Docker for developers

Razvan Anghel
virtual7 GmbH
Karlsruhe, Germany

Keywords:

Docker, Oracle Fusion Middleware, Development

Introduction

This presentation focuses on using Docker and the images and Dockerfiles released by Oracle to create small, efficient environments for developers to use on their workstations. It isn't intended to give you exact recipes for every possible project requirements but to show you the possibilities and allow you to form your own ideas and workflows.

Docker fundamentals for beginners

Docker leverages open source technologies like Control Groups, Container Formats, Union File Systems and Namespaces to create containers that wrap up a piece of software in a complete filesystem containing everything it needs to run. This guarantees that it will always run the same, regardless of the hardware it is running on.

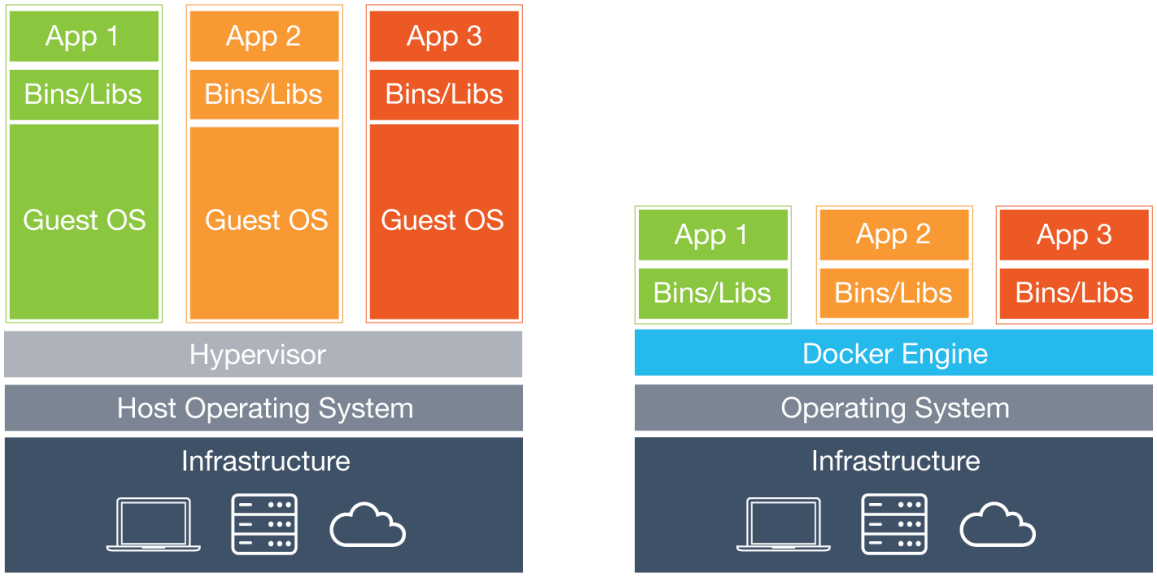

Illustration. 1: Virtual Machines and containers

The main advantages of using Docker instead of Virtual Machines are: lower system overhead, better application performance and higher deployment density.

Functional components:

- Docker Images
  - Read-only template for creating containers
  - Can be created by Docker files, updated, pulled and pushed to Docker registries
  - The *build* component of Docker
- Docker Containers
  - Created from Docker images
  - Holds everything needed for an application to run
  - Each container is an isolated and secure application platform
  - The *run* component of Docker
- Docker Registry
  - Private or public repository for Docker images
  - The official repository is called the Docker Hub
  - The *distribution* component of Docker

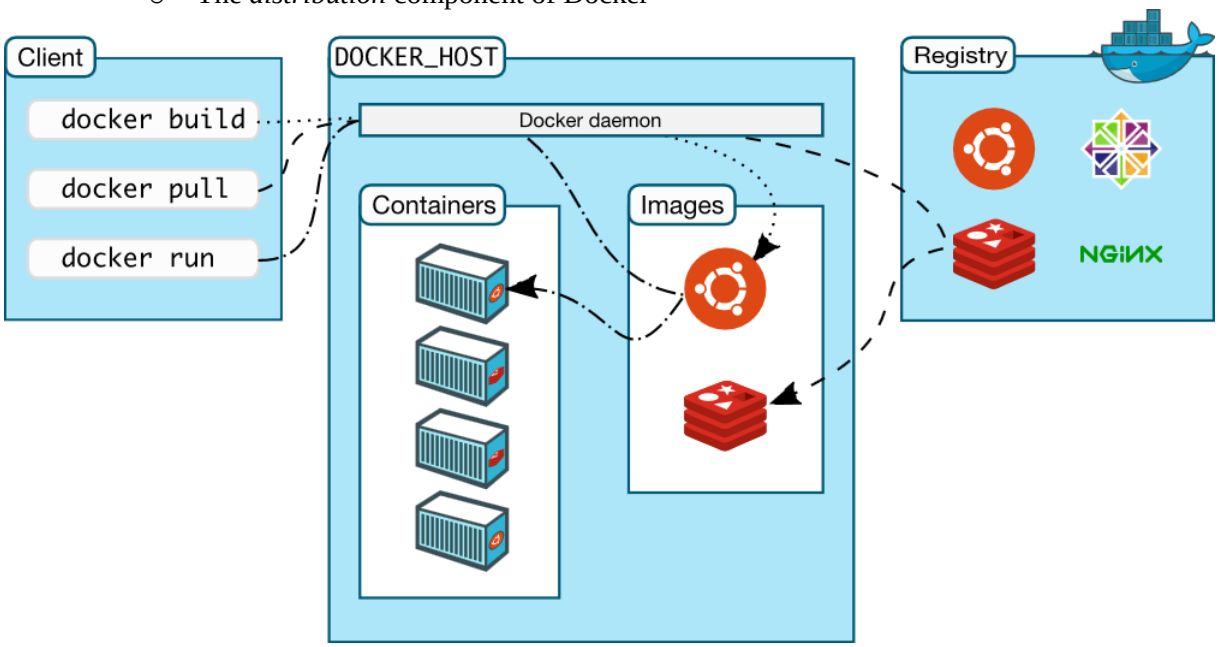

Illustration 2: Docker architecture

Using Docker to run a simple Elasticsearch container:

```
docker run -d --name es -p 9300:9300 -v
/data:/usr/share/elasticsearch/data elasticsearch
```

- run command
- Daemonize
- Name it “es”
- Map container port to host port
- Mount local dir to container dir
- Image from which to spawn container

Where to find resources

https://github.com/oracle/docker-images

- Provides Dockerfiles to build your own images for Oracle software (Database, Weblogic, SOA, etc.)
- You still need to download the installation kits from Oracle
- Contains detailed guides

https://container-registry.oracle.com/

- Provides ready built images for a few Oracle products
- You need to link your Oracle account
- This is a Docker registry so you can pull from it directly from the command line

http://hub.docker.com

- The official Docker hub, for OSS

https://docs.docker.com/

- The official Docker documentation
- Important for understanding how images and containers work.
- Use it to learn about Docker storage and networking.

Why use Docker for development and testing environments

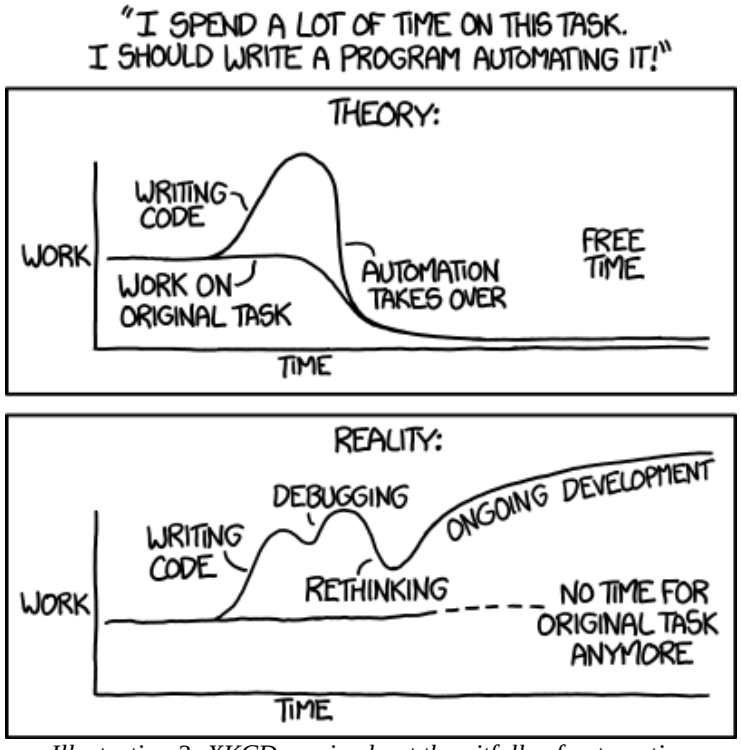

Illustration 3: XKCD comic about the pitfalls of automation

Advantages of using Docker:

- Keep your workstation clean and use resources efficiently
- Store Dockerfiles that are used to build your whole environment in Git
- No need to store and maintain tens of gigabytes of VMs
- Easily share environments and bring new developers into projects
- Guarantee identical environments between teammates and between developers and testers (can also be extended to production)

Disadvantages:

- Additional work to develop Dockerfiles
- If the customer doesn't use Docker in production, extra work for delivery

Using Docker for Oracle development

The purpose of this presentation is to show the advantages of using Docker for development and to get you started on using the available resources to build environments. Keeping this in mind let's talk about 3 development areas: ADF, Middleware and APEX.

