Docker Containers Configuration and Deployment

System Benchmarking and Deployment

2018/2019

The main goal of this guide is to understand how to configure and deploy docker containers running different services/applications.

Useful Docker documentation is available at:

- Docker Documentation - `https://docs.docker.com`
- Step by Step Example - `https://docs.docker.com/get-started/`

Setup

1. Use the VagrantFile, written in the *Virtual Machine Deployment and Configuration* lab guide, to install the needed Docker utilities in both VMs. Docker and Docker-Compose utilities can be installed with the following commands:

```
sudo apt-get install -y apt-transport-https ca-
    certificates curl software-properties-common
curl -fsSL https://download.docker.com/linux/
    ubuntu/gpg | sudo apt-key add -
sudo apt-key fingerprint 0EBFCD88
sudo add-apt-repository "deb [arch=amd64] https
    ://download.docker.com/linux/ubuntu $(
    lsb_release -cs) stable"
sudo apt-get -y update
sudo apt-get -y install docker-ce
sudo curl -L https://github.com/docker/compose/
    releases/download/1.16.1/docker-compose-‘
    uname -s‘-‘uname -m‘ -o /usr/local/bin/docker
    -compose
sudo chmod +x /usr/local/bin/docker-compose
```

2. Deploy the VMs using Vagrant.
3. At the host machine, create the folder *docker_env* which will be used as the base directory for the next steps.

4. Download the sample.war file at `https://tomcat.apache.org/tomcat-6.0-doc/appdev/sample/` to a folder called *myapps*, while this folder should be placed inside the *docker_env* folder. This war file will be used to test the tomcat service that will be deployed during this lab guide.

Steps

Deploying a single Docker Container

1. Setup a Docker configuration file (*Dockerfile*) that will deploy apache tomcat in a Ubuntu 16.04 image. The commands for installing and starting tomcat in ubuntu are:

```
apt-get update && apt-get -y upgrade
apt-get -y install software-properties-common curl
apt-get -y install default-jdk
curl -O http://archive.apache.org/dist/tomcat/
    tomcat-7/v7.0.55/bin/apache-tomcat-7.0.55.tar.
    gz
tar xzf apache-tomcat-7.0.55.tar.gz
apache-tomcat-7.0.55/bin/startup.sh && tail -f
    apache-tomcat-7.0.55/logs/catalina.out
```

2. The Dockerfile should specify a command to copy the *myapps* folder to the path */apache-tomcat-7.0.55/webapps/* at the Docker container.
3. Port 8080 should be exposed to the host.
4. For the previous configurations see the documentation of FROM, RUN, EXPOSE, CMD and COPY Dockerfile commands.
5. Use the command *scp* to copy the *docker_env* folder to the VM with the ip *10.0.0.101*. scp docker_env vagrant@10.0.0.101:
6. At the VM, build the Docker image using the *docker build* command. The image should be named *my/tomcat*. docker build -t my/tomcat .
7. Deploy the container with the command *docker run*. In order to be able to reach tomcat from the host machine use the *-p 8080:8080* flag. To run the container in background use the *-d* flag. docker run -p 8080:8080 my/tomcat
8. Understand the usage of the commands *docker ps, exec, stop, kill. E.g.*, understand and explore the *docker exec -ti "container_id" /bin/bash* command. docker stop é equivalente ao docker kill sem especificação do tipo de sinal
   docker ps, lista os containers
9. Access the tomcat service from the host machine browser (*10.0.0.101:8080/sample*).

docker exec, executa um comando bash dentro de um container

o nó VM2 entra no swarm (colmeia), e pode passar a desempenhar um serviço, ser um <<worker>> (exemplo: hospedar uma base de dados), sendo que a VM1, é nó <<manager>>, ou seja, pode remover/adicionar os outros nós e executar serviços

VM2: sudo docker swarm join --token SWMTKN-1-5xpgtcspe70e8j34gz9981vtqm5c3jhppcit6lqalb41xpkuob-2fx3e av0ht00c3yzlvdjoeruy 10.0.0.101:2377

Docker-Swarm and Compose Utilities

1. Setup a Swarm cluster in which VM1 (10.0.0.101) is the master and VM2 (10.0.0.102) is the worker (`https://docs.docker.com/get-started/part4/#create-a-cluster`). VM1: sudo docker swarm init --advertise-addr 10.0.0.101

2. Understand the usage of the command *docker node* to check if the nodes have successfully joined the Swarