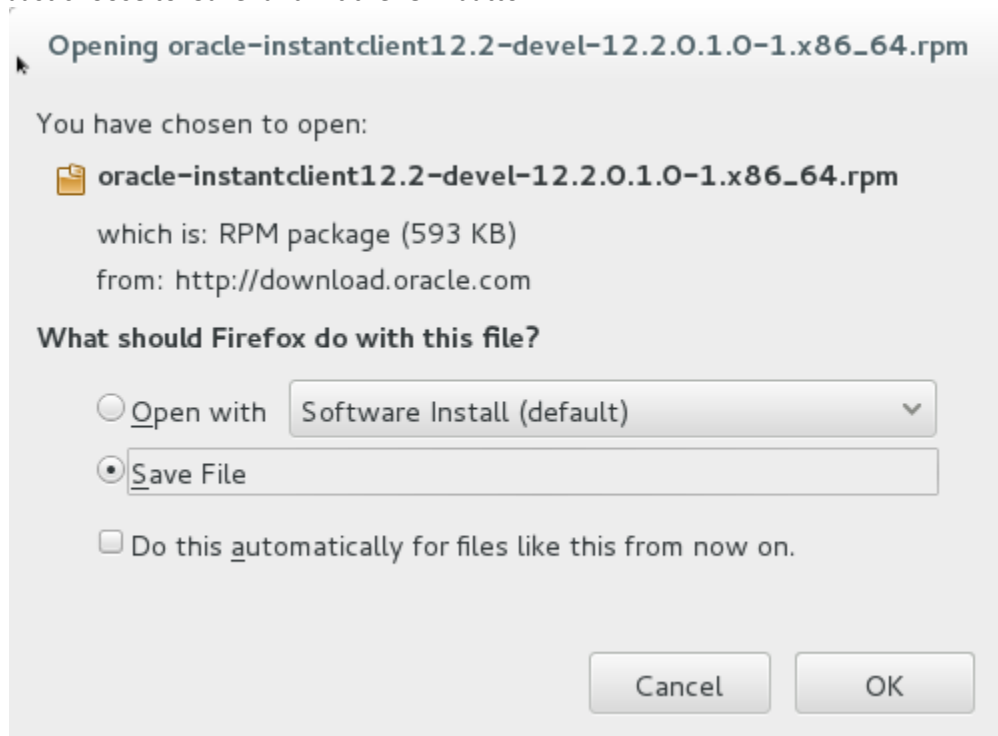
Instant Client Package - JDBC Supplement: Additional support for XA, Internationalization, and RowSet operations under JDBC

[instantclient-jdbc-linux.x64-12.2.0.1.0.zip](#) (1,572,942 bytes) (cksum - 1346784794)
[oracle-instantclient12.2-jdbc-12.2.0.1.0-1.x86_64.rpm](#) (1,524,160 bytes) (cksum - 723838987)

Instant Client Package - SQL*Plus: Additional libraries and executable for running SQL*Plus with Instant Client

[instantclient-sqlplus-linux.x64-12.2.0.1.0.zip](#) (904,309 bytes) (cksum - 2291973160)
[oracle-instantclient12.2-sqlplus-12.2.0.1.0-1.x86_64.rpm](#) (708,104 bytes) (cksum - 9377165168)

Just choose to 'Save' and hit the 'OK' button.



Open a Terminal, go to 'Downloads' directory and execute the following:

```
1. [holuser@docker Downloads]$ sudo yum install oracle-instantclient12.2-basic-12.2.0.1.0-1.x86_64.rpm
2. Loaded plugins: langpacks, ulninfo
3. Examining oracle-instantclient12.2-basic-12.2.0.1.0-1.x86_64.rpm: oracle-instantclient12.2-basic-12.2.0.1.0-1.x86_64
4. Marking oracle-instantclient12.2-basic-12.2.0.1.0-1.x86_64.rpm to be installed
5. Resolving Dependencies
6. --> Running transaction check
7. --> Package oracle-instantclient12.2-basic.x86_64 0:12.2.0.1.0-1 will be installed
8. --> Finished Dependency Resolution
9.
10. Dependencies Resolved
11.
12. =====
13. Package
14.    Arch Version
15.    Repository Size
16. =====
17. Installing:
18.  oracle-instantclient12.2-basic
19.    x86_64 12.2.0.1.0-1
20.    /oracle-instantclient12.2-basic-12.2.0.1.0-1.x86_64 211 M
21.
22. Transaction Summary
23. =====
24. Install 1 Package
25.
26. Total size: 211 M
27. Installed size: 211 M
```

```

28. Is this ok [y/d/N]: y
29. Downloading packages:
30. Running transaction check
31. Running transaction test
32. Transaction test succeeded
33. Running transaction
34. Installing : oracle-instantclient12.2-basic-12.2.0.1.0-1.x86_64      1/1
35. Verifying : oracle-instantclient12.2-basic-12.2.0.1.0-1.x86_64      1/1
36.
37. Installed:
38.  oracle-instantclient12.2-basic.x86_64 0:12.2.0.1.0-1
39.
40. Complete!
41. [holuser@docker Downloads]$

```

And the other packages:

```

1. [holuser@docker Downloads]$ sudo yum install oracle-instantclient12.2-sqlplus-12.2.0.1.0-1.x86_64.rpm
2. [sudo] password for holuser:
3. Loaded plugins: langpacks, ulninfo
4. Examining oracle-instantclient12.2-sqlplus-12.2.0.1.0-1.x86_64.rpm: oracle-instantclient12.2-sqlplus-12.2.0.1.0-1.x86_64
5. Marking oracle-instantclient12.2-sqlplus-12.2.0.1.0-1.x86_64.rpm to be installed
6. Resolving Dependencies
7. --> Running transaction check
8. ---> Package oracle-instantclient12.2-sqlplus.x86_64 0:12.2.0.1.0-1 will be installed
9. --> Finished Dependency Resolution
10.
11. Dependencies Resolved
12.
13. =====
14. Package
15. Arch Version
16. Repository Size
17. =====
18. Installing:
19. oracle-instantclient12.2-sqlplus
20. x86_64 12.2.0.1.0-1
21. /oracle-instantclient12.2-sqlplus-12.2.0.1.0-1.x86_64 3.1 M
22.
23. Transaction Summary
24. =====
25. Install 1 Package
26.
27. Total size: 3.1 M
28. Installed size: 3.1 M
29. Is this ok [y/d/N]: y
30. Downloading packages:
31. Running transaction check
32. Running transaction test
33. Transaction test succeeded
34. Running transaction
35. Installing : oracle-instantclient12.2-sqlplus-12.2.0.1.0-1.x86_64      1/1
36. Verifying : oracle-instantclient12.2-sqlplus-12.2.0.1.0-1.x86_64      1/1
37.
38. Installed:
39.  oracle-instantclient12.2-sqlplus.x86_64 0:12.2.0.1.0-1
40.
41. Complete!
42. [holuser@docker Downloads]$

```

To finish installation execute:

1. [holuser@docker Downloads]\$ sudo sh -c "echo /usr/lib/oracle/12.2/client64/lib > /etc/ld.so.conf.d/oracle-instantclient.conf"
2. [holuser@docker Downloads]\$ sudo ldconfig
3. [holuser@docker Downloads]\$ export PATH=/usr/lib/oracle/12.2/client64/bin:\$PATH
4. [holuser@docker Downloads]\$ which sqlplus
5. /usr/lib/oracle/12.2/client64/bin/sqlplus
6. [holuser@docker Downloads]\$

To connect from outside the container start the container with -P or -p option as,

1. \$ docker run -d -it --name MyDB2 -P store/oracle/database-enterprise:12.2.0.1

option -P indicates the ports are allocated by Docker. The mapped port can be discovered by executing

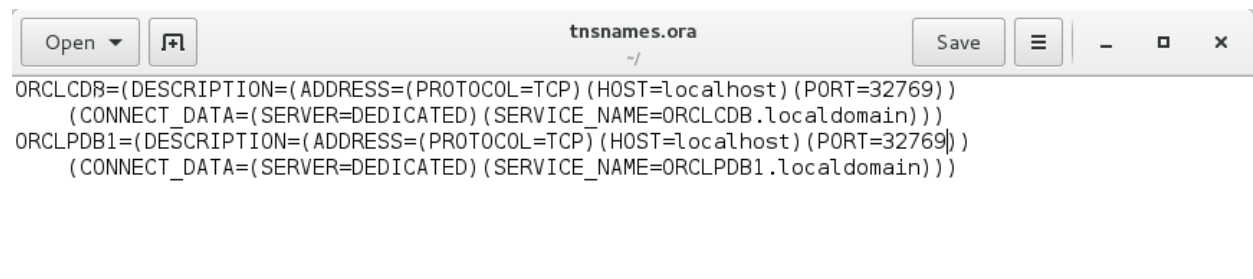
1. \$ [holuser@docker ~]\$ docker port MyDB2
2. 1521/tcp -> 0.0.0.0:32769
3. 5500/tcp -> 0.0.0.0:32768
4. [holuser@docker ~]\$

Using this <mapped host port> and <ip-address of host> create tnsnames.ora in the directory pointed to by environment variable TNS_ADMIN.

```
ORCLCDB=(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=<ip-address of
host>)(PORT=<mapped host port>))
(CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=ORCLCDB.localdomain)))
ORCLPDB1=(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=<ip-address> of
host)(PORT=<mapped host port>))
(CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=ORCLPDB1.localdomain)))
```

Go to **/home/holuser** ,use gedit to create tnsnames.ora and set the TNS_ADMIN env variable:

1. [holuser@docker ~]\$ gedit tnsnames.ora
2. [holuser@docker ~]\$



```
tnsnames.ora
~/
ORCLCDB=(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=32769))
(CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=ORCLCDB.localdomain)))
ORCLPDB1=(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=32769))
(CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=ORCLPDB1.localdomain)))
```

3. [holuser@docker ~]\$ export TNS_ADMIN=/home/holuser
4. [holuser@docker ~]\$

To connect from outside the container using SQL*Plus,

1. \$ sqlplus sys/Oradoc_db1@ORCLCDB as sysdba

Custom Configurations

The Oracle database server container also provides custom configuration parameters for starting up the container. All the custom configuration parameters are optional. The following list of custom configuration parameters can be provided in the ENV file (ora.conf).

DB_SID

This parameter changes the ORACLE_SID of the database. The default value is set to ORCLCDB.

DB_PDB

This parameter modifies the name of the PDB. The default value is set to ORCLPDB1.

DB_MEMORY

This parameter sets the memory requirement for the Oracle server. This value determines the amount of memory to be allocated for SGA and PGA. The default value is set to 2GB.

DB_DOMAIN

This parameter sets the domain to be used for database server. The default value is localdomain.

Use gedit to create an *env.conf* with the following:



```
DB_SID=test
DB_MEMORY=4g
DB_PDB=pdb1
DB_DOMAIN=mydomain|
```

Save it!

To start an Oracle database server with custom configuration parameters

1. [holuser@docker ~]\$ docker run -d -it --name MyDB3 -P --env-file env.conf store/oracle/database-enterprise:12.2.0.1
2. 65728ae7dcd8a0728f583c0c6167ec21c1ac5eafe791843f06efbcef76ed6802
3. [holuser@docker ~]\$

Ensure custom values for DB_SID, DB_PDB and DB_DOMAIN are updated in the tnsnames.ora.

```
4. [holuser@docker ~]$ sqlplus sys/Oradoc_db1@test as sysdba
5.
6. SQL*Plus: Release 12.2.0.1.0 Production on Tue Jan 16 06:28:19 2018
7.
8. Copyright (c) 1982, 2016, Oracle. All rights reserved.
9.
10. Last Successful login time: Tue Jan 16 2018 06:26:19 -08:00
11.
12. Connected to:
13. Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production
14.
15. SQL>
```

Caveats

This Docker image has the following restrictions.

1. Supports a single instance database.
2. Dataguard is not supported.
3. Database options and patching are not supported.

Changing default password for SYS user

The Oracle database server is started with a default password Oradoc_db1. The password used during the container creation is not secure and should be changed. To change the password connect to the database with SQL*Plus and execute

```
alter user sys identified by <new-password>;
```

Resource Requirements

The minimum requirements for the container is 8GB of disk space and 2GB of memory.

Database Logs

The database alert log can be viewed with

```
1. [holuser@docker ~]$ docker logs MyDB2
2. Setup Oracle Database
3. Oracle Database 12.2.0.1 Setup
4. Tue Jan 16 09:47:08 UTC 2018
5.
6. Check parameters .....
7. log file is : /home/oracle/setup/log/paramChk.log
8. paramChk.sh is done at 0 sec
9.
10. untar DB bits .....
11. log file is : /home/oracle/setup/log/untarDB.log
12. untarDB.sh is done at 97 sec
13.
14. config DB .....
15. log file is : /home/oracle/setup/log/configDB.log
16. Tue Jan 16 09:48:45 UTC 2018
```


17. Start Docker DB configuration
18.

Reusing existing database

This Oracle database server image uses Docker data volumes to store data files, redo logs, audit logs, alert logs and trace files. The data volume is mounted inside the container at /ORCL. To start a database with a data volume using docker run command,

```
$ docker run -d -it --name <Oracle-DB> -v OracleDBData:/ORCL store/oracle/database-enterprise:12.2.0.1
```

OracleDBData is the data volume that is created by Docker and mounted inside the container at /ORCL. The persisted data files can be reused with another container by reusing the OracleDBData data volume.

Using host system directory for data volume

To use a directory on the host system for the data volume,

```
[holuser@docker ~]$ docker run -d -it --name <Oracle-DB> -v /data/OracleDB:/ORCL store/oracle/database-enterprise:12.2.0.1
```

where */data/OracleDB* is a directory in the host system. NOTE: you need to use */ORCL*

Oracle Database Server 12.2.0.1 Enterprise Edition Slim Variant

The Slim Variant (12.2.0.1-slim tag) of EE has reduced disk space (4GB) requirements and a quicker container startup. This image does not support the following features - Analytics, Oracle R, Oracle Label Security, Oracle Text, Oracle Application Express and Oracle DataVault. To use the slim variant

```
$ docker run -d -it --name <Oracle-DB> store/oracle/database-enterprise:12.2.0.1-slim
```

where <Oracle-DB> is the name of the container and 12.2.0.1-slim is the Docker image tag.

This concludes the lab for the prebuilt Oracle Database Database Docker Image. Before proceeding to the next lab make sure to stop and remove all running containers by usage of ***docker container rm -force <container_id> <container_id>.....***

You can get <container_id> from ***docker ps --all***

How to build your own base Oracle Database Enterprise Docker Image

We have now seen how we can use the pre-build official Oracle Database Enterprise Docker Image to very fast and in an easy way to kick off an Oracle Database Enterprise Instance running in a Docker Container. But if you some reason need to build your own Oracle Database Docker Image it's simple to do so.

In the following we will build an Oracle Database Enterprise 12.2.0.1 Docker Image from scratch or more precise from the `oraclelinux:7-slim` Docker Image. We will be using a Docker File containing all the necessary steps to install the Oracle Database Enterprise 12.2.0.1.

Open a Terminal and go to the `/u01/OracleDatabase/dockerfiles/12.2.0.1` directory.

```

1. [holuser@docker ~]$ cd /u01/OracleDatabase/dockerfiles/12.2.0.1/
2. [holuser@docker 12.2.0.1]$
3. [holuser@docker 12.2.0.1]$ ls -l
4. total 76
5. -rwxr-xr-x 1 holuser holuser 1071 Jan  4 13:03 checkDBStatus.sh
6. -rwxr-xr-x 1 holuser holuser  904 Jan  4 13:03 checkSpace.sh
7. -rw-r--r-- 1 holuser holuser  62 Jan  4 13:03 Checksum.ee
8. -rw-r--r-- 1 holuser holuser  62 Jan  4 13:03 Checksum.se2
9. -rwxr-xr-x 1 holuser holuser 2953 Jan  4 13:03 createDB.sh
10. -rw-r--r-- 1 holuser holuser 9203 Jan  4 13:03 dbca.rsp.tmpl
11. -rw-r--r-- 1 holuser holuser 6878 Jan  4 13:03 db_inst.rsp
12. -rw-r--r-- 1 holuser holuser 2639 Jan  4 13:03 Dockerfile.ee
13. -rw-r--r-- 1 holuser holuser 2645 Jan  4 13:03 Dockerfile.se2
14. -rwxr-xr-x 1 holuser holuser 2250 Jan  4 13:03 installDBBinaries.sh
15. -rwxr-xr-x 1 holuser holuser 6140 Jan  4 13:03 runOracle.sh
16. -rwxr-xr-x 1 holuser holuser 1015 Jan  4 13:03 runUserScripts.sh
17. -rwxr-xr-x 1 holuser holuser  758 Jan  4 13:03 setPassword.sh
18. -rwxr-xr-x 1 holuser holuser  876 Jan  4 13:03 setupLinuxEnv.sh
19. -rwxr-xr-x 1 holuser holuser  678 Jan  4 13:03 startDB.sh
20. [holuser@docker 12.2.0.1]$

```

Open the **DockerFile.ee** in your favorite editor (like **gedit** or **vi**) and familize yourself with the content.

```

21. # LICENSE UPL 1.0
22. #
23. # Copyright (c) 1982-2017 Oracle and/or its affiliates. All rights reserved.
24. #
25. # ORACLE DOCKERFILES PROJECT
26. # -----
27. # This is the Dockerfile for Oracle Database 12c Release 2 Enterprise Edition
28. #
29. # REQUIRED FILES TO BUILD THIS IMAGE
30. # -----
31. # (1) linuxx64_12201_database.zip
32. #   Download Oracle Database 12c Release 12 Enterprise Edition for Linux x64
33. #   from http://www.oracle.com/technetwork/database/enterprise-edition/downloads/index.html
34. #
35. # HOW TO BUILD THIS IMAGE
36. # -----
37. # Put all downloaded files in the same directory as this Dockerfile
38. # Run:
39. #   $ docker build -t oracle/database:12.2.0.1-ee .
40. #
41. # Pull base image
42. # -----
43. FROM oraclelinux:7-slim
44.
45. # Maintainer
46. # -----
47. MAINTAINER Gerald Venzl <gerald.venzl@oracle.com>
48.

```

```

49. # Environment variables required for this build (do NOT change)
50. # -----
51. ENV ORACLE_BASE=/opt/oracle \
52.   ORACLE_HOME=/opt/oracle/product/12.2.0.1/dbhome_1 \
53.   INSTALL_FILE_1="linuxx64_12201_database.zip" \
54.   INSTALL_RSP="db_inst.rsp" \
55.   CONFIG_RSP="dbca.rsp.tmpl" \
56.   PWD_FILE="setPassword.sh" \
57.   RUN_FILE="runOracle.sh" \
58.   START_FILE="startDB.sh" \
59.   CREATE_DB_FILE="createDB.sh" \
60.   SETUP_LINUX_FILE="setupLinuxEnv.sh" \
61.   CHECK_SPACE_FILE="checkSpace.sh" \
62.   CHECK_DB_FILE="checkDBStatus.sh" \
63.   USER_SCRIPTS_FILE="runUserScripts.sh" \
64.   INSTALL_DB_BINARIES_FILE="installDBBinaries.sh"
65.
66. # Use second ENV so that variable get substituted
67. ENV INSTALL_DIR=$ORACLE_BASE/install \
68.   PATH=$ORACLE_HOME/bin:$ORACLE_HOME/OPatch/:/usr/sbin:$PATH \
69.   LD_LIBRARY_PATH=$ORACLE_HOME/lib:/usr/lib \
70.   CLASSPATH=$ORACLE_HOME/jlib:$ORACLE_HOME/rdbms/jlib
71.
72. # Copy binaries
73. # -----
74. COPY $INSTALL_FILE_1 $INSTALL_RSP $SETUP_LINUX_FILE $CHECK_SPACE_FILE
   $INSTALL_DB_BINARIES_FILE $INSTALL_DIR/
75. COPY $RUN_FILE $START_FILE $CREATE_DB_FILE $CONFIG_RSP $PWD_FILE $CHECK_DB_FILE
   $USER_SCRIPTS_FILE $ORACLE_BASE/
76.
77. RUN chmod ug+x $INSTALL_DIR/*.sh && \
78.   sync && \
79.   $INSTALL_DIR/$CHECK_SPACE_FILE && \
80.   $INSTALL_DIR/$SETUP_LINUX_FILE
81.
82. # Install DB software binaries
83. USER oracle
84. RUN $INSTALL_DIR/$INSTALL_DB_BINARIES_FILE EE
85.
86. USER root
87. RUN $ORACLE_BASE/oraInventory/orainstRoot.sh && \
88.   $ORACLE_HOME/root.sh && \
89.   rm -rf $INSTALL_DIR
90.
91. USER oracle
92. WORKDIR /home/oracle
93.
94. VOLUME ["$ORACLE_BASE/oradata"]
95. EXPOSE 1521 5500
96. HEALTHCHECK --interval=1m --start-period=5m \
97.   CMD "$ORACLE_BASE/$CHECK_DB_FILE" >/dev/null || exit 1
98.
99. # Define default command to start Oracle Database.
100. CMD exec $ORACLE_BASE/$RUN_FILE

```

All files used to install the binaries and create the database are referred to by environment variables in “ENV” section of the Dockerfile.ee.

First the database binaries are copied to the image – this happens in the “COPY” section of the Dockerfile.ee.

Secondly permissions of the files copied to installation directory in the image are changed and pre-installation checks are performed (Check space and setup Linux files) in the first “RUN” section of Dockerfile.ee.

Then the user is changed to “oracle” and in the second “RUN” section of Dockerfile.ee the installation of the database binaries is kickoff. When finished user is changed to “root” and the mandatory “orainstRoot.sh” is executed.

Finally a volume is created, the 1521 and 5500 ports are exposed in the “EXPOSE” section of the Dockerfile.ee and the command to start the database is executed in the “CMD” section of the Dockerfile.ee.

Now we have an understanding of the Dockerfile.ee we can create our customized Oracle Database Enterprise Docker image.

So copy the linux64_12201_database.zip from the software catalog to current catalog:

1. [holuser@docker 12.2.0.1]\$ cp /u01/software/linuxx64_12201_database.zip .
2. [holuser@docker 12.2.0.1]\$

And then run the Docker build command to build the image:

1. [holuser@docker 12.2.0.1]\$ docker build --force-rm=true --no-cache=true -t oracle/database:12.2.0.1-ee -f Dockerfile.ee .
2. Sending build context to Docker daemon 3.454GB
3. Step 1/17 : FROM oraclelinux:7-slim
4. 7-slim: Pulling from library/oraclelinux
5. 4040fe120662: Already exists
6. Digest: sha256:ef28f2f806afd778bb706fbf6ff90fc98e624c4113ea4d2f4aa508d813ef32fd
7. Status: Downloaded newer image for oraclelinux:7-slim
8. ---> 9870bebf1d5
9. Step 2/17 : MAINTAINER Gerald Venzl <gerald.venzl@oracle.com>
10. ---> Running in 2d10ad33dc5b
11. ---> 45182211efff
12.
13.
14. Step 15/17 : EXPOSE 1521 5500
15. ---> Running in e4684cdf7729
16. ---> 31364ab87488
17. Removing intermediate container e4684cdf7729
18. Step 16/17 : HEALTHCHECK --interval=1m --start-period=5m CMD "\$ORACLE_BASE/\$CHECK_DB_FILE" >/dev/null || exit 1
19. ---> Running in 0b1e60095a71
20. ---> fad68246955b
21. Removing intermediate container 0b1e60095a71
22. Step 17/17 : CMD exec \$ORACLE_BASE/\$RUN_FILE
23. ---> Running in 529e8881243b
24. ---> 8d99b7bc4f06

```

25. Removing intermediate container 529e8881243b
26. Successfully built 8d99b7bc4f06
27. Successfully tagged oracle/database:12.2.0.1-ee
28. [holuser@docker 12.2.0.1]$ docker images
29. REPOSITORY          TAG          IMAGE ID      CREATED      SIZE
30. oracle/database      12.2.0.1-ee  8d99b7bc4f06  2 minutes ago 13.3GB
31. store/oracle/weblogic 12.2.1.3     1fcee95cc21b  4 weeks ago  1.14GB
32. oraclelinux           latest       f426f15a5793  6 weeks ago  229MB
33. oraclelinux           7-slim      9870bebf1d5   6 weeks ago  118MB
34. oraclelinux           6           63f04a5ae058  2 months ago 171MB
35. mysql/mysql-server    latest      a3ee341faefb  2 months ago 246MB
36. store/oracle/database-enterprise 12.2.0.1    12a359cd0528  4 months ago 3.44GB
37. [holuser@docker 12.2.0.1]$

```

Please note – it will take some time to build the image – so please be patient ☺

After the build finishes successfully you run it by executing the following:

```

1. [holuser@docker 12.2.0.1]$ docker run -d -it --name MyCustomDB oracle/database:12.2.0.1-ee
2. 4c947678e5155244891d8e34cf7f7f32d0e46106094471294c317c300b69ea49
3. [holuser@docker 12.2.0.1]$ docker ps
4. CONTAINER ID   IMAGE          COMMAND          CREATED      STATUS
   PORTS         NAMES
5. 4c947678e515   oracle/database:12.2.0.1-ee  "/bin/sh -c 'exec ..."  15 seconds ago  Up 14 seconds (health:
   starting) 1521/tcp, 5500/tcp  MyCustomDB
6. [holuser@docker 12.2.0.1]$

```

The database server is ready to use when the STATUS field shows (healthy) in the output of docker ps.

You can find the password for SYS, SYSTEM and PDBADMIN by executing the following:

```

1. [holuser@docker 12.2.0.1]$ docker logs MyCustomDB | grep PASSWORD
2. ORACLE PASSWORD FOR SYS, SYSTEM AND PDBADMIN: JAQdUOqEDGk=1
3. [holuser@docker 12.2.0.1]$

```

To use Sqlplus you can do the same things as described before for the Official Oracle Database Enterprise Docker Image like:

```

4. [holuser@docker 12.2.0.1]$ docker exec -it MyCustomDB bash -c "source /home/oracle/.bashrc; sqlplus /nolog"
5.
6. SQL*Plus: Release 12.2.0.1.0 Production on Wed Jan 10 11:08:50 2018
7.
8. Copyright (c) 1982, 2016, Oracle. All rights reserved.
9.
10. SQL> connect sys/JAQdUOqEDGk=1@ORCLPDB1 as sysdba
11. Connected.
12. SQL>

```

Externalize persistence of oradata in host directory

To use a directory on the host system for the data volume and achieve external persistence first create the system directory on the host like **/home/holuser/OracleDBData** and execute Docker run with the **-v** option. NOTE remember to retrieve the password from the database log for later usage !!

1. [holuser@docker ~]\$ mkdir OracleDBData
2. [holuser@docker ~]\$ chmod 777 OracleDBData
3. [holuser@docker ~]\$ docker run -d -it --name plugpdb -p 1521:1521 -p 5500:5500 -e ORACLE_SID=DEVOPSUATCDB -v /home/holuser/OracleDBData:/opt/oracle/oradata oracle/database:12.2.0.1-ee
4. 3c9e0b92c7578b316097c607391bbd51373dece2594a9946841f34bd41264939
5. [holuser@docker ~]\$ ls -l OracleDBData
6. total 0
7. drwxr-xr-x 1 54321 54321 24 Jan 16 12:50 dbconfig
8. drwxr-x--- 1 54321 54321 258 Jan 16 12:45 DEVOPSUATCDB
9. [holuser@docker ~]\$ docker logs plugpdb | grep PASSWORD
10. ORACLE PASSWORD FOR SYS, SYSTEM AND PDBADMIN: 3NXIZQ+vJRY=1
11. [holuser@docker ~]\$

Use Sqlplus to create a new user 'test', a new table 'testname' for that user and insert some data into the table. We will then stop the container and start a new database pointing to the same volume on the host to see that data has reused in the new database container. If we hadn't externalized persistence for the database data would have been lost!

1. [holuser@docker ~]\$ docker exec -it plugpdb bash -c "source /home/oracle/.bashrc; sqlplus /nolog"
- 2.
3. SQL*Plus: Release 12.2.0.1.0 Production on Tue Jan 16 21:27:09 2018
- 4.
5. Copyright (c) 1982, 2016, Oracle. All rights reserved.
- 6.
7. SQL> connect sys/3NXIZQ+vJRY=1@orclpdb1 as sysdba
8. Connected.
9. SQL> create user test identified by test;
- 10.
11. User created.
- 12.
13. SQL> grant dba to test;
- 14.
15. Grant succeeded.
- 16.
17. SQL> connect test/test@orclpdb1
18. Connected.
19. SQL> create table testname (name varchar2(100));
- 20.
21. Table created.
- 22.
23. SQL> insert into testname values ('karsten');
- 24.
25. 1 row created.
- 26.
27. SQL> insert into testname values ('john');
- 28.
29. 1 row created.
- 30.

```

31. SQL> commit;
32.
33. Commit complete.
34.
35. SQL> exit
36. Disconnected from Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production
37. [holuser@docker ~]$ docker ps
38. CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
39. 7e27730f320d        oracle/database:12.2.0.1-ee  "/bin/sh -c 'exec ..."  15 minutes ago      Up 15 minutes (healthy)
    0.0.0.0:1521->1521/tcp, 0.0.0.0:5500->5500/tcp  plugpdb
40. [holuser@docker ~]$ docker container stop 7e27730f320d
41. 7e27730f320d
42. [holuser@docker ~]$ docker ps
43. CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
44. [holuser@docker ~]$ docker ps --all
45. CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
46. 7e27730f320d        oracle/database:12.2.0.1-ee  "/bin/sh -c 'exec ..."  16 minutes ago      Exited (137) 9 seconds ago
    plugpdb
47. [holuser@docker ~]$ docker container rm 7e27730f320d
48. 7e27730f320d
49. [holuser@docker ~]$ docker run -d -it --name mydb -p 1521:1521 -p 5500:5500 -e
    ORACLE_SID=DEVOPSUATCDB -v /home/holuser/OracleDBData:/opt/oracle/oradata oracle/database:12.2.0.1-ee
50. 862dfed3168bdc2af47d12bfd88f594631de50082599861c862a3a6fdc1bd303
51. [holuser@docker ~]$ docker logs mydb
52.
53. LSNRCTL for Linux: Version 12.2.0.1.0 - Production on 16-JAN-2018 21:32:29
54.
55. Copyright (c) 1991, 2016, Oracle. All rights reserved.
56.
57. Starting /opt/oracle/product/12.2.0.1/dbhome_1/bin/tnslsnr: please wait...
58.
59. TNSLSNR for Linux: Version 12.2.0.1.0 - Production
60. System parameter file is /opt/oracle/product/12.2.0.1/dbhome_1/network/admin/listener.ora
61. Log messages written to /opt/oracle/diag/tnslsnr/862dfed3168b/listener/alert/log.xml
62. Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1)))
63. Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=0.0.0.0)(PORT=1521)))
64.
65. Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=EXTPROC1)))
66. STATUS of the LISTENER
67. -----
68. Alias                LISTENER
69. Version               TNSLSNR for Linux: Version 12.2.0.1.0 - Production
70. Start Date            16-JAN-2018 21:32:29
71. Uptime                0 days 0 hr. 0 min. 0 sec
72. Trace Level           off
73. Security              ON: Local OS Authentication
74. SNMP                 OFF
75. Listener Parameter File /opt/oracle/product/12.2.0.1/dbhome_1/network/admin/listener.ora
76. Listener Log File     /opt/oracle/diag/tnslsnr/862dfed3168b/listener/alert/log.xml
77. Listening Endpoints Summary...
78. (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1)))

```

```
79. (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=0.0.0.0)(PORT=1521)))
80. The listener supports no services
81. The command completed successfully
82.
83. SQL*Plus: Release 12.2.0.1.0 Production on Tue Jan 16 21:32:29 2018
84.
85. Copyright (c) 1982, 2016, Oracle. All rights reserved.
86.
87. Connected to an idle instance.
88.
89. SQL> ORACLE instance started.
90.
91. Total System Global Area 1610612736 bytes
92. Fixed Size 8793304 bytes
93. Variable Size 520094504 bytes
94. Database Buffers 1073741824 bytes
95. Redo Buffers 7983104 bytes
96. Database mounted.
97. Database opened.
98. SQL> Disconnected from Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production
99. The Oracle base remains unchanged with value /opt/oracle
100. #####
101. DATABASE IS READY TO USE!
102. #####
103. The following output is now a tail of the alert.log:
104. ORCLPDB1(3):Undo initialization finished serial:0 start:289516304 end:289516391 diff:87 ms (0.1 seconds)
105. ORCLPDB1(3):Database Characterset for ORCLPDB1 is AL32UTF8
106. 2018-01-16T21:32:46.679322+00:00
107. ORCLPDB1(3):Opatch validation is skipped for PDB ORCLPDB1 (con_id=0)
108. ORCLPDB1(3):Opening pdb with no Resource Manager plan active
109. Pluggable database ORCLPDB1 opened read write
110. Starting background process CJQ0
111. 2018-01-16T21:32:47.500240+00:00
112. CJQ0 started with pid=40, OS id=289
113. Completed: ALTER DATABASE OPEN
114. 2018-01-16T21:32:48.615012+00:00
115. Shared IO Pool defaulting to 64MB. Trying to get it from Buffer Cache for process 88.
116. =====
117. Dumping current patch information
118. =====
119. No patches have been applied
120. =====
121. [holuser@docker ~]$docker exec -it mydb bash -c "source /home/oracle/.bashrc; sqlplus /nolog"
122.
123. SQL*Plus: Release 12.2.0.1.0 Production on Tue Jan 16 21:35:51 2018
124.
125. Copyright (c) 1982, 2016, Oracle. All rights reserved.
126.
127. SQL> connect test/test@orclpdb1
128. Connected.
129. SQL> select * from testname;
130.
131. NAME
132. -----
```



```
133. karsten
134. john
135.
136. SQL>
```

Data has been reused ☺

This concludes the lab. Before proceeding to the next lab make sure to stop and remove all running containers by usage of ***docker container rm -force <container_id> <container_id>.....***

You can get <container_id> from ***docker ps --all***

Using the official Docker Certified Oracle Weblogic image

The following tags are available for this image:

- store/oracle/weblogic:12.2.1.3
- store/oracle/weblogic:12.2.1.2

The documentation below uses 12.2.1.3 which is the latest version available. You may choose to modify the commands to use 12.2.1.2 instead.

Get Started

To create an empty domain with an Admin Server running, you simply call

```
1. [holuser@docker ~]$docker run -d store/oracle/weblogic:12.2.1.3
```

The WebLogic Server image will invoke createAndStartEmptyDomain.sh as the default CMD, and the Admin Server will be running on port 7001. When running multiple containers map port 7001 to a different port on the host:

```
1. [holuser@docker ~]$docker run -d -p 7001:7001 store/oracle/weblogic:12.2.1.3
2. 21c7164753ce732b6cfd3e3b9f7a250536181388b60086337b1ee287e6cdb434
3. [holuser@docker ~]
```

To run a second container on port 7002:

```
1. [holuser@docker ~]$docker run -d -p 7002:7001 store/oracle/weblogic:12.2.1.3
2. 2f073e3e3ce89d6ed7a72154165a75609474829453125b4211c1694e361b2024
3. [holuser@docker ~]$
```

Now you can access the AdminServer Web Console at <http://localhost:7001/console> or at <http://localhost:7002/console>.

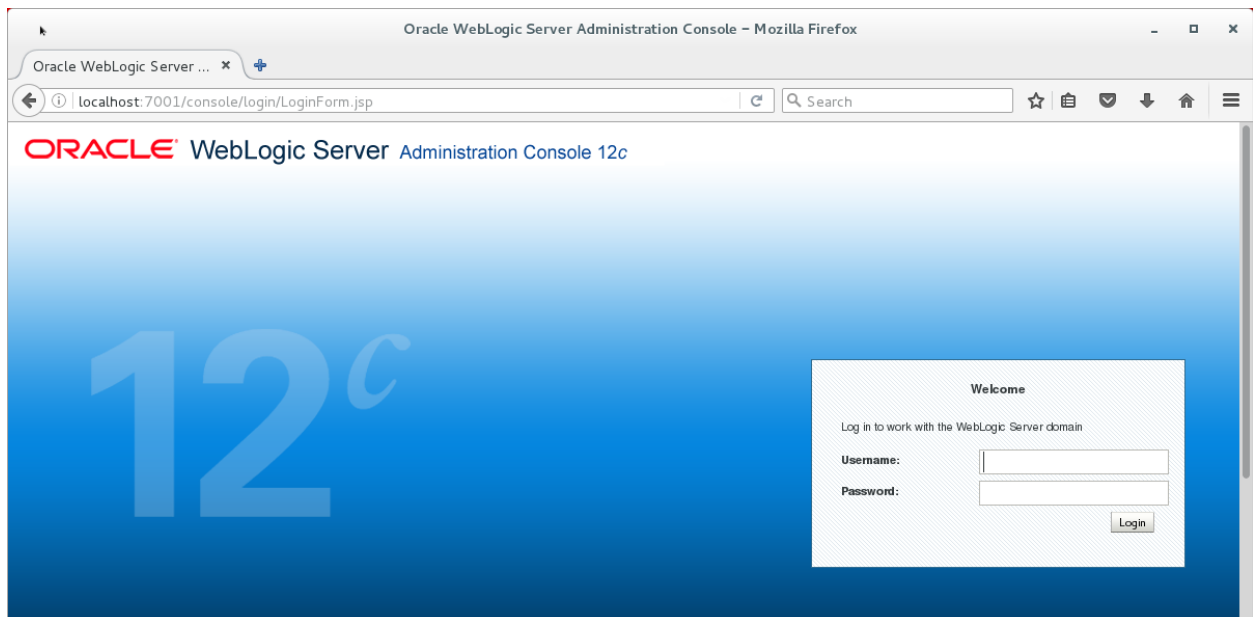
Administration Password

Before you can log into the Weblogic Console you need to obtain the administration password. On the first startup of the container a random password will be generated for the Administration of the domain. You can find this password in the output line, Oracle WebLogic Server auto generated Admin password.

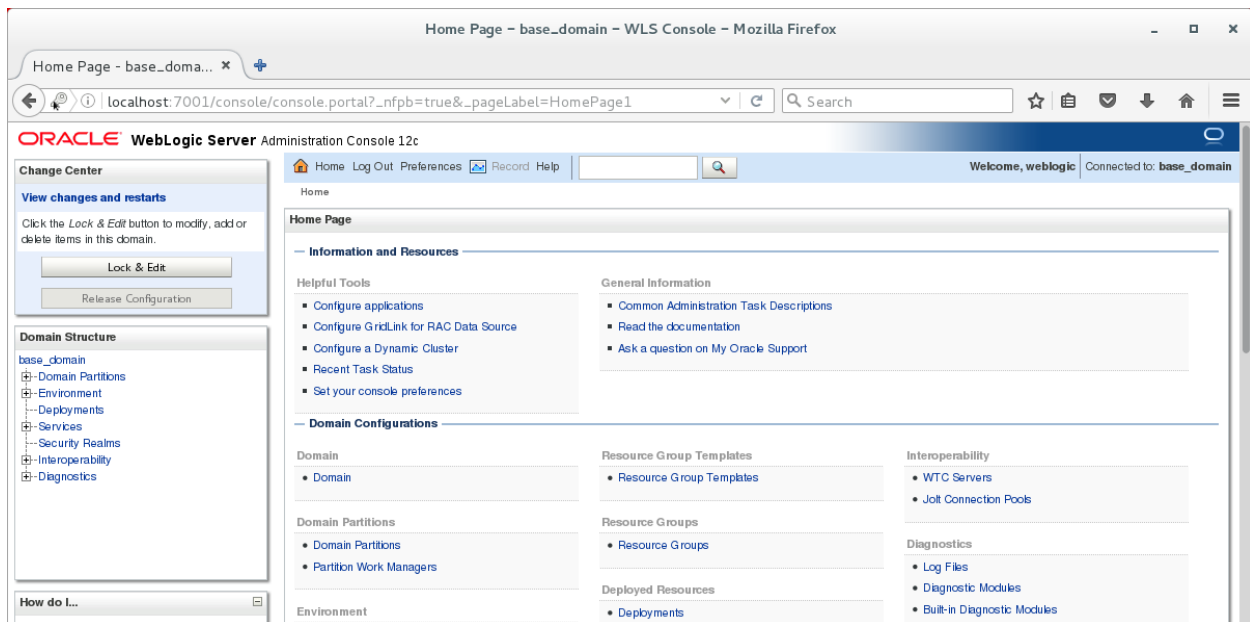
If you need to find the password at a later time, grep for password in the Docker logs generated during the startup of the container, run command

```
1. [holuser@docker ~]$ docker ps
2. CONTAINER ID   IMAGE                                COMMAND                  CREATED    STATUS    PORTS
   NAMES
3. 2f073e3e3ce8   store/oracle/weblogic:12.2.1.3    "/u01/oracle/creat..." 3 minutes ago    Up 3 minutes
   0.0.0.0:7002->7001/tcp   keen_johnson
4. 21c7164753ce   store/oracle/weblogic:12.2.1.3    "/u01/oracle/creat..." 3 minutes ago    Up 3 minutes
   0.0.0.0:7001->7001/tcp   quizzical_saha
5. [holuser@docker ~]$ docker logs 2f073e3e3ce8 | grep password
6. ----> 'weblogic' admin password: Ov6wpda5
7. admin password : [Ov6wpda5]
8. * password assigned to an admin-level user. For *
9. [holuser@docker ~]$
```

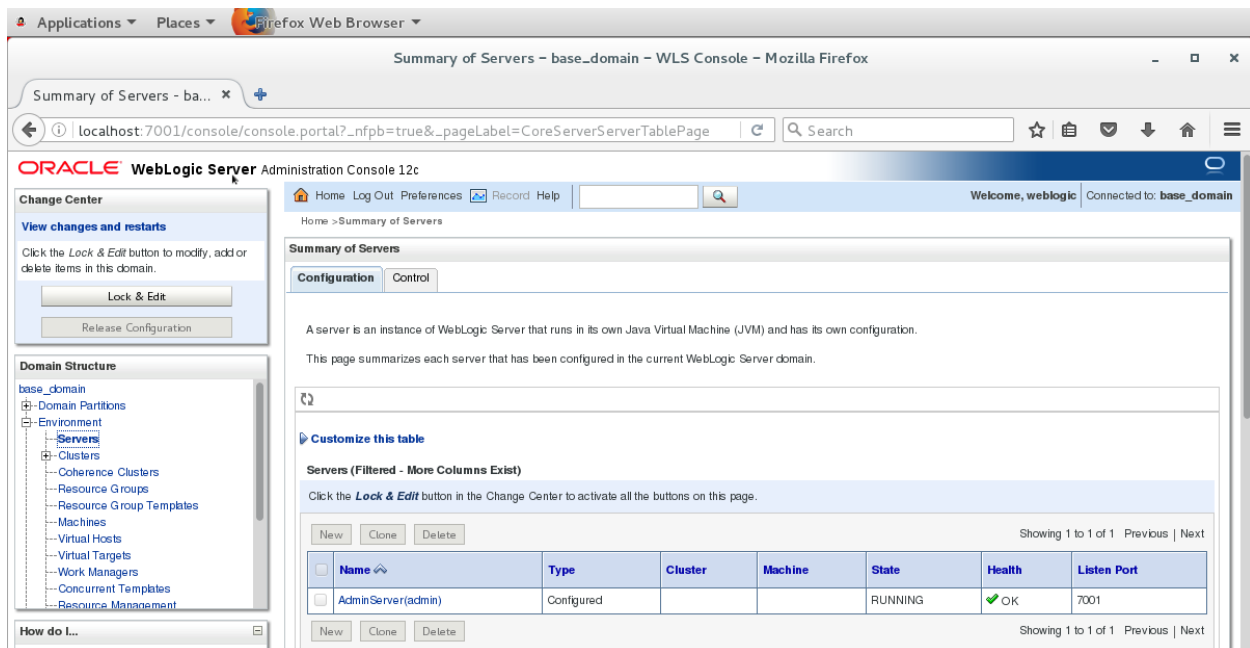
So now we have the administration password start Firefox and access the Weblogic Console – <http://localhost:7001/console> :



Login with 'Weblogic' and the just found password:



Go to 'Environment'-'>'Servers' to confirm that base_domain consists of just the AdminServer.



Customize your own WebLogic Server Domain

You might want to customize your own WebLogic Server domain by extending this image. The best way to create your own domain is by writing your own Dockerfiles, and using WebLogic Scripting Tool (WLST) to create clusters, Data Sources, JMS Servers, Security Realms, and deploy applications.

In your Dockerfile you will extend the WebLogic Server image with the **FROM store/oracle/weblogic:12.2.1.3** directive.

We provide a variety of examples (Dockerfiles, shell scripts, and WLST scripts) to create domains, configure resources, deploy applications, and use load balancer in the Oracle Docker Images GitHub repository.

This concludes the lab. Before proceeding to the next lab make sure to stop and remove all running containers by usage of ***docker container rm -force <container_id> <container_id>.....***

You can get <container_id> from ***docker ps --all***

How to build your own base Oracle Weblogic Docker Image

We have now seen how we can use the pre-build official Oracle Weblogic Docker Image to very fast and in an easy way to kick off an Oracle Weblogic installation with an empty domain running in a Docker Container. But if you some reason need to build your own Oracle Weblogic Docker Image it's simple to do so.

In the following we will build an Oracle Weblogic 12.2.1.3 Docker Image from scratch or more precise from the store/oracle/serverjre:8 Docker Image. We will be using a Docker File containing all the necessary steps to install the Oracle Weblogic 12.2.1.3 Server. The resulting images will like the pre-build oracle Weblogic Docker Image have an empty domain (only Admin Server) by default. You must extend the image with your own Dockerfile, and create your domain using WLST. We will be doing this in a subsequent section of this document....

Open a Terminal and go to the **/u01/OracleWeblogic/dockerfiles/12.2.1.3** directory.

The certification of Oracle WebLogic Server on Docker does not require the use of any file presented in this repository. Customers and users are welcome to use them as starters, and customize/tweak, or create from scratch new scripts and Dockerfiles.

For more information on the certification, please check the [Oracle WebLogic Server on Docker Certification Whitepaper](#) and [WebLogic Server Blog](#) for updates.

The image oracle/weblogic:12.2.1.3-developer will configure a base_domain with the following settings:

- Admin Username: weblogic
- Admin Password: Auto generated
- Oracle Linux Username: oracle
- Oracle Linux Password: welcome1
- WebLogic Server Domain Name: base_domain
- Admin Server on port: 7001
- Production Mode: developer

IMPORTANT: If you intend to run these images in production you must change the Production Mode to production.

###Admin Password

On the first startup of the container a random password will be generated for the Administration of the domain. You can find this password in the output line:

Oracle WebLogic Server auto generated Admin password:

If you need to find the password at a later time, grep for "password" in the Docker logs generated during the startup of the container. To look at the Docker Container logs run:

```
$ docker logs --details <Container-id> | grep password
```

Let's get started with the Docker Image creation. In the terminal execute the following:

```
1. [holuser@docker 12.2.0.1]$ cd /u01/OracleWebLogic/dockerfiles/12.2.1.3/
2. [holuser@docker 12.2.1.3]$ ls -l
3. total 40
4. -rw-r--r-- 1 holuser holuser 226 Jan  4 13:03 Checksum.developer
5. -rw-r--r-- 1 holuser holuser 222 Jan  4 13:03 Checksum.generic
6. drwxr-xr-x 1 holuser holuser  96 Jan  4 13:03 container-scripts
7. -rw-r--r-- 1 holuser holuser 3391 Jan  4 13:03 Dockerfile.developer
8. -rw-r--r-- 1 holuser holuser 3405 Jan  4 13:03 Dockerfile.generic
9. -rw-r--r-- 1 holuser holuser 221 Jan  4 13:03 fmw_12.2.1.3.0_wls_Disk1_1of1.zip.download
10. -rw-r--r-- 1 holuser holuser 225 Jan  4 13:03 fmw_12.2.1.3.0_wls_quick_Disk1_1of1.zip.download
11. -rw-r--r-- 1 holuser holuser 225 Jan  4 13:03 install.file
12. -rw-r--r-- 1 holuser holuser  55 Jan  4 13:03 oraInst.loc
13. -rw-r--r-- 1 holuser holuser 7917 Jan  4 13:03 README.md
14. [holuser@docker 12.2.1.3]$
```

Open the **Dockerfile.developer** in your favorite editor (like **gedit** or **vi**) and familize yourself with the content.

```
15. #Copyright (c) 2014-2017 Oracle and/or its affiliates. All rights reserved.
16. #
17. #Licensed under the Universal Permissive License v 1.0 as shown at http://oss.oracle.com/licenses/upl.
18. #
19. # ORACLE DOCKERFILES PROJECT
20. # -----
21. # This is the Dockerfile for WebLogic 12.2.1.3 Quick Install Distro
22. #
23. # REQUIRED FILES TO BUILD THIS IMAGE
24. # -----
25. # (1) fmw_12.2.1.3.0_wls_quick_Disk1_1of1.zip
26. #   Download the Developer Quick installer from
27. #   http://www.oracle.com/technetwork/middleware/weblogic/downloads/wls-for-dev-1703574.html
28. #
29. # (2) server-jre-8uXX-linux-x64.tar.gz
30. #   Download from http://www.oracle.com/technetwork/java/javase/downloads/server-jre8-downloads-2133154.html
31. #
32. # HOW TO BUILD THIS IMAGE
33. # -----
34. # Put all downloaded files in the same directory as this Dockerfile
35. # Run:
36. #   $ docker build -t oracle/weblogic:12.2.1.3-developer .
37. #
38. # IMPORTANT
39. # -----
```

```

39. # The resulting image of this Dockerfile contains a WLS Empty Domain.
40. #
41. # Pull base image
42. # From the Oracle Registry
43. # -----
44. FROM store/oracle/serverjre:8
45.
46. # Maintainer
47. # -----
48. MAINTAINER Monica Riccelli <monica.riccelli@oracle.com>
49.
50. # Common environment variables required for this build (do NOT change)
51. # -----
52. ENV ORACLE_HOME=/u01/oracle \
53.     USER_MEM_ARGS="-Djava.security.egd=file:/dev/./urandom" \
54.     SCRIPT_FILE=/u01/oracle/createAndStartEmptyDomain.sh \
55.     PATH=$PATH:/usr/java/default/bin:/u01/oracle/oracle_common/common/bin:/u01/oracle/wlserver/common/bin
56.
57. # Setup filesystem and oracle user
58. # Adjust file permissions, go to /u01 as user 'oracle' to proceed with WLS installation
59. # -----
60. RUN mkdir -p /u01 && \
61.     chmod a+rx /u01 && \
62.     useradd -b /u01 -d /u01/oracle -m -s /bin/bash oracle
63.
64. # Copy scripts
65. # -----
66. COPY container-scripts/createAndStartEmptyDomain.sh container-scripts/create-wls-domain.py /u01/oracle/
67.
68. # Domain and Server environment variables
69. # -----
70. ENV DOMAIN_NAME="${DOMAIN_NAME:-base_domain}" \
71.     DOMAIN_HOME=/u01/oracle/user_projects/domains/${DOMAIN_NAME:-base_domain} \
72.     ADMIN_PORT="${ADMIN_PORT:-7001}" \
73.     ADMIN_USERNAME="${ADMIN_USERNAME:-weblogic}" \
74.     ADMIN_NAME="${ADMIN_NAME:-AdminServer}" \
75.     ADMIN_PASSWORD="${ADMIN_PASSWORD:-""}" \
76.     DEBUG_FLAG=true \
77.     PRODUCTION_MODE=dev
78.
79.
80.
81. # Environment variables required for this build (do NOT change)
82. # -----
83. ENV FMW_PKG=fmw_12.2.1.3.0_wls_quick_Disk1_1of1.zip \
84.     FMW_JAR=fmw_12.2.1.3.0_wls_quick.jar
85.
86. # Copy packages
87. # -----
88. COPY $FMW_PKG install.file oraInst.loc /u01/
89. RUN chown oracle:oracle -R /u01 && \
90.     chmod +rx $SCRIPT_FILE
91.
92. # Install

```

```

93. # -----
94. USER oracle
95. RUN cd /u01 && $JAVA_HOME/bin/jar xf /u01/$FMW_PKG && cd - && \
96.   $JAVA_HOME/bin/java -jar /u01/$FMW_JAR -invPtrLoc /u01/oraInst.loc -jreLoc $JAVA_HOME -
   ignoreSysPrereqs -force -novalidation ORACLE_HOME=$ORACLE_HOME && \
97.   rm /u01/$FMW_JAR /u01/$FMW_PKG /u01/oraInst.loc /u01/install.file
98.
99. WORKDIR ${ORACLE_HOME}
100.
101. # Define default command to start script.
102. CMD ["/u01/oracle/createAndStartEmptyDomain.sh"]

```

At this point in time you should be able to read the Docker file now. If you are in doubt of anything please contact your instructor ☺ Note that the empty domain is created at container start.

Copy the quick installation zip to **/u01/OracleWeblogic/dockerfiles/12.2.1.3** directory.

```

1. [holuser@docker 12.2.1.3]$ cp /u01/software/fmw_12.2.1.3.0_wls_quick_Disk1_1of1.zip .
2. [holuser@docker 12.2.1.3]$

```

And then run the Docker build command to build the image:

```

1. [holuser@docker 12.2.1.3]$ docker login
2. Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to
   https://hub.docker.com to create one.
3. Username (kstniels):
4. Password:
5. Login Succeeded!
6. [holuser@docker 12.2.1.3]$ docker build --force-rm=true --no-cache=true -t oracle/weblogic:12.2.1.3-developer -f
   Dockerfile.developer .
7. Sending build context to Docker daemon 242.2MB
8. Step 1/13 : FROM store/oracle/serveryre:8
9. 8: Pulling from store/oracle/serveryre
10. 497341ef9d71: Pull complete
11. 8eae6d607524: Pull complete
12. Digest: sha256:315306e85195315afd4f855a32cd9a0af6040d69e9ba84a64133934964e4d683
13. Status: Downloaded newer image for store/oracle/serveryre:8
14. ---> c5e638a9290e
15. Step 2/13 : MAINTAINER Monica Riccelli <monica.riccelli@oracle.com>
16. ---> Running in f2364afe7ca5
17. ....
18. ....
19. ....
20. Removing intermediate container 0de71d4555be
21. Step 12/13 : WORKDIR ${ORACLE_HOME}
22. ---> ac55da9e2309
23. Removing intermediate container 710e272726b3
24. Step 13/13 : CMD /u01/oracle/createAndStartEmptyDomain.sh
25. ---> Running in 547f3cae2d6a
26. ---> 0a6c5cc6af48
27. Removing intermediate container 547f3cae2d6a

```

28. Successfully built 0a6c5cc6af48
29. Successfully tagged oracle/weblogic:12.2.1.3-developer
30. [holuser@docker 12.2.1.3]\$

Please note – it will take some time to build the image – so please be patient ☺

1. [holuser@docker 12.2.1.3]\$ docker images
2.

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
oracle/weblogic	12.2.1.3-developer	0a6c5cc6af48	43 minutes ago	1.43GB
oracle/database	12.2.0.1-ee2	95b1c2533f11	28 hours ago	13.3GB
oracle/database	12.2.0.1-ee	8d99b7bc4f06	29 hours ago	13.3GB
store/oracle/weblogic	12.2.1.3	1fcee95cc21b	4 weeks ago	1.14GB
oraclelinux	latest	f426f15a5793	6 weeks ago	229MB
oraclelinux	7-slim	9870bebf1d5	6 weeks ago	118MB
oraclelinux	6	63f04a5ae058	2 months ago	171MB
mysql/mysql-server	latest	a3ee341faefb	2 months ago	246MB
store/oracle/database-enterprise	12.2.0.1	12a359cd0528	4 months ago	3.44GB
store/oracle/serverjre	8	c5e638a9290e	7 months ago	274MB
13. [holuser@docker 12.2.1.3]\$
- 14.

After the build finishes successfully you run it by executing the following:

1. [holuser@docker 12.2.1.3]\$ docker run -d -it --name MyCustomWLS oracle/weblogic:12.2.1.3-developer
2. f6cb5be870d563265e0b3217b2866b4124a6454b697cca35a03e03266a95dd11
3. [holuser@docker 12.2.1.3]\$ docker ps
4.

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
f6cb5be870d5	oracle/weblogic:12.2.1.3-developer	"/u01/oracle/creat..."	24 seconds ago	Up 23 seconds
5. MyCustomWLS
6. [holuser@docker 12.2.1.3]\$

To start the console for the running Weblogic Docker Image you will need to retrieve the ip address of the Docker Container as we did not map the container exposed ports to the host by usage of **‘-p’** or **‘-P’**. You can find the IP address for the **MyCustomWLS** container you just started by executing the following:

1. [holuser@docker 12.2.1.3]\$ docker ps
2.

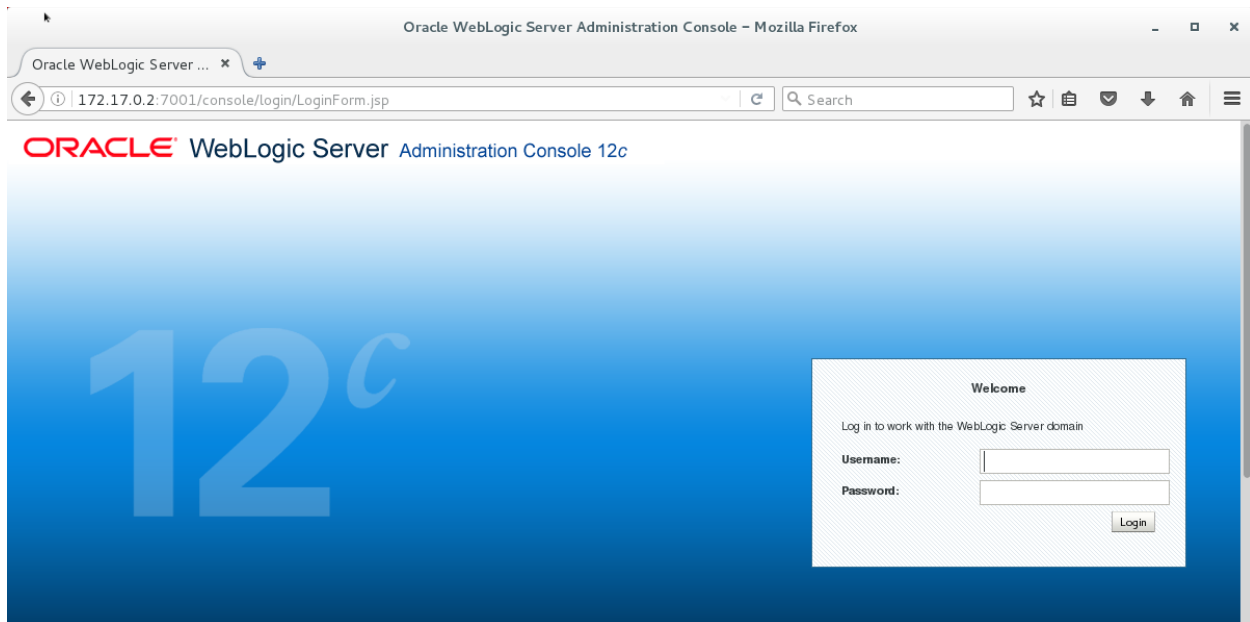
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
f6cb5be870d5	oracle/weblogic:12.2.1.3-developer	"/u01/oracle/creat..."	15 minutes ago	Up 15 minutes
3. MyCustomWLS
4. [holuser@docker 12.2.1.3]\$ docker inspect --format '{{.NetworkSettings.IPAddress}}' f6cb5be870d5
5. 172.17.0.2
6. [holuser@docker 12.2.1.3]\$

To get the weblogic user password you execute the following:

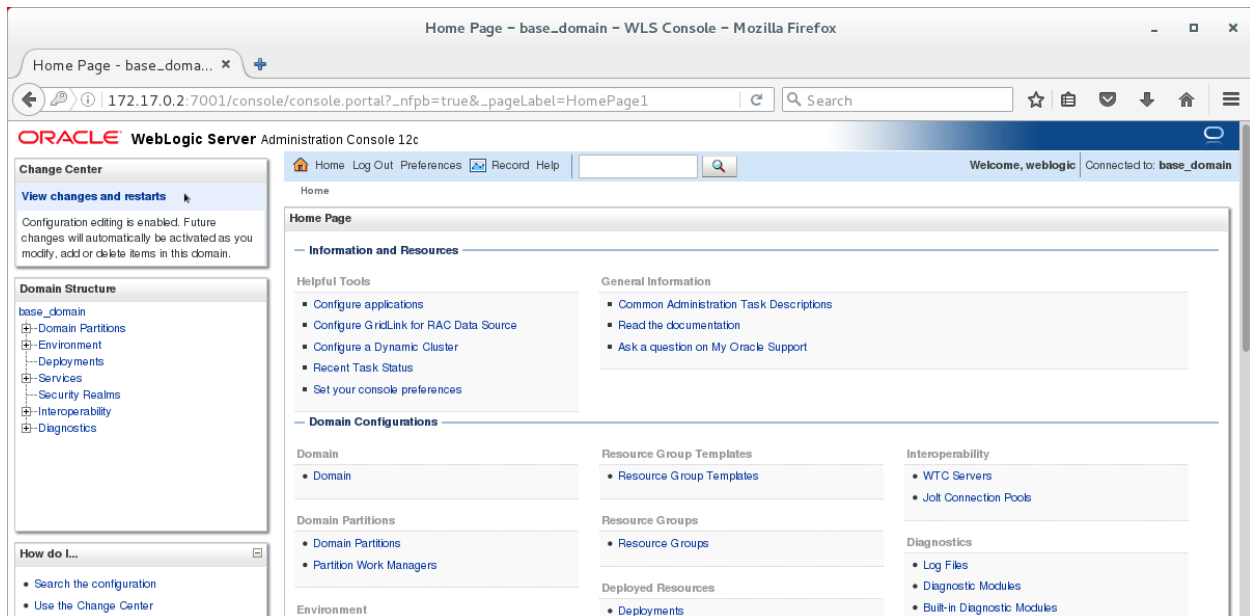
1. [holuser@docker 12.2.1.3]\$ docker logs f6cb5be870d5 | grep password

2. ----> 'weblogic' admin password: 5A63rMXg
3. admin password : [5A63rMXg]
4. * password assigned to an admin-level user. For *
5. [holuser@docker 12.2.1.3]\$

Now we can access and login into the Weblogic Console for the **MyCustomWLS** container. Start Firefox and enter <http://172.17.0.2/console> - this start the Weblogic Console.



Log in with *weblogic*/*<pwd you found>* and you will get:



When running the Docker Image you can override the default values of the following parameters during runtime with the `-e` option:

- ADMIN_NAME (default: AdminServer)
- ADMIN_PORT (default: 7001)
- ADMIN_USERNAME (default: weblogic)
- ADMIN_PASSWORD (default: Auto Generated)
- DOMAIN_NAME (default: base_domain)
- DOMAIN_HOME (default: /u01/oracle/user_projects/domains/base_domain)

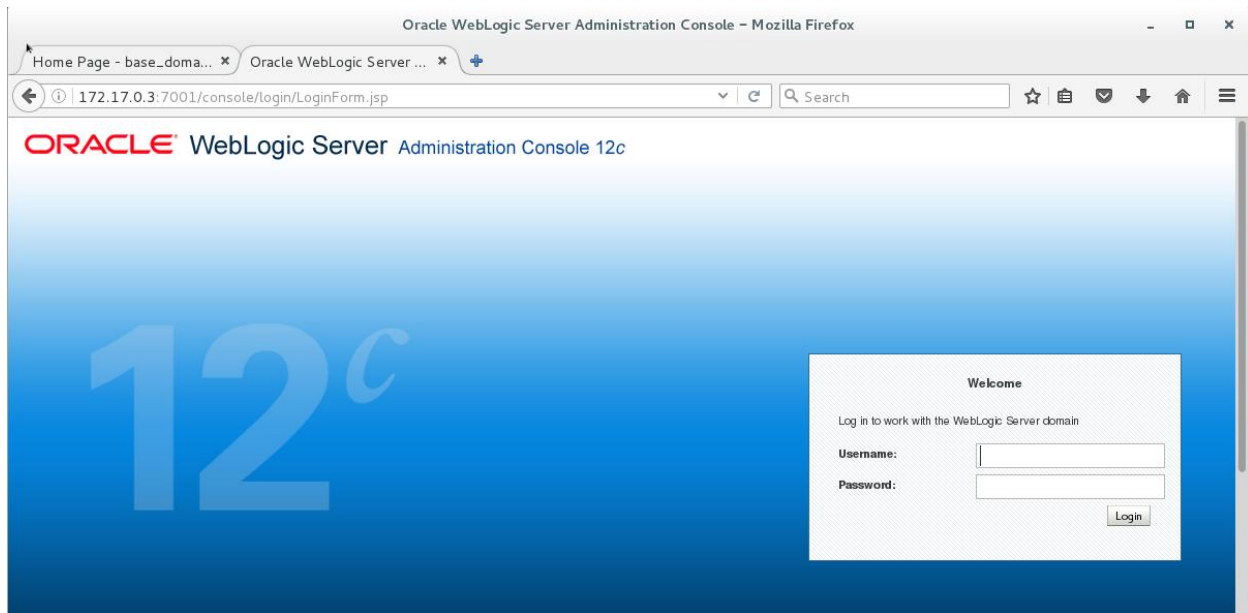
NOTE To set the DOMAIN_NAME, you must set both DOMAIN_NAME and DOMAIN_HOME.

1. [holuser@docker 12.2.1.3]\$ docker run -d -it --name MyCustomWLS1 -e ADMIN_USERNAME=weblogic -e ADMIN_PASSWORD=welcome1 -e DOMAIN_HOME=/u01/oracle/user_projects/domains/abc_domain -e DOMAIN_NAME=abc_domain oracle/weblogic:12.2.1.3-developer
2. b06db0f6500c0d2267e9b1c3d26ff7443d2155813b1f132a6e66a2d5bd3a4625
3. [holuser@docker 12.2.1.3]\$ docker ps
4.

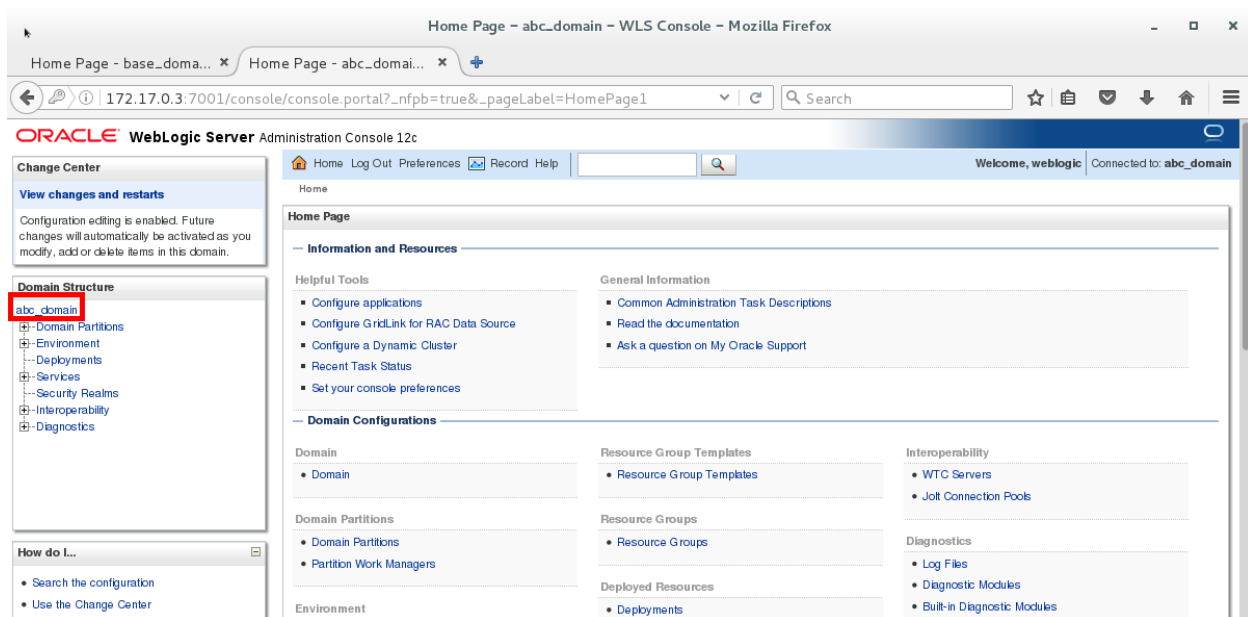
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
b06db0f6500c	oracle/weblogic:12.2.1.3-developer	"/u01/oracle/creat..."	18 seconds ago	Up 17 seconds
5. MyCustomWLS1
6. 1f387349a886 oracle/weblogic:12.2.1.3-developer "/u01/oracle/creat..." 8 minutes ago Up 8 minutes MyCustomWLS
7. [holuser@docker 12.2.1.3]\$

Find the IP address of the MyCustomWLS1 container, the administration password (should hopefully be 'welcome1' 😊) and open the Weblogic Console to check that the customization is correct:

1. [holuser@docker 12.2.1.3]\$ docker logs MyCustomWLS1 | grep password
2. ----> 'weblogic' admin password: welcome1
3. admin password : [welcome1]
4. * password assigned to an admin-level user. For *
5. [holuser@docker 12.2.1.3]\$ docker inspect --format '{{.NetworkSettings.IPAddress}}' MyCustomWLS1
6. 172.17.0.3
7. [holuser@docker 12.2.1.3]\$



Log in with weblogic/welcome1 and check domain name:



This project hosts two to three configurations (depending on Oracle WebLogic Server version) for building Docker images with WebLogic Server 12c.

- Quick Install Developer Distribution
 - For more information on the Oracle WebLogic Server 12cR2 Quick Install Developer Distribution, visit [WLS Quick Install Distribution for Oracle WebLogic Server 12.2.1.3.0](#).
- Generic Distribution
 - For more information on the Oracle WebLogic Server 12cR2 Generic Full Distribution, visit [WebLogic Server 12.2.1.3 Documentation](#).
- Fusion Middleware Infrastructure Distribution

- For more information on the Oracle WebLogic Server 12cR2 Infrastructure Full Distribution, visit [WebLogic Server 12.2.1.3 Infrastructure Documentation](#).

This concludes the *‘How to build your own base Weblogic Server Docker Image’* lab. Before proceeding to the next lab make sure to stop and remove all running containers by usage of *docker container rm -force <container_id> <container_id>.....*

You can get <container_id> from *docker ps --all*

How to extend the Docker Weblogic Image

Now we will extend the Oracle WebLogic image by creating a sample WLS 12.2.1.3 domain and cluster. Each managed server in the cluster will run in its own Docker container.

Util scripts are copied into the image enabling users to plug NodeManager automatically into the AdminServer running on another container.

The following assumes *oracle/weblogic:12.2.1.3-developer* has been build to the Docker Registry.

Open a terminal and go to the **/u01/OracleWeblogic/samples/12213-domain** directory.

```
1. [holuser@docker 12.2.1.3]$ cd /u01/OracleWebLogic/samples/12213-domain/
2. [holuser@docker 12213-domain]$ ls -l
3. total 12
4. -rwxr-xr-x 1 holuser holuser 227 Jan  4 13:03 build.sh
5. drwxr-xr-x 1 holuser holuser 320 Jan  4 13:03 container-scripts
6. -rw-r--r-- 1 holuser holuser 2435 Jan  4 13:03 Dockerfile
7. -rw-r--r-- 1 holuser holuser 3709 Jan  4 13:03 README.md
8. [holuser@docker 12213-domain]$
```

Open the **Dockerfile** in your favorite editor (like **gedit** or **vi**) and familize yourself with the content.

```
1. #Copyright (c) 2014-2017 Oracle and/or its affiliates. All rights reserved.
2. #
3. #Licensed under the Universal Permissive License v 1.0 as shown at http://oss.oracle.com/licenses/upl.
4. #
5. # ORACLE DOCKERFILES PROJECT
6. # -----
7. # This Dockerfile extends the Oracle WebLogic image by creating a sample domain.
8. #
9. # Util scripts are copied into the image enabling users to plug NodeManager
10. # automatically into the AdminServer running on another container.
11. #
12. # HOW TO BUILD THIS IMAGE
13. # -----
14. # Put all downloaded files in the same directory as this Dockerfile
```

```

15. # Run:
16. #   $ sudo docker build -t 12213-domain
17. #
18. # Pull base image
19. # -----
20. FROM oracle/weblogic:12.2.1.3-developer
21.
22. # Maintainer
23. # -----
24. MAINTAINER Monica Riccelli <monica.riccelli@oracle.com>
25.
26. # WLS Configuration
27. # -----
28. ENV ADMIN_HOST="wlsadmin" \
29.     NM_PORT="5556" \
30.     MS_PORT="8001" \
31.     DEBUG_PORT="8453" \
32.     ORACLE_HOME=/u01/oracle \
33.     SCRIPT_FILE=/u01/oracle/createAndStartWLSDomain.sh \
34.     CONFIG_JVM_ARGS="-Dweblogic.security.SSL.ignoreHostnameVerification=true" \
35.
36.     PATH=$PATH:/u01/oracle/oracle_common/common/bin:/u01/oracle/wlserver/common/bin:/u01/oracle/user_projects/
37.     domains/${DOMAIN_NAME:-base_domain}/bin:/u01/oracle
38.
39. # Domain and Server environment variables
40. # -----
41. ENV DOMAIN_NAME="${DOMAIN_NAME:-base_domain}" \
42.     PRE_DOMAIN_HOME=/u01/oracle/user_projects \
43.     ADMIN_PORT="${ADMIN_PORT:-7001}" \
44.     ADMIN_USERNAME="${ADMIN_USERNAME:-weblogic}" \
45.     ADMIN_NAME="${ADMIN_NAME:-AdminServer}" \
46.     MS_NAME="${MS_NAME:-""}" \
47.     NM_NAME="${NM_NAME:-""}" \
48.     ADMIN_PASSWORD="${ADMIN_PASSWORD:-""}" \
49.     CLUSTER_NAME="${CLUSTER_NAME:-DockerCluster}" \
50.     DEBUG_FLAG=true \
51.     PRODUCTION_MODE=dev
52.
53. # Add files required to build this image
54. COPY container-scripts/* /u01/oracle/
55.
56. #Create directory where domain will be written to
57. USER root
58. RUN chmod +xw /u01/oracle/*.sh && \
59.     chmod +xw /u01/oracle/*.py && \
60.     mkdir -p $PRE_DOMAIN_HOME && \
61.     chmod a+rx $PRE_DOMAIN_HOME && \
62.     chown -R oracle:oracle $PRE_DOMAIN_HOME
63.
64. VOLUME $PRE_DOMAIN_HOME
65.
66. # Expose Node Manager default port, and also default for admin and managed server
67. EXPOSE $NM_PORT $ADMIN_PORT $MS_PORT $DEBUG_PORT
68.
69. USER oracle

```

```

67. WORKDIR $ORACLE_HOME
68.
69. # Define default command to start bash.
70. CMD ["/u01/oracle/createAndStartWLSDomain.sh"]

```

As you can see that this Weblogic Docker Image is created from the **oracle/weblogic:12.2.1.3-developer** image. It exposes a number of ports:

- Node Manager Port
- Administration Port
- Managed Server Port
- Debug Port

It defines a new set of Environment variables with default values which can be overridden with the 'Docker run' command at time of instantiation:

```

1. ADMIN_HOST="wlsadmin" \
2. NM_PORT="5556" \
3. MS_PORT="8001" \
4. DEBUG_PORT="8453" \
5. ORACLE_HOME=/u01/oracle \
6. SCRIPT_FILE=/u01/oracle/createAndStartWLSDomain.sh \
7. CONFIG_JVM_ARGS="-Dweblogic.security.SSL.ignoreHostnameVerification=true" \
8. DOMAIN_NAME="${DOMAIN_NAME:-base_domain}" \
9. PRE_DOMAIN_HOME=/u01/oracle/user_projects \
10. ADMIN_PORT="${ADMIN_PORT:-7001}" \
11. ADMIN_USERNAME="${ADMIN_USERNAME:-weblogic}" \
12. ADMIN_NAME="${ADMIN_NAME:-AdminServer}" \
13. MS_NAME="${MS_NAME:-}" \
14. NM_NAME="${NM_NAME:-}" \
15. ADMIN_PASSWORD="${ADMIN_PASSWORD:-}" \
16. CLUSTER_NAME="${CLUSTER_NAME:-DockerCluster}" \
17. DEBUG_FLAG=true \
18. PRODUCTION_MODE=dev

```

You can define the following environment variables at docker runtime using the `-e` option in the command line or defining them in the domain.properties file. These environmental variables need to be set for the Admin Server as well as for the Managed Servers.

Besides that the Docker file defines a couple of new paths, copy the scripts **container-scripts/** to the image and defines a default CMD (which is executed upon instantiation/container start) for the image.

For the Cluster to work across different containers the domain directory needs to be externalized by using Data Volumes (`-v` option). The Admin Server as well as the Managed Servers need to read/write to the same DOMAIN_HOME. Here the AdminServer will host the volumes and act as a Data Container.

In order to work the steps are:

- The Admin Server container and all Managed Server containers are started from the above Docker Image
- The Admin Server is started with the default CMD on a user defined bridge network
- The Admin Server is started with the '`-v`' option and hence define the domain directory
- The Managed Servers are all started by overriding the default CMD with the createServer.sh script

- The Managed Servers are all started with the 'volumes-from' option referring to the Admin Server and all the volumes defines here and hence they can use the same domain directory
- The Managed Server are all started on the user defined bridge network hence the will all be able to communicate with each other

But first we need to build the Docker Image:

```

1. [holuser@docker 12213-domain]$ docker build --force-rm=true --no-cache=true -t 12213-domain .
2. Sending build context to Docker daemon 32.26kB
3. Step 1/12 : FROM oracle/weblogic:12.2.1.3-developer
4. ---> 0a6c5cc6af48
5. Step 2/12 : MAINTAINER Monica Riccelli <monica.riccelli@oracle.com>
6. ---> Running in 9ac2d26836b9
7. ---> 8f26c4f16b67
8. Removing intermediate container 9ac2d26836b9
9. Step 3/12 : ENV ADMIN_HOST "wlsadmin" NM_PORT "5556" MS_PORT "8001" DEBUG_PORT "8453"
    ORACLE_HOME /u01/oracle SCRIPT_FILE /u01/oracle/createAndStartWLSDomain.sh CONFIG_JVM_ARGS "-
    Dweblogic.security.SSL.ignoreHostnameVerification=true" PATH
    $PATH:/u01/oracle/oracle_common/common/bin:/u01/oracle/wlserver/common/bin:/u01/oracle/user_projects/domains
    /${DOMAIN_NAME:-base_domain}/bin:/u01/oracle
10. ---> Running in 04946ca8166f
11. ---> 305c2911990d
12. ....
13. ....
14. Step 12/12 : CMD /u01/oracle/createAndStartWLSDomain.sh
15. ---> Running in 65de9566af52
16. ---> adce6372091c
17. Removing intermediate container 65de9566af52
18. Successfully built adce6372091c
19. Successfully tagged 12213-domain:latest
20. [holuser@docker 12213-domain]$
21. [holuser@docker 12213-domain]$ docker images
22.
    REPOSITORY              TAG          IMAGE ID          CREATED          SIZE
23. 12213-domain             latest       adce6372091c     About a minute ago 1.43GB
24. oracle/weblogic          12.2.1.3-developer 0a6c5cc6af48     21 hours ago    1.43GB
25. oracle/database          12.2.0.1-ee2 95b1c2533f11     2 days ago      13.3GB
26. oracle/database          12.2.0.1-ee 8d99b7bc4f06     2 days ago      13.3GB
27. store/oracle/weblogic    12.2.1.3     1fcee95cc21b     5 weeks ago     1.14GB
28. oraclelinux              latest       f426f15a5793     6 weeks ago     229MB
29. oraclelinux              7-slim      9870bebf1d5      6 weeks ago     118MB
30. oraclelinux              6           63f04a5ae058     2 months ago    171MB
31. mysql/mysql-server       latest      a3ee341faefb     2 months ago    246MB
32. store/oracle/database-enterprise 12.2.0.1    12a359cd0528     4 months ago    3.44GB
33. store/oracle/optimizer   8           c5e638a9290e     7 months ago    274MB
34. [holuser@docker 12213-domain]$

```

Now we have built the image we need to start the Admin Server and the Managed Servers (we will start two managed servers in this session)

But first we will create the user defined bridge network so the managed servers and the AdminServer can access each other:

1. [holuser@docker 12213-domain]\$ docker network create --driver bridge isolated_nw
2. 248fe06859cd9603c19d03d23c9b3c8df100bb23046bdc2031f3de3382ea4600
3. [holuser@docker 12213-domain]

To start the containerized Admin Server, we create user_projects directory which we will map the shared volume to and start the container:

1. [holuser@docker 12213-domain]\$ mkdir user_projects
2. [holuser@docker 12213-domain]\$ chmod 777 user_projects/
3. [holuser@docker 12213-domain]\$ ls -l
4. total 12
5. -rwxr-xr-x 1 holuser holuser 227 Jan 4 13:03 build.sh
6. drwxr-xr-x 1 holuser holuser 320 Jan 4 13:03 container-scripts
7. -rw-r--r-- 1 holuser holuser 2435 Jan 4 13:03 Dockerfile
8. -rw-r--r-- 1 holuser holuser 3709 Jan 4 13:03 README.md
9. drwxrwxrwx 1 holuser holuser 0 Jan 18 07:07 user_projects
10. [holuser@docker 12213-domain]\$ docker run -d --name wlsadmin --hostname wlsadmin -p 7001:7001 --network=isolated_nw --env-file ./container-scripts/domain.properties -e ADMIN_PASSWORD=welcome1 -v /u01/OracleWebLogic/samples/12213-domain/user_projects:/u01/oracle/user_projects 12213-domain
11. 51071cc88229e524552fe23bc9528d45275b029d98b59c20d227897bf4b2127b
12. [holuser@docker 12213-domain]\$ docker ps
13. CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
14. 51071cc88229 12213-domain "/u01/oracle/creat..." 6 seconds ago Up 6 seconds 5556/tcp, 8001/tcp, 8453/tcp, 0.0.0.0:7001->7001/tcp wlsadmin
15. [holuser@docker 12213-domain]\$ docker logs 51071cc88229
16. ----> 'weblogic' admin password: welcome1
- 17.
18. Initializing WebLogic Scripting Tool (WLST) ...
- 19.
20. Welcome to WebLogic Server Administration Scripting Shell
- 21.
22. Type help() for help on available commands
- 23.
24. domain_name : [abc_domain]
25. admin name : [AdminServer]
26. admin_port : [7001]
27. cluster_name : [DockerCluster]
28. domain_path : [/u01/oracle/user_projects/domains/abc_domain]
29. production_mode : [dev]
30. [holuser@docker 12213-domain]\$ docker logs 51071cc88229
31. ----> 'weblogic' admin password: welcome1
- 32.
33. Initializing WebLogic Scripting Tool (WLST) ...
- 34.
35. Welcome to WebLogic Server Administration Scripting Shell
- 36.
37. Type help() for help on available commands
- 38.
39. domain_name : [abc_domain]
40. admin name : [AdminServer]


```

41. admin_port : [7001]
42. cluster_name : [DockerCluster]
43. domain_path : [/u01/oracle/user_projects/domains/abc_domain]
44. production_mode : [dev]
45. [holuser@docker 12213-domain]$ docker logs 51071cc88229
46. ----> 'weblogic' admin password: welcome1
47.
48. Initializing WebLogic Scripting Tool (WLST) ...
49.
50. Welcome to WebLogic Server Administration Scripting Shell
51.
52. Type help() for help on available commands
53.
54. domain_name : [abc_domain]
55. admin name : [AdminServer]
56. admin_port : [7001]
57. cluster_name : [DockerCluster]
58. domain_path : [/u01/oracle/user_projects/domains/abc_domain]
59. production_mode : [dev]
60.
61.
62. Exiting WebLogic Scripting Tool.
63.
64. .
65. .
66. JAVA Memory arguments: -Djava.security.egd=file:/dev/./urandom
67. .
68. CLASSPATH=/usr/java/jdk1.8.0_131/lib/tools.jar:/u01/oracle/wlserver/server/lib/weblogic.jar:/u01/oracle/wlserver/..o
   racle_common/modules/thirdparty/ant-contrib-
   1.0b3.jar:/u01/oracle/wlserver/modules/features/oracle.wls.common.nodemanager.jar:/u01/oracle/wlserver/common/de
   rby/lib/derbynet.jar:/u01/oracle/wlserver/common/derby/lib/derbyclient.jar:/u01/oracle/wlserver/common/derby/lib/der
   by.jar
69. .
70. PATH=/u01/oracle/user_projects/domains/abc_domain/bin:/u01/oracle/wlserver/server/bin:/u01/oracle/wlserver/..o
   racle_common/modules/thirdparty/org.apache.ant/1.9.8.0.0/apache-ant-
   1.9.8/bin:/usr/java/jdk1.8.0_131/jre/bin:/usr/java/jdk1.8.0_131/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/
   bin:/usr/java/default/bin:/u01/oracle/oracle_common/common/bin:/u01/oracle/wlserver/common/bin:/u01/oracle/oracle
   _common/common/bin:/u01/oracle/wlserver/common/bin:/u01/oracle/user_projects/domains/base_domain/bin:/u01/ora
   cle
71. .
72. *****
73. * To start WebLogic Server, use a username and *
74. * password assigned to an admin-level user. For *
75. * server administration, use the WebLogic Server *
76. * console at http://hostname:port/console *
77. *****
78. Starting WLS with line:
79. /usr/java/jdk1.8.0_131/bin/java -server -Djava.security.egd=file:/dev/./urandom -cp
   /u01/oracle/wlserver/server/lib/weblogic-launcher.jar -Dlaunch.use.env.classpath=true -
   Dweblogic.Name=AdminServer -Djava.security.policy=/u01/oracle/wlserver/server/lib/weblogic.policy -
   Djava.system.class.loader=com.oracle.classloader.weblogic.LaunchClassLoader -
   javaagent:/u01/oracle/wlserver/server/lib/debugpatch-agent.jar -da -Dwls.home=/u01/oracle/wlserver/server -
   Dweblogic.home=/u01/oracle/wlserver/server weblogic.Server

```

80. <Jan 12, 2018 10:46:24 AM GMT> <Info> <Security> <BEA-090905> <Disabling the CryptoJ JCE Provider self-integrity check for better startup performance. To enable this check, specify -Dweblogic.security.allowCryptoJDefaultJCEVerification=true.>
81. <Jan 12, 2018 10:46:24 AM GMT> <Info> <Security> <BEA-090906> <Changing the default Random Number Generator in RSA CryptoJ from ECDRBG128 to HMACDRBG. To disable this change, specify -Dweblogic.security.allowCryptoJDefaultPRNG=true.>
82. <Jan 12, 2018 10:46:25 AM GMT> <Info> <WebLogicServer> <BEA-000377> <Starting WebLogic Server with Java HotSpot(TM) 64-Bit Server VM Version 25.131-b11 from Oracle Corporation.>
83. <Jan 12, 2018 10:46:25 AM GMT> <Info> <RCM> <BEA-2165021> <"ResourceManagement" is not enabled in this JVM. Enable "ResourceManagement" to use the WebLogic Server "Resource Consumption Management" feature. To enable "ResourceManagement", you must specify the following JVM options in the WebLogic Server instance in which the JVM runs: -XX:+UnlockCommercialFeatures -XX:+ResourceManagement.>
84. <Jan 12, 2018 10:46:25 AM GMT> <Info> <Management> <BEA-141107> <Version: WebLogic Server 12.2.1.3.0 Thu Aug 17 13:39:49 PDT 2017 1882952>
85. [holuser@docker 12213-domain]\$

As you can see in the 'docker ps' the container exposes the 7001 port and maps it to 7001 on the host. That means you can start Mozilla/Firefox and enter the following url to access the Weblogic Console: <http://localhost:7001/console>. Username/Password is 'weblogic/welcome1'.

The screenshot shows the Oracle WebLogic Server Administration Console. The main content area is titled 'Summary of Servers' and shows a table of servers. The table has columns for Name, Type, Cluster, Machine, State, Health, and Listen Port. There is one server listed: 'AdminServer(admin)' with Type 'Configured', State 'RUNNING', Health 'OK', and Listen Port '7001'. The left sidebar contains a 'Domain Structure' tree and a 'System Status' section.

Name	Type	Cluster	Machine	State	Health	Listen Port
AdminServer(admin)	Configured			RUNNING	OK	7001

The console correctly shows that we have a Admin Server running in the 'abc_domain' Weblogic Domain. If you browse around in the console you will notice that a Weblogic Cluster named 'DockerCluster' has been created. Machine and Node Manager have too been created and configured.

To start a containerized Managed Server (MS1) to self-register with the Admin Server above, run:

1. [holuser@docker 12213-domain]\$ docker run -d --name MS1 --network=isolated_nw -p 8001:8001 --env-file ./container-scripts/domain.properties -e ADMIN_PASSWORD=welcome1 -e MS_NAME=MS1 --volumes-from wlsadmin 12213-domain createServer.sh
2. 8ee3bb0168eba97f08f928c03d5e873d047c705ef709ac170693449117d4f10b
3. [holuser@docker 12213-domain]\$ docker ps
4. CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
5. NAMES
6. 8ee3bb0168eb 12213-domain "createServer.sh" 7 seconds ago Up 6 seconds 5556/tcp, 7001/tcp, 8453/tcp, 0.0.0.0:8001->8001/tcp MS1
7. 51071cc88229 12213-domain "/u01/oracle/creat..." About an hour ago Up About an hour 5556/tcp, 8001/tcp, 8453/tcp, 0.0.0.0:7001->7001/tcp wlsadmin
7. [holuser@docker 12213-domain]\$

If you refresh the Weblogic console you will get:

The screenshot shows the Oracle WebLogic Server Administration Console. The main content area is titled 'Summary of Servers' and contains a table of servers. The table has columns for Name, Type, Cluster, Machine, State, Health, and Listen Port. There are two servers listed: AdminServer(admin) and MS1. Both are in a RUNNING state with a Health of OK. The MS1 server is part of the DockerCluster and has Machine_MS1. The console also shows a left sidebar with navigation options and a top navigation bar with user information.

Name	Type	Cluster	Machine	State	Health	Listen Port
AdminServer(admin)	Configured			RUNNING	OK	7001
MS1	Configured	DockerCluster	Machine_MS1	RUNNING	OK	8001

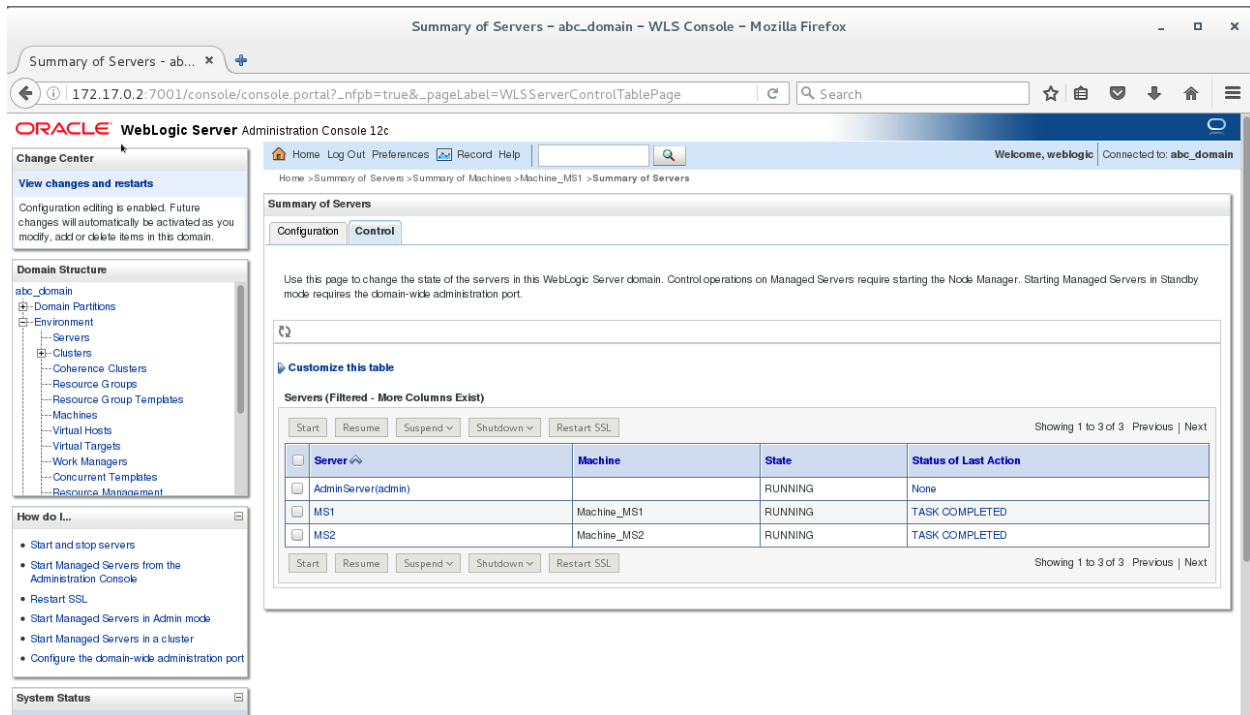
You have now two containers running: one hosting the Admin Server and domain volume and another hosting the Managed Server MS1. The Managed Server MS1 correctly belong to the Weblogic Cluster 'DockerCluster'. If you browse around in the Weblogic console you can the Node Manager is fully functional and that you can start and stop managed servers from the console.

To start a second Managed Server (MS2), run the following command:

1. [holuser@docker 12213-domain]\$ docker run -d --name MS2 --network=isolated_nw -p 8002:8001 --env-file ./container-scripts/domain.properties -e ADMIN_PASSWORD=welcome1 -e MS_NAME=MS2 --volumes-from wlsadmin 12213-domain createServer.sh
2. 737753b33e99f64c9f080cfdea3a1479b5b7762a36ebc57a6134dc98aa19a556
3. [holuser@docker 12213-domain]\$ docker ps
4. CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
5. NAMES

5. 737753b33e99 12213-domain "createServer.sh" 7 seconds ago Up 5 seconds 5556/tcp, 7001/tcp, 8453/tcp, 0.0.0.0:8002->8001/tcp MS2
6. 8ee3bb0168eb 12213-domain "createServer.sh" 33 minutes ago Up 33 minutes 5556/tcp, 7001/tcp, 8453/tcp, 0.0.0.0:8001->8001/tcp MS1
7. 51071cc88229 12213-domain "/u01/oracle/creat..." 2 hours ago Up 2 hours 5556/tcp, 8001/tcp, 8453/tcp, 0.0.0.0:7001->7001/tcp wlsadmin
8. [holuser@docker 12213-domain]\$

If you again refresh the console you will get:



If you inspect the 'isolated_nw' you will see that all containers are attached to the 'isolated_nw' and hence able to communicate with each other. Only ports 7001, 8001 and 8002 are opened in the user defined network.

1. [holuser@docker 12213-domain]\$ docker network inspect isolated_nw
2. [
3. {
4. "Name": "isolated_nw",
5. "Id": "248fe06859cd9603c19d03d23c9b3c8df100bb23046bdc2031f3de3382ea4600",
6. "Created": "2018-01-12T05:11:52.652351538-08:00",
7. "Scope": "local",
8. "Driver": "bridge",
9. "EnableIPv6": false,
10. "IPAM": {
11. "Driver": "default",
12. "Options": {},
13. "Config": [
14. {
15. "Subnet": "172.18.0.0/16",

```

16.         "Gateway": "172.18.0.1"
17.     }
18. ]
19. },
20. "Internal": false,
21. "Attachable": false,
22. "Ingress": false,
23. "ConfigFrom": {
24.     "Network": ""
25. },
26. "ConfigOnly": false,
27. "Containers": {
28.     "02d7a1f06b212a74fe1a2256ec3aa64cb5ed774b457df41b9cf80d74c8f81124": {
29.         "Name": "wlsadmin",
30.         "EndpointID": "f5bb089d984eb6217b8dc681a25778d094b699f9d9026ad22d612ace796f9872",
31.         "MacAddress": "02:42:ac:12:00:02",
32.         "IPv4Address": "172.18.0.2/16",
33.         "IPv6Address": ""
34.     },
35.     "67d7b99ced2ec535e254aead4badc041c01500b0e8498050a595ee245ed4941a": {
36.         "Name": "MS2",
37.         "EndpointID": "80607ce1c12d29354a66c301c693d01973ce9ba8b3564d3114dc803bb805d924",
38.         "MacAddress": "02:42:ac:12:00:04",
39.         "IPv4Address": "172.18.0.4/16",
40.         "IPv6Address": ""
41.     },
42.     "7d32be06f5fc835fccfe5fcc76f113663c62afd184f2a374ae59b8f92d62236": {
43.         "Name": "MS1",
44.         "EndpointID": "28e9c7d8f038724feee6067de9ebd73ef9aa96272d4f36689dead6be95b59e40",
45.         "MacAddress": "02:42:ac:12:00:03",
46.         "IPv4Address": "172.18.0.3/16",
47.         "IPv6Address": ""
48.     }
49. },
50. "Options": {},
51. "Labels": {}
52. }
53. ]
54. [holuser@docker 12213-domain]$

```

We now have a full operational Weblogic Cluster with two Managed Servers MS1 and MS2 running in different Docker Containers. The managed servers can be stopped and started through Weblogic Console.

The above scenario from this sample will give you a WebLogic domain with a cluster setup, on a single host environment.

You may create more containerized Managed Servers by calling the docker command above for createServer.sh as long you link properly with the Admin Server. For an example of multihost environment, check the sample 1221-multihost.

This concludes the lab. Before proceeding to the next lab make sure to stop and remove all running containers by usage of ***docker container rm -force <container_id> <container_id>.....***

You can get <container_id> from ***docker ps --all***

Connecting a Weblogic Container to an Oracle Database Container

In this session we will extend the Oracle WebLogic install image from the DockerStore to configure a WebLogic domain, create a DataSource that connects to a DB, and deploys a WLS application that does DB operations. We will also create a new Oracle Database Docker image that contains a prebuild Oracle Database with pre-populated schemas for the Weblogic Application. Remember the Oracle Database Docker images we have built both the custom and the Oracle prebuild Docker image first build the database instance upon Docker image start!

Open a terminal and go to the ***/u01/OracleWebLogic/samples/12212-oradb-wlsstore*** directory. First create a Docker Network which will establish the connection between The Weblogic container and the Oracle Database container:

```
1. [holuser@docker 12212-oradb-wlsstore]$ docker network create -d bridge SampleNET
2. a8f42b7288085837c2c7ac63b1e9f6550005123508711cb8df268f3e076f9ac4
3. [holuser@docker 12212-oradb-wlsstore]$ docker network inspect SampleNET
4. [
5.   {
6.     "Name": "SampleNET",
7.     "Id": "a8f42b7288085837c2c7ac63b1e9f6550005123508711cb8df268f3e076f9ac4",
8.     "Created": "2018-01-19T01:20:23.636424676-08:00",
9.     "Scope": "local",
10.    "Driver": "bridge",
11.    "EnableIPv6": false,
12.    "IPAM": {
13.      "Driver": "default",
14.      "Options": {},
15.      "Config": [
16.        {
17.          "Subnet": "172.19.0.0/16",
18.          "Gateway": "172.19.0.1"
19.        }
20.      ]
21.    },
22.    "Internal": false,
23.    "Attachable": false,
24.    "Ingress": false,
25.    "ConfigFrom": {
26.      "Network": ""
27.    },
28.    "ConfigOnly": false,
29.    "Containers": {},
30.    "Options": {},
31.    "Labels": {}
32.  }
```

```
33. ]
34. [holuser@docker 12212-oradb-wlsstore]$
```

First we will create the oracle Database Docker image with a prebuild data base instance containing prebuild schemas etc. For the prebuild database instance we want it to be stored as a part of the image itself without external persistence. However both the Prebuild Oracle Database Docker Store image and our custom Oracle Database Docker image uses volumes (default anonymous volumes and host file volumes) to the configuration and datafiles. So we need to build a new image that don't externalize persistence and here we have to use the custom way of doing this.

Open a terminal and go to `/u01/OracleDatabase/dockerfiles/12.2.0.1` directory. Edit the `Dockerfile.ee` file in your favorite editor (`gedit`, `vi` etc.).

```
1. [holuser@docker ~]$ cd /u01/OracleDatabase/dockerfiles/12.2.0.1
2. [holuser@docker 12.2.0.1]$ ls -l
3. total 3372828
4. -rwxr-xr-x 1 holuser holuser    1071 Jan  4 13:03 checkDBStatus.sh
5. -rwxr-xr-x 1 holuser holuser     904 Jan  4 13:03 checkSpace.sh
6. -rw-r--r-- 1 holuser holuser     62 Jan  4 13:03 Checksum.ee
7. -rw-r--r-- 1 holuser holuser     62 Jan  4 13:03 Checksum.se2
8. -rwxr-xr-x 1 holuser holuser   2953 Jan  4 13:03 createDB.sh
9. -rw-r--r-- 1 holuser holuser   9203 Jan  4 13:03 dbca.rsp.tmpl
10. -rw-r--r-- 1 holuser holuser   6878 Jan  4 13:03 db_inst.rsp
11. -rw-r--r-- 1 holuser holuser   2639 Jan  4 13:03 Dockerfile.ee
12. -rw-r--r-- 1 holuser holuser   2645 Jan  4 13:03 Dockerfile.se2
13. -rwxr-xr-x 1 holuser holuser    2250 Jan  4 13:03 installDBBinaries.sh
14. -rw-rw-r-- 1 holuser holuser 3453696911 Jan 18 11:53 linuxx64_12201_database.zip
15. -rwxr-xr-x 1 holuser holuser    6140 Jan  4 13:03 runOracle.sh
16. -rwxr-xr-x 1 holuser holuser    1015 Jan  4 13:03 runUserScripts.sh
17. -rwxr-xr-x 1 holuser holuser     758 Jan  4 13:03 setPassword.sh
18. -rwxr-xr-x 1 holuser holuser     876 Jan  4 13:03 setupLinuxEnv.sh
19. -rwxr-xr-x 1 holuser holuser     678 Jan  4 13:03 startDB.sh
20. [holuser@docker 12.2.0.1]$ gedit Dockerfile.ee
```

To create an image with external persistence you'll need to comment out the `VOLUME` command in the `Dockerfile.ee`. Comment out the following line:

```
VOLUME ["$ORACLE_BASE/oradata"]
```

Look below for the result:

```
1. # LICENSE UPL 1.0
2. #
3. # Copyright (c) 1982-2017 Oracle and/or its affiliates. All rights reserved.
4. #
5. # ORACLE DOCKERFILES PROJECT
6. # -----
7. # This is the Dockerfile for Oracle Database 12c Release 2 Enterprise Edition
8. #
```

```

9.  # REQUIRED FILES TO BUILD THIS IMAGE
10. # -----
11. # (1) linuxx64_12201_database.zip
12. #   Download Oracle Database 12c Release 12 Enterprise Edition for Linux x64
13. #   from http://www.oracle.com/technetwork/database/enterprise-edition/downloads/index.html
14. #
15. # HOW TO BUILD THIS IMAGE
16. # -----
17. # Put all downloaded files in the same directory as this Dockerfile
18. # Run:
19. #   $ docker build -t oracle/database:12.2.0.1-ee .
20. #
21. # Pull base image
22. # -----
23. FROM oraclelinux:7-slim
24.
25. # Maintainer
26. # -----
27. MAINTAINER Gerald Venzl <gerald.venzl@oracle.com>
28.
29. # Environment variables required for this build (do NOT change)
30. # -----
31. ENV ORACLE_BASE=/opt/oracle \
32.     ORACLE_HOME=/opt/oracle/product/12.2.0.1/dbhome_1 \
33.     INSTALL_FILE_1="linuxx64_12201_database.zip" \
34.     INSTALL_RSP="db_inst.rsp" \
35.     CONFIG_RSP="dbca.rsp.tmpl" \
36.     PWD_FILE="setPassword.sh" \
37.     RUN_FILE="runOracle.sh" \
38.     START_FILE="startDB.sh" \
39.     CREATE_DB_FILE="createDB.sh" \
40.     SETUP_LINUX_FILE="setupLinuxEnv.sh" \
41.     CHECK_SPACE_FILE="checkSpace.sh" \
42.     CHECK_DB_FILE="checkDBStatus.sh" \
43.     USER_SCRIPTS_FILE="runUserScripts.sh" \
44.     INSTALL_DB_BINARIES_FILE="installDBBinaries.sh"
45.
46. # Use second ENV so that variable get substituted
47. ENV INSTALL_DIR=$ORACLE_BASE/install \
48.     PATH=$ORACLE_HOME/bin:$ORACLE_HOME/OPatch/:/usr/sbin:$PATH \
49.     LD_LIBRARY_PATH=$ORACLE_HOME/lib:/usr/lib \
50.     CLASSPATH=$ORACLE_HOME/jlib:$ORACLE_HOME/rdbms/jlib
51.
52. # Copy binaries
53. # -----
54. COPY $INSTALL_FILE_1 $INSTALL_RSP $SETUP_LINUX_FILE $CHECK_SPACE_FILE
55.     $INSTALL_DB_BINARIES_FILE $INSTALL_DIR/
56. COPY $RUN_FILE $START_FILE $CREATE_DB_FILE $CONFIG_RSP $PWD_FILE $CHECK_DB_FILE
57.     $USER_SCRIPTS_FILE $ORACLE_BASE/
58.
59. RUN chmod ug+x $INSTALL_DIR/*.sh && \
60.     sync && \
61.     $INSTALL_DIR/$CHECK_SPACE_FILE && \
62.     $INSTALL_DIR/$SETUP_LINUX_FILE

```



```

61.
62. # Install DB software binaries
63. USER oracle
64. RUN $INSTALL_DIR/$INSTALL_DB_BINARIES_FILE EE
65.
66. USER root
67. RUN $ORACLE_BASE/oraInventory/orainstRoot.sh && \
68.   $ORACLE_HOME/root.sh && \
69.   rm -rf $INSTALL_DIR
70.
71. USER oracle
72. WORKDIR /home/oracle
73.
74. #VOLUME ["$ORACLE_BASE/oradata"]
75. EXPOSE 1521 5500
76. HEALTHCHECK --interval=1m --start-period=5m \
77.   CMD "$ORACLE_BASE/$CHECK_DB_FILE" >/dev/null || exit 1
78.
79. # Define default command to start Oracle Database.
80. CMD exec $ORACLE_BASE/$RUN_FILE

```

Rename your existing image in order to keep it:

```

81. [holuser@docker 12.2.0.1]$ docker tag oracle/database:12.2.0.1-ee oracle/database-org:12.2.0.1-ee
82. [holuser@docker 12.2.0.1]$

```

Build the new Oracle Database Docker image with ephemeral database persistence (before building make sure that the **linuxx64_12201_database.zip** exists in the directory otherwise copy it from **/u01/software/linuxx64_12201_database.zip**):

```

1. [holuser@docker 12.2.0.1]$ docker build --force-rm=true --no-cache=true -t oracle/database:12.2.0.1-ee -f
   Dockerfile.ee .
2. Sending build context to Docker daemon 3.454GB
3. Step 1/16 : FROM oraclelinux:7-slim
4. ---> 9870bebf1d5
5. Step 2/16 : MAINTAINER Gerald Venzl <gerald.venzl@oracle.com>
6. ---> Running in a58bb72a5c16
7. ---> 3dde1e2f9207
8. Removing intermediate container a58bb72a5c16
9. Step 3/16 : ENV ORACLE_BASE /opt/oracle ORACLE_HOME /opt/oracle/product/12.2.0.1/dbhome_1
   INSTALL_FILE_1 "linuxx64_12201_database.zip" INSTALL_RSP "db_inst.rsp" CONFIG_RSP "dbca.rsp.tmpl"
   PWD_FILE "setPassword.sh" RUN_FILE "runOracle.sh" START_FILE "startDB.sh" CREATE_DB_FILE
   "createDB.sh" SETUP_LINUX_FILE "setupLinuxEnv.sh" CHECK_SPACE_FILE "checkSpace.sh"
   CHECK_DB_FILE "checkDBStatus.sh" USER_SCRIPTS_FILE "runUserScripts.sh"
   INSTALL_DB_BINARIES_FILE "installDBBinaries.sh"
10. ---> Running in 3b483f54c8ae
11. ---> 8f1a1514e376
12. Removing.....
13. ....

```

```

14. ---> Running in 99e1e5e20de3
15. ---> fb6324da3c8e
16. Removing intermediate container 99e1e5e20de3
17. Step 16/16 : CMD exec $ORACLE_BASE/$RUN_FILE
18. ---> Running in 1effd6210160
19. ---> 467178968536
20. Removing intermediate container 1effd6210160
21. Successfully built 467178968536
22. Successfully tagged oracle/database:12.2.0.1-ee
23. [holuser@docker 12.2.0.1]$

```

Now we want to create the database, prepopulate it with schema, tables and data. But first start the container:

```

1. [holuser@docker ~]$ docker run -d --name oracle-build -p 1521:1521 -p 5500:5500
   oracle/database:12.2.0.1-ee
2. 52edf70e6f2ecd1f399b992b662f9814bf217a3bd93d06081aeabf47db8c6004
3. [holuser@docker ~]$

```

When it's ready – check the log – connect to the container and execute the setPassword.sh script (set the password to 'welcome1'):

```

1. [holuser@docker 12.2.0.1]$ docker logs oracle-build
2.
3. ORACLE PASSWORD FOR SYS, SYSTEM AND PDBADMIN: GEx/D0XpC8o=1
4.
5. LSNRCTL for Linux: Version 12.2.0.1.0 - Production on 22-JAN-2018 16:38:35
6.
7. Copyright (c) 1991, 2016, Oracle. All rights reserved.
8.
9. Starting /opt/oracle/product/12.2.0.1/dbhome_1/bin/tnslsnr: please wait...
10.
11. TNSLSNR for Linux: Version 12.2.0.1.0 - Production
12. System parameter file is /opt/oracle/product/12.2.0.1/dbhome_1/network/admin/listener.ora
13. Log messages written to /opt/oracle/diag/tnslsnr/52edf70e6f2e/listener/alert/log.xml
14. Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1)))
15. Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=0.0.0.0)(PORT=1521)))
16.
17. Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=EXTPROC1)))
18. STATUS of the LISTENER
19. -----
20. Alias                LISTENER
21. Version              TNSLSNR for Linux: Version 12.2.0.1.0 - Production
22. Start Date           22-JAN-2018 16:38:36
23. Uptime               0 days 0 hr. 0 min. 0 sec
24. Trace Level          off
25. Security             ON: Local OS Authentication
26. SNMP                OFF
27. Listener Parameter File /opt/oracle/product/12.2.0.1/dbhome_1/network/admin/listener.ora
28. Listener Log File    /opt/oracle/diag/tnslsnr/52edf70e6f2e/listener/alert/log.xml
29. Listening Endpoints Summary...

```

```

30. (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1)))
31. (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=0.0.0.0)(PORT=1521)))
32. The listener supports no services
33. The command completed successfully
34. [WARNING] [DBT-10102] The listener configuration is not selected for the database. EM DB
    Express URL will not be accessible.
35. CAUSE: The database should be registered with a listener in order to access the EM DB Express
    URL.
36. ACTION: Select a listener to be registered or created with the database.
37. Copying database files
38. 1% complete
39. 13% complete
40. 25% complete
41. Creating and starting Oracle instance
42. 26% complete
43. 30% complete
44. 31% complete
45. 35% complete
46. 38% complete
47. 39% complete
48. 41% complete
49. Completing Database Creation
50. 42% complete
51. 43% complete
52. 44% complete
53. 46% complete
54. [holuser@docker 12.2.0.1]$ docker logs oracle-build
55. ORACLE PASSWORD FOR SYS, SYSTEM AND PDBADMIN: GEx/D0XpC8o=1
56.
57. LSNRCTL for Linux: Version 12.2.0.1.0 - Production on 22-JAN-2018 16:38:35
58.
59. Copyright (c) 1991, 2016, Oracle. All rights reserved.
60.
61. Starting /opt/oracle/product/12.2.0.1/dbhome_1/bin/tnslsnr: please wait...
62.
63. TNSLSNR for Linux: Version 12.2.0.1.0 - Production
64. System parameter file is /opt/oracle/product/12.2.0.1/dbhome_1/network/admin/listener.ora
65. Log messages written to /opt/oracle/diag/tnslsnr/52edf70e6f2e/listener/alert/log.xml
66. Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1)))
67. Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=0.0.0.0)(PORT=1521)))
68.
69. Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=EXTPROC1)))
70. STATUS of the LISTENER
71. -----
72. Alias                LISTENER
73. Version              TNSLSNR for Linux: Version 12.2.0.1.0 - Production
74. Start Date           22-JAN-2018 16:38:36
75. Uptime                0 days 0 hr. 0 min. 0 sec
76. Trace Level          off
77. Security              ON: Local OS Authentication
78. SNMP                 OFF
79. Listener Parameter File /opt/oracle/product/12.2.0.1/dbhome_1/network/admin/listener.ora
80. Listener Log File     /opt/oracle/diag/tnslsnr/52edf70e6f2e/listener/alert/log.xml

```

81. Listening Endpoints Summary...

82. (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1)))

83. (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=0.0.0.0)(PORT=1521)))

84. The listener supports no services

85. The command completed successfully

86. [WARNING] [DBT-10102] The listener configuration is not selected for the database. EM DB Express URL will not be accessible.

87. CAUSE: The database should be registered with a listener in order to access the EM DB Express URL.

88. ACTION: Select a listener to be registered or created with the database.

89. Copying database files

90. 1% complete

91. 13% complete

92. 25% complete

93. Creating and starting Oracle instance

94. 26% complete

95. 30% complete

96. 31% complete

97. 35% complete

98. 38% complete

99. 39% complete

100. 41% complete

101. Completing Database Creation

102. 42% complete

103. 43% complete

104. 44% complete

105. 46% complete

106. 47% complete

107. 50% complete

108. Creating Pluggable Databases

109. 55% complete

110. 75% complete

111. Executing Post Configuration Actions

112. 100% complete

113. Look at the log file "/opt/oracle/cfgtoollogs/dbca/ORCLCDB/ORCLCDB.log" for further details.

114.

115. SQL*Plus: Release 12.2.0.1.0 Production on Mon Jan 22 16:46:08 2018

116.

117. Copyright (c) 1982, 2016, Oracle. All rights reserved.

118.

119.

120. Connected to:

121. Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production

122.

123. SQL>

124. System altered.

125.

126. SQL>

127. Pluggable database altered.

128.

129. SQL> Disconnected from Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production

```

130. The Oracle base remains unchanged with value /opt/oracle
131. #####
132. DATABASE IS READY TO USE!
133. #####
134. The following output is now a tail of the alert.log:
135. Completed: alter pluggable database ORCLPDB1 open
136. 2018-01-22T16:46:07.307410+00:00
137. ORCLPDB1(3):CREATE SMALLFILE TABLESPACE "USERS" LOGGING DATAFILE
    '/opt/oracle/oradata/ORCLCDB/ORCLPDB1/users01.dbf' SIZE 5M REUSE AUTOEXTEND ON
    NEXT 1280K MAXSIZE UNLIMITED EXTENT MANAGEMENT LOCAL SEGMENT SPACE
    MANAGEMENT AUTO
138. ORCLPDB1(3):Completed: CREATE SMALLFILE TABLESPACE "USERS" LOGGING
    DATAFILE '/opt/oracle/oradata/ORCLCDB/ORCLPDB1/users01.dbf' SIZE 5M REUSE
    AUTOEXTEND ON NEXT 1280K MAXSIZE UNLIMITED EXTENT MANAGEMENT LOCAL
    SEGMENT SPACE MANAGEMENT AUTO
139. ORCLPDB1(3):ALTER DATABASE DEFAULT TABLESPACE "USERS"
140. ORCLPDB1(3):Completed: ALTER DATABASE DEFAULT TABLESPACE "USERS"
141. 2018-01-22T16:46:08.184957+00:00
142. ALTER SYSTEM SET control_files='/opt/oracle/oradata/ORCLCDB/control01.ctl'
    SCOPE=SPFILE;
143. ALTER PLUGGABLE DATABASE ORCLPDB1 SAVE STATE
144. Completed: ALTER PLUGGABLE DATABASE ORCLPDB1 SAVE STATE
145.
146.
147. [holuser@docker 12.2.0.1]$ docker exec oracle-build ./setPassword.sh welcome1
148.
149. The Oracle base remains unchanged with value /opt/oracle
150.
151. SQL*Plus: Release 12.2.0.1.0 Production on Mon Jan 22 16:46:24 2018
152.
153. Copyright (c) 1982, 2016, Oracle. All rights reserved.
154.
155.
156. Connected to:
157. Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production
158.
159. SQL>
160. User altered.
161.
162. SQL>
163. User altered.
164.
165. SQL>
166. Session altered.
167.
168. SQL>
169. User altered.
170.
171. SQL> Disconnected from Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit
    Production
172. [holuser@docker 12.2.0.1]$

```

In the terminal go to the /u01/OracleWebLogic/samples/12212-oradb-wlsstore directory:

1. [holuser@docker 12.2.0.1]\$ cd /u01/OracleWebLogic/samples/12212-oradb-wlsstore/
2. [holuser@docker 12212-oradb-wlsstore]\$

Create the schemas. To run the DDL that creates the tables needed by the application, copy createSchema.sql into the Database container:

1. [holuser@docker 12212-oradb-wlsstore]\$ docker cp createSchema.sql oracle-build:/opt/oracle
2. [holuser@docker 12212-oradb-wlsstore]\$

Run sqlplus to run the DDL

1. [holuser@docker 12212-oradb-wlsstore]\$ docker exec -ti oracle-build
/opt/oracle/product/12.2.0.1/dbhome_1/bin/sqlplus system/welcome1@orclpdb1
@/opt/oracle/createSchema.sql
- 2.
3. SQL*Plus: Release 12.2.0.1.0 Production on Mon Jan 22 17:11:50 2018
- 4.
5. Copyright (c) 1982, 2016, Oracle. All rights reserved.
- 6.
- 7.
8. Connected to:
9. Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production
- 10.
11. DROP TABLE Bid
12. *
13. ERROR at line 1:
14. ORA-00942: table or view does not exist
- 15.
- 16.
17. DROP TABLE Auction
18. *
19. ERROR at line 1:
20. ORA-00942: table or view does not exist
- 21.
- 22.
23. DROP TABLE Image
24. *
25. ERROR at line 1:
26. ORA-00942: table or view does not exist
- 27.
- 28.
- 29.
30. Table created.
- 31.
- 32.
33. Table created.
- 34.

```

35.
36. Table created.
37.
38.
39. Sequence created.
40.
41.
42. Sequence created.
43.
44.
45. Sequence created.
46.
47.
48. Sequence created.
49.
50. Disconnected from Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production
51. [holuser@docker 12212-oradb-wlsstore]$

```

Stop the container (and therefore also the database) before generating your new prebuilt image:

```

1. [holuser@docker 12212-oradb-wlsstore]$ docker container stop oracle-build
2. oracle-build
3. [holuser@docker 12212-oradb-wlsstore]$

```

Create the image with the prebuilt Oracle Database:

```

1. [holuser@docker 12212-oradb-wlsstore]$ docker commit -m "Image with prebuilt database" oracle-
   build oracle/db-prebuilt:12.2.0.1-ee
2. sha256:9e1668212cf1618614d2fe2b3fbcd9bbdfef87f84a00e8369750212bb231f4e5
3. [holuser@docker 12212-oradb-wlsstore]$ docker images
4. REPOSITORY          TAG          IMAGE ID          CREATED          SIZE
5. oracle/db-prebuilt   12.2.0.1-ee  9e1668212cf1     8 seconds ago   17GB
6. ....
7. ....

```

Clean up. First remove the temporary container:

```

1. [holuser@docker 12212-oradb-wlsstore]$ docker rm oracle-build

```

Remove the image without VOLUME and rename the old image (if exists):

```

1. [holuser@docker 12212-oradb-wlsstore]$ docker rmi oracle/database:12.2.0.1-ee
2. Untagged: oracle/database:12.2.0.1-ee
3. [holuser@docker 12212-oradb-wlsstore]$ docker tag oracle/database-org:12.2.0.1-ee
   oracle/database:12.2.0.1-ee
4. [holuser@docker 12212-oradb-wlsstore]$ docker images

```

5. REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
6. oracle/db-prebuilt	12.2.0.1-ee	9e1668212cf1	25 minutes ago	17GB
7. 12213-domain	latest	7cf548c2b3a0	3 days ago	1.43GB
8. oracle/weblogic	12.2.1.3-developer	79311affda49	3 days ago	1.43GB
9. oracle/database-org	12.2.0.1-ee	1748a49d6137	3 days ago	13.3GB
10. oracle/database	12.2.0.1-ee	1748a49d6137	3 da	
11. ...				
12. ...				

Remove the '-org' image:

1. [holuser@docker 12212-oradb-wlsstore]\$ docker rmi oracle/database-org:12.2.0.1-ee
2. Untagged: oracle/database-org:12.2.0.1-ee
3. [holuser@docker 12212-oradb-wlsstore]\$

Uncomment the volume instruction in your Dockerfile:

VOLUME ["\$ORACLE_BASE/oradata"]

1. [holuser@docker 12212-oradb-wlsstore]\$ gedit
/u01/OracleDatabase/dockerfiles/12.2.0.1/Dockerfile.ee
2. [holuser@docker 12212-oradb-wlsstore]\$

Now we have made the prebuild Oracle Database Docker Image. Let's build the Weblogic Docker Image from the supplied Dockerfile:

1. #
2. # Copyright (c) 2014-2017 Oracle and/or its affiliates. All rights reserved.
3. #
4. #
5. #Licensed under the Universal Permissive License v 1.0 as shown at
http://oss.oracle.com/licenses/upl.
6. #
7. # ORACLE DOCKERFILES PROJECT
8. # -----
9. # This Dockerfile extends the Oracle WebLogic image in DockerStore
10. # configures a Data Source using WLST online to connect to an Oracle DB
11. # container from DockerStore
12. #
13. # REQUIRED FILES TO BUILD THIS IMAGE
14. # -----
15. #
16. # HOW TO BUILD THIS IMAGE
17. # -----
18. # Put all downloaded files in the same directory as this Dockerfile
19. # Run:
20. # \$ sudo docker build -t 12213-oradb-wlsstore .


```

21. #
22.
23. # Pull base image
24. # -----
25. FROM store/oracle/weblogic:12.2.1.3
26.
27. # Maintainer
28. # -----
29. MAINTAINER Monica Riccelli <monica.riccelli@oracle.com>
30.
31. # Environment variables required for this build (do NOT change)
32. # -----
33. ENV MW_HOME="$ORACLE_HOME" \
34.     DOMAIN_HOME=/u01/oracle/user_projects/domains/base_domain \
35.     PATH=$PATH:/u01/oracle:$DOMAIN_HOME
36.
37. # Environment variables required for application deployment
38. # -----
39. ENV APP_NAME="auction" \
40.     APP_PKG_FILE="auction.war" \
41.     APP_PKG_LOCATION="/u01/oracle"
42.
43.
44. # Copy supplemental package and scripts
45. # -----
46. USER root
47. COPY container-scripts/* /u01/oracle/
48. RUN chmod +xr /u01/oracle/*.sh
49.
50. # Installation of Supplemental Quick Installer
51. # -----
52. USER oracle
53. RUN cd /u01/oracle && \
54.     ./ds-wlst-online-config.sh && \
55.     wlst /u01/oracle/app-deploy.py
56.
57.
58. EXPOSE $ADMIN_PORT
59. WORKDIR $DOMAIN_HOME
60.
61. CMD ["startWebLogic.sh"]

```

Take your time to understand the Dockerfile. If any questions ask the instructor.

Build the Oracle Weblogic Docker Image:

```

1. [holuser@docker 12212-oradb-wlsstore]$ docker build -t 12213-oradb-wlsstore .
2. Sending build context to Docker daemon 948.7kB
3. Step 1/12 : FROM store/oracle/weblogic:12.2.1.3
4. ---> 1fcee95cc21b
5. Step 2/12 : MAINTAINER Monica Riccelli <monica.riccelli@oracle.com>
6. ---> Running in 010171112f2b

```

```

7. ---> 1a639ff061b5
8. Removing intermediate container 010171112f2b
9. Step 3/12 : ENV MW_HOME "$ORACLE_HOME" DOMAIN_HOME
/u01/oracle/user_projects/domains/base_domain PATH $PATH:/u01/oracle:$DOMAIN_HOME
10. ---> Running in 812e34c22ab2
11. ---> a814258ec4eb
12. ....
13. ....
14. ....
15. [holuser@docker 12212-oradb-wlsstore]$ docker images
16. REPOSITORY          TAG          IMAGE ID      CREATED      SIZE
17. 12213-oradb-wlsstore latest       8e75c5cc6b3d About a minute ago 1.17GB
18. oracle/db-prebuilt  12.2.0.1-ee 9e1668212cf1 About an hour ago 17GB
19. 12213-domain
20. ....

```

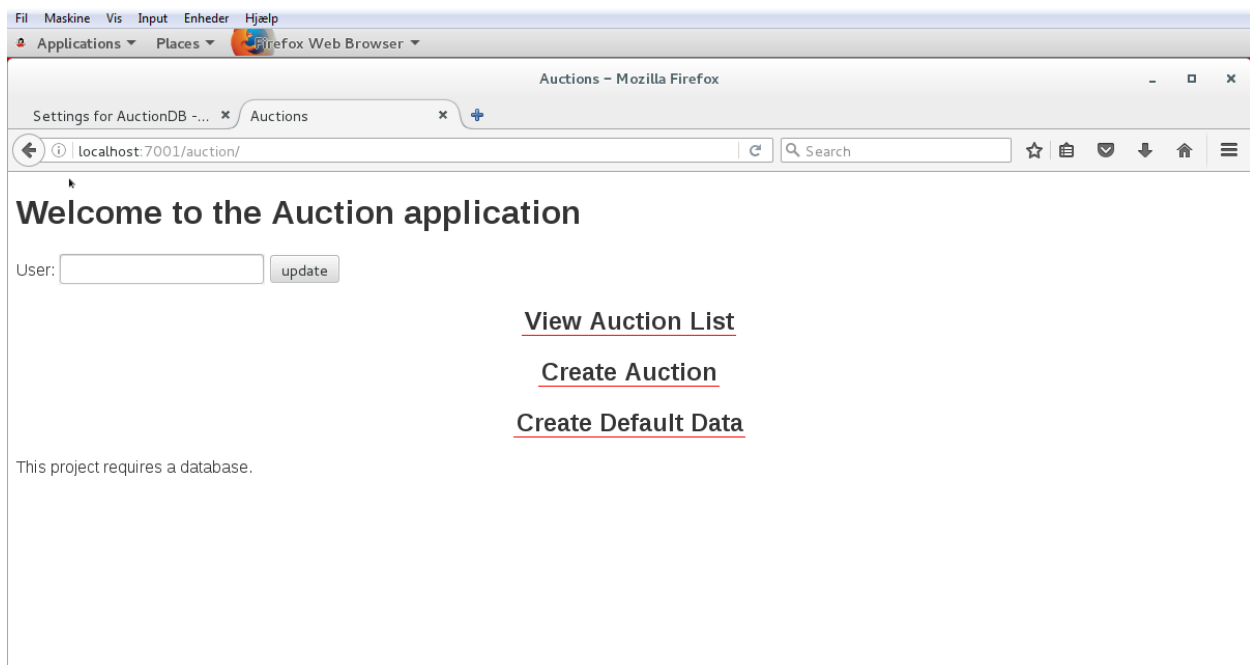
Start both the prebuilt Oracle Database Docker Image and the Oracle Weblogic Docker Image that we just build:

```

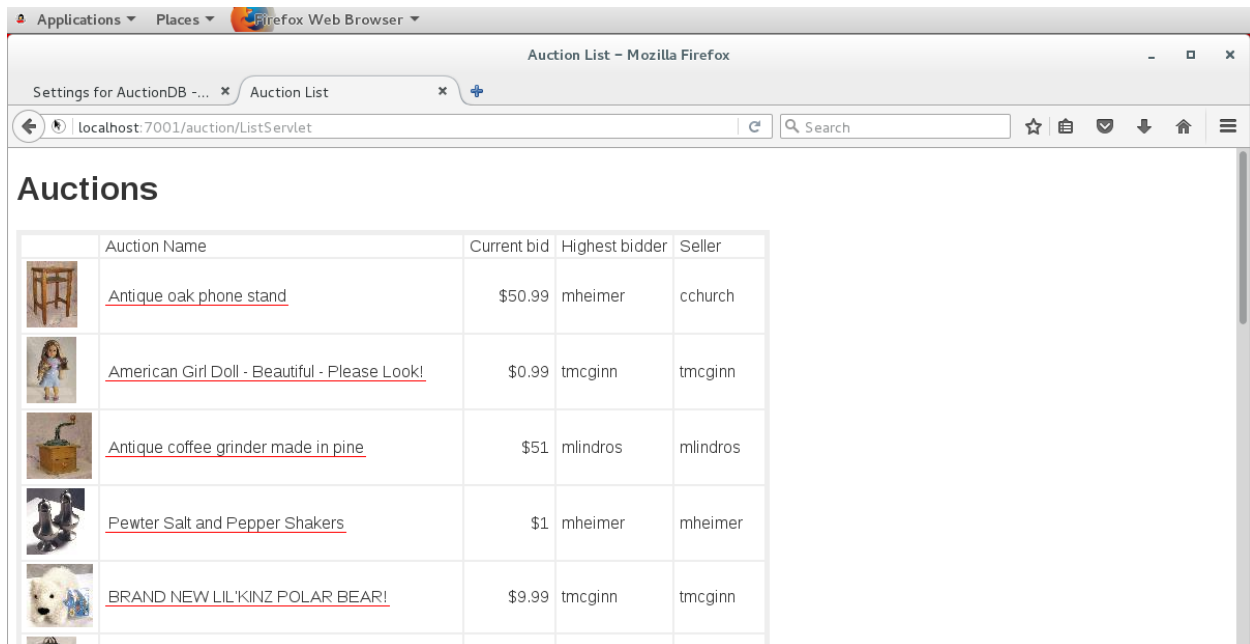
1. [holuser@docker 12212-oradb-wlsstore]$ docker run -d --name oracle-build --network=SampleNET -p 1521:1521 -p 5500:5500 -it --shm-size="8g" oracle/db-prebuilt:12.2.0.1-ee
2. 64eebf6f589196477e97b04e8b4eb50ddf8faa066864545d1b43808dc9d632b
3. [holuser@docker 12212-oradb-wlsstore]$ docker run -d -i -t -p 7001:7001 --network=SampleNET --name WLSStoreContainer 12213-oradb-wlsstore:latest
4. 595d8335b7baa029d65222dbcc335d5c979b696e2610d441570b514ac96ecd01
5. [holuser@docker 12212-oradb-wlsstore]$