For ADF development a common scenario is needing a database and a Weblogic server. A database container is relatively easy to build. If you need persistent data or you have specific networking requirements you can find information about this in the README files on GitHub and more generically in the Docker documentation. Almost everything is possible with Docker, you just need to make the decision whether or not it's worth automating all tasks or just some of them.

Simple scenario:

- Use a long-lived DB container, execute SQL scripts to create schemas and import data
- Use a Weblogic container that includes a domain with an AdminServer to deploy and test your applications

The database image can be populated with schemas and data at build time or you can run scripts after you start it. Make your choice based on how often you need to create new containers. Another choice would be if this is a common database used for multiple apps and projects or just specific to one project.
A Weblogic images if built from a base Linux image with java installed and configured, named *serverjre* in the following steps. On top of this you install Weblogic and create the domain. Other configurations can be done on top of this domain image.
Your application can be deployed at build time if you like. For this you would build another image on top of the domain image that uses WLST scripts to deploy. You can use prebuild binaries or you can trigger a build process with Maven.

Steps:

- Build your DB container
  ```
  ./buildDockerImage.sh -v 11.2.0.2 -x
  ```
- Run it
  ```
  docker run --name oracle-xe -d --shm-size=1g -p 1521:1521 -e
  ORACLE_PWD=welcome1 –v oracle/database:11.2.0.2-xe
  ```
- Import SQL script (for app data)
  ```
  docker exec -ti oracle-xe bash
  sqlplus sys@xe as sysdba @/mnt/script.sql
  ```
- Build Weblogic Image. First the clean Linux filesystem with Java, then the clean WLS install.
  ```
  docker build -t oracle/serverjre:8 .
  ```

```
./buildDockerImage.sh -v 12.2.1.2 -i
```

- Build Weblogic domain image (including JRF)
```
cd /root/OracleWebLogic/samples/12212-domain-adf
./build.sh welcome1
```
- Build App deploy image
```
docker build -t 12212-domain-adf-deploy .
```
- Run container
```
docker run -d --name wlsadmin --hostname wlsadmin 12212-domain-adf-deploy
```

Advanced scenario:

- Use a long-lived DB container, use SQL scripts that run automatically when building domain image
- Use containers for each managed server. Details on how to do this can be found on the oracle-images GitHub.

For Middleware development you use the exact same steps to create the database container and Weblogic container with a created domain. Since Middleware requires a separate managed server create containers for those and connect them to the domain. Installation of middleware will be handled using silent installs and kickstart files.
This is an advanced topic and requires you to understand how process virtualization handles multiple processes in the same container.
The extra steps would be: Running the RCU, Extending the domain, Installing the middleware.

APEX development is a great scenario for Docker as it gives you the ability to easily work with multiple versions of APEX at the same time. You would do this by creating a base database image and building separate images on top of that with different APEX installs.
Example Dockerfile for database with APEX:

```
FROM ubuntu
MAINTAINER Maksym Bilenko <sath891@gmail.com>
ENV DEBIAN_FRONTEND noninteractive
ENV LD_LIBRARY_PATH /instantclient_12_1
ENV USER sys
ENV PASS oracle
ENV HOST oracle-database
ENV PORT 1521
ENV SID XE
ENV HTTP_PORT 8080
ENV APEX_VERSION 5.1.2
RUN apt-get update && apt-get -y install libaio1 unzip && apt-get clean && rm -rf /tmp/* /var/lib/apt/lists/* /var/tmp/*
ADD instantclient-* /tmp/
ADD apex* /apex_5.1.2/
ADD entrypoint.sh /entrypoint.sh
ENTRYPOINT ["/entrypoint.sh"]
```

The *entrypoint.sh* script handles installation and is too large to include here. It can be found at: https://github.com/MaksymBilenko/docker-oracle-apex/blob/5.1.2/entrypoint.sh

Extra tools for automation

- Use Vagrant with Docker to also automate Host OS provisioning
- Use Docker Compose to manage multiple containers together. Example compose file:

```
version: '2'

networks:
  &network wlsnet:
    driver: bridge

services:

  # Deploys the AdminServer
  adminserver:
    container_name: adminserver
    image: 1221-app-jms-domain
    restart: always
    environment:
      DS_NAME: "Oracle"
      DS_DB_TYPE: "Oracle"
      DS_DB_NAME: "dockerdb"
      DS_JNDI_NAME: "jdbc/oracleDatabase"
      DS_JDBC_DRIVER: "oracle.jdbc.driver.OracleDriver"
      DS_JDBC_URL:
"jdbc:oracle:thin:system/password123@oracledb:1521:XE"
      DS_DB_HOST: "oracledb"
      DS_DB_PORT: "1521"
      DS_DB_USER: "system"
      DS_DB_PASSWORD: "password123"
    ports:
      - "8001:8001"
    networks:
      - *network

  # Deploys the Database Service
  database:
    container_name: oracledb
    image: oracle/database:11.2.0.2-xe
    restart: always
    shm_size: 1G
    environment:
      ORACLE_PASSWORD: "password123"
      ORACLE_DATABASE: "oracledb"
    ports:
      - "1521:1521"
      - "8080:8080"
    networks:
      - *network

  # WebTier for load balancing requests to cluster
  webtier:
```

```
    image: dockerhub.accenture.com/adop-afpo/oracle-
webtier:1221-webtier-apache
    restart: always
    networks:
      - *network
    environment:
      WEBLOGIC_CLUSTER: "adminserver:8001"
    ports:
- "80:80"
```

- Use Jenkins with Git hooks to build images and run containers automatically for continuous integration
- Use Portainer if you want an easy to use Docker UI

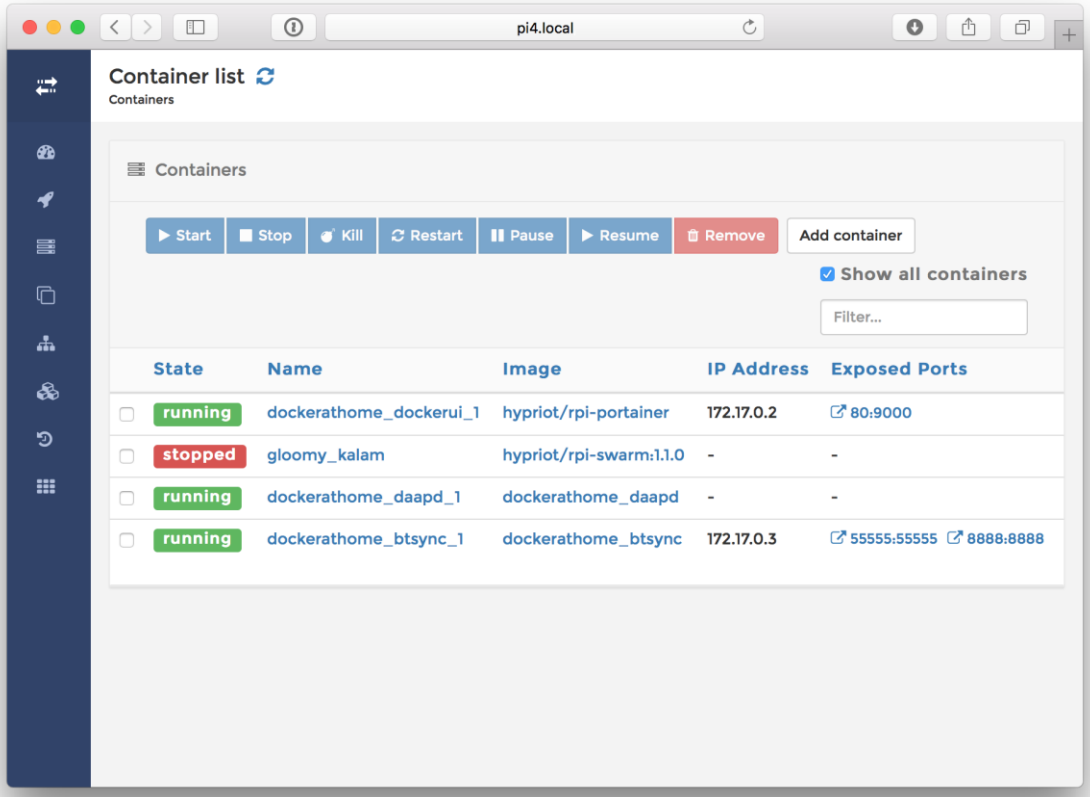

Illustration 4: Portainer UI

Contact address:

Razvan Anghel
virtual7 GmbH
Zeppelinstr. 2
76185 Karlsruhe

Phone: +40 256 222 173
Email razvan.anghel@virtual7.ro
Internet: http://www.virtual7.de