```

Access the application by invoking from the browser in the VBox: ***http://localhost:7001/auction***



Create default data by clicking on the link “Create Default Data” and then hit “Yes, Create Default Data”. This will create sample auctions if everything is working as expected. The result should be as below when you “View Auction List”:



	Auction Name	Current bid	Highest bidder	Seller
	Antique oak phone stand	\$50.99	mheimer	cchurch
	American Girl Doll - Beautiful - Please Look!	\$0.99	tmcginn	tmcginn
	Antique coffee grinder made in pine	\$51	mlindros	mlindros
	Pewter Salt and Pepper Shakers	\$1	mheimer	mheimer
	BRAND NEW LIL'KINZ POLAR BEAR!	\$9.99	tmcginn	tmcginn

Hence you are now able to create prebuilt Oracle Database Docker Images and connect Oracle Weblogic containers to Oracle Database Docker containers.

When you run the Auction Application you want to treat both Docker images as one meaning that you won't probably not run the one without the other one. To simplify this we can take advantage of Docker Compose.

Utilize Docker Compose

Docker Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application's services. Then, with a single command, you create and start all the services from your configuration. To learn more about all the features of Compose, see the list of features.

First we need to install Docker Compose (you don't need to be in a specific directory):

```
1. [holuser@docker container-scripts]$ sudo curl -L https://github.com/docker/compose/releases/download/1.18.0/docker-
compose -o /usr/local/bin/docker-compose
2. [sudo] password for holuser:
3. % Total % Received % Xferd Average Speed Time Time Time Current
4. Dload Upload Total Spent Left Speed
5. 100 617 0 617 0 0 856 0 --:--:-- --:--:-- --:--:-- 855
6. 100 8280k 100 8280k 0 0 1413k 0 0:00:05 0:00:05 --:--:-- 1874k
7. [holuser@docker container-scripts]$ sudo chmod +x /usr/local/bin/docker-compose
8. [holuser@docker container-scripts]$ docker-compose --version
9. docker-compose version 1.18.0, build 8dd22a9
10. [holuser@docker container-scripts]$
```

Open a terminal and go to the /u01/OracleWebLogic/samples/12212-oradb-wlsstore directory. Create the following dokcer-compose.yml file:

```
11. version: "3"
12. services:
13.   auction:
14.     build: .
15.     ports:
16.       - "7001:7001"
17.     depends_on:
18.       - "oracle-build"
19.     networks:
20.       - SampleNET
21.   oracle-build:
22.     image: oracle/db-prebuilt:12.2.0.1-ee
23.     ports:
24.       - "1521:1521"
25.       - "5500:5500"
26.     networks:
27.       - SampleNET
28. networks:
29.   SampleNET:
30.     driver: bridge
```

You can control the order of service startup with the `depends_on` option. Compose always starts containers in dependency order, where dependencies are determined by `depends_on`, `links`, `volumes_from`, and `network_mode: "service:..."`.

However, Compose will not wait until a container is “ready” (whatever that means for your particular application) - only until it’s running. There’s a good reason for this.

The problem of waiting for a database (for example) to be ready is really just a subset of a much larger problem of distributed systems. In production, your database could become unavailable or move hosts at any time. Your application needs to be resilient to these types of failures.

To handle this, your application should attempt to re-establish a connection to the database after a failure. If the application retries the connection, it should eventually be able to connect to the database.

The best solution is to perform this check in your application code, both at startup and whenever a connection is lost for any reason. However, if you don’t need this level of resilience, you can work around the problem with a wrapper script like a tool such as [wait-for-it](#), [dockerize](#), or [sh-compatible wait-for](#). These are small wrapper scripts which you can include in your application’s image and will poll a given host and port until it’s accepting TCP connections.

For example, to use `wait-for-it.sh` or `wait-for` to wrap your service’s command:

```

31. version: "3"
32. services:
33.   web:
34.     build: .
35.     ports:
36.       - "7001:7001"
37.     depends_on:
38.       - "oracle-build"
39.     command: ["/wait-for-it.sh", "oracle-build:5432"]
40.   networks:
41.     - SampleNET
42. oracle-build:
43.   image: oracle/db-prebuilt:12.2.0.1-ee
44.   ports:
45.     - "1521:1521"
46.     - "5500:5500"
47.   networks:
48.     - SampleNET
49. networks:
50.   SampleNET:
51.     driver: bridge

```

Note: There are limitations to this solution; e.g., it doesn't verify when a specific service is really ready. If you add more arguments to the command, you'll need to use the bash shift command with a loop, as shown in the next example.

Make sure you have stopped all containers before continuing:

```

1. [holuser@docker ~]$ docker ps --all
2. CONTAINER ID   IMAGE          COMMAND                  CREATED    STATUS    PORTS
   NAMES
3. [holuser@docker ~]$

```

If there are containers listed do the following:

```

1. [holuser@docker ~]$ docker container rm --force <container_id> <container_id> .....

```

Start up the application by running:

```

1. [holuser@docker 12212-oradb-wlsstore]$ docker-compose up -d
2. Creating network "12212oradbwlsstore_SampleNET" with driver "bridge"
3. Building auction
4. Step 1/12 : FROM store/oracle/weblogic:12.2.1.3
5. ---> 1fcee95cc21b
6. Step 2/12 : MAINTAINER Monica Riccelli <monica.riccelli@oracle.com>
7. ---> Using cache
8. ---> 2a8b6d8fa3da
9. Step 3/12 : ENV MW_HOME "$ORACLE_HOME" DOMAIN_HOME
   /u01/oracle/user_projects/domains/base_domain PATH $PATH:/u01/oracle:$DOMAIN_HOME
10. ---> Using cache
11. ---> 43351d76cbb8
12. Step 4/12 : ENV APP_NAME "auction" APP_PKG_FILE "auction.war" APP_PKG_LOCATION "/u01/oracle"

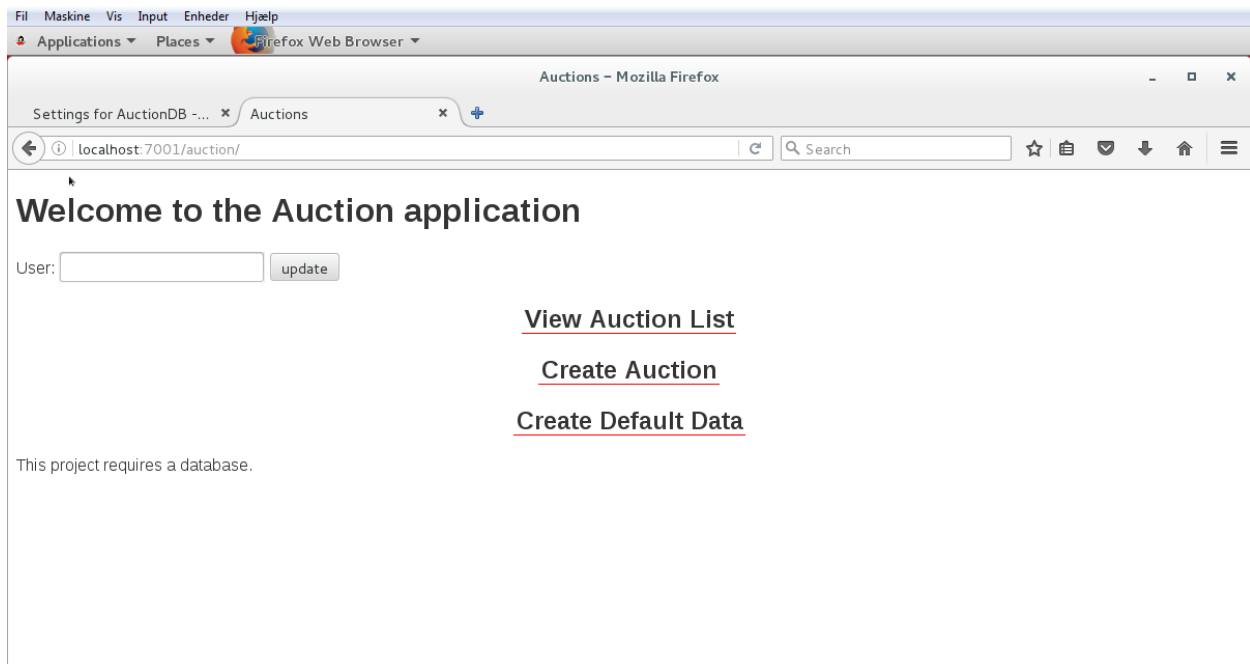
```

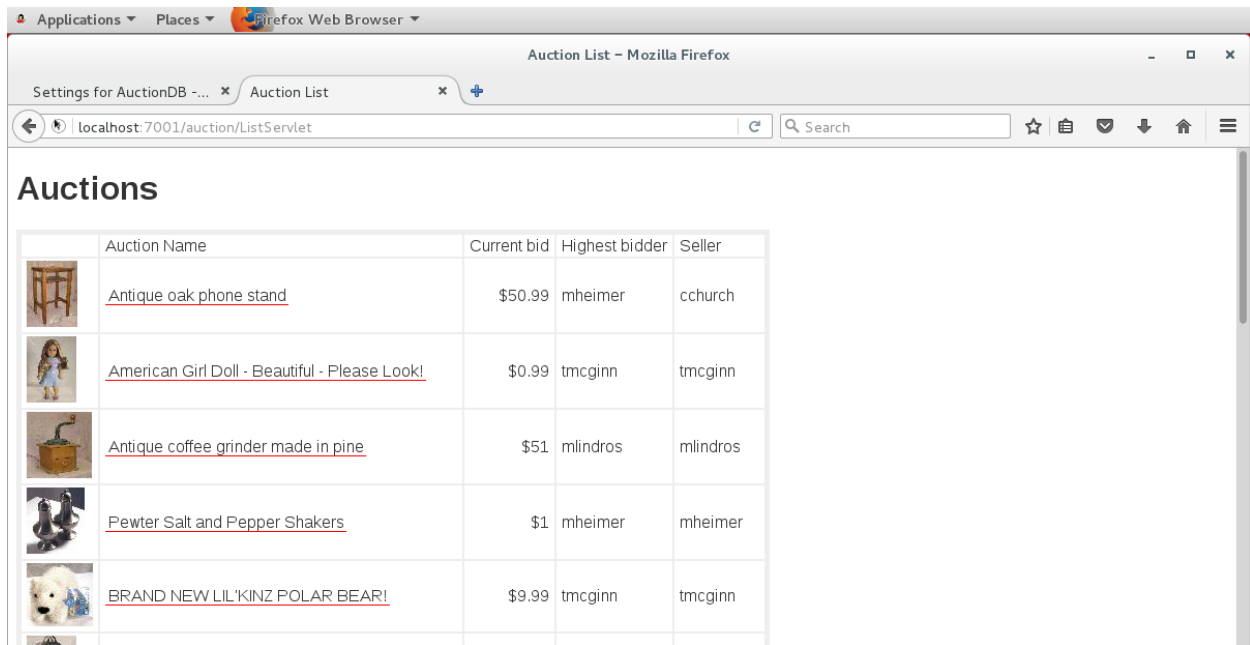
```

13. ---> Using cache
14. ---> d1b44fc48ed1
15. Step 5/12 : USER root
16. ---> Using cache
17. ---> 1dcd75e7ce60
18. Step 6/12 : COPY container-scripts/* /u01/oracle/
19. ---> ce8eb535a4d7
20. Removing intermediate container 75ed6983be8f
21. Step 7/12 : RUN chmod +xr /u01/oracle/*.sh
22. ....
23. ....
24. Creating 12212oradbwlsstore_orcl-node_1 ... done
25. Creating 12212oradbwlsstore_orcl-node_1 ...
26. Creating 12212oradbwlsstore_auction_1 ... done
27. [holuser@docker 12212-oradb-wlsstore]$ docker ps
28. CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
    PORTS              NAMES
29. 175097398cd4        12212oradbwlsstore_auction    "startWebLogic.sh"   14 seconds ago     Up 13 seconds
    0.0.0.0:7001->7001/tcp    12212oradbwlsstore_auction_1
30. bd7625cf1b9d        oracle/db-prebuilt:12.2.0.1-ee "/bin/sh -c 'exec ..." 15 seconds ago     Up 13 seconds (health:
    starting) 0.0.0.0:1521->1521/tcp, 0.0.0.0:5500->5500/tcp 12212oradbwlsstore_orcl-node_1
31. [holuser@docker 12212-oradb-wlsstore]$

```

Take your browser and check that the Auction application on <http://localhost:7001/auction> is working by creating default auction data and list the data afterwards...





You can now control the Auction Application by commands like:

- Docker-compose stop
- Docker-compose build
- Docker-compose rm
- Docker-compose up -d

If you execute:

1. [holuser@docker 12212-oradb-wlsstore]\$ docker-compose up -d
2. 12212oradbwlsstore_oracle-build_1 is up-to-date
3. 12212oradbwlsstore_auction_1 is up-to-date
4. [holuser@docker 12212-oradb-wlsstore]\$

“Docker-compose up -d” checks if the images are up to date. That means if you trigger a rebuild of the iamges and run it again you will see the following:

1. [holuser@docker 12212-oradb-wlsstore]\$ docker-compose build --no-cache
2. oracle-build uses an image, skipping
3. Building auction
4. Step 1/12 : FROM store/oracle/weblogic:12.2.1.3
5. ---> 1fcee95cc21b
6. Step 2/12 : MAINTAINER Monica Riccelli <monica.riccelli@oracle.com>
7. ---> Running in bd2aa18c36ff
8. ---> 10ab63a10a8c
9. Removing intermediate container bd2aa18c36ff
10. Step 3/12 : ENV MW_HOME "\$ORACLE_HOME" DOMAIN_HOME
/u01/oracle/user_projects/domains/base_domain PATH \$PATH:/u01/oracle:\$DOMAIN_HOME
11. ---> Running in bc2796eaaa72

```

12. ....
13. ....
14. ---> 43dd2cd183f7
15. Removing intermediate container 20d5e323d9f8
16. Step 10/12 : EXPOSE $ADMIN_PORT
17. ---> Running in 2560129497d8
18. ---> f78b8d633f44
19. Removing intermediate container 2560129497d8
20. Step 11/12 : WORKDIR $DOMAIN_HOME
21. ---> 1e71f94f53c6
22. Removing intermediate container db8233875e0f
23. Step 12/12 : CMD startWebLogic.sh
24. ---> Running in fe161c5e4064
25. ---> 12311aec1025
26. Removing intermediate container fe161c5e4064
27. Successfully built 12311aec1025
28. Successfully tagged 12212oradbwlsstore_auction:latest
29. [holuser@docker 12212-oradb-wlsstore]$ docker-compose up -d
30. 12212oradbwlsstore_oracle-build_1 is up-to-date
31. Recreating 12212oradbwlsstore_auction_1 ... done
32. [holuser@docker 12212-oradb-wlsstore]$ docker ps
33. CONTAINER ID        IMAGE                                     COMMAND                  CREATED              STATUS
    PORTS              NAMES
34. 76228bba9204        12212oradbwlsstore_auction             "startWebLogic.sh"      5 minutes ago       Up 5 minutes
    0.0.0.0:7001->7001/tcp    12212oradbwlsstore_auction_1
35. 136f2db55168        oracle/db-prebuilt:12.2.0.1-ee         "/bin/sh -c 'exec ..."  2 hours ago         Up 2 hours (healthy)
    0.0.0.0:1521->1521/tcp, 0.0.0.0:5500->5500/tcp  12212oradbwlsstore_oracle-build_1
36. [holuser@docker 12212-oradb-wlsstore]$

```

As you can see “docker-compose up -d” detects a new image and recreates the container with the new image and starts it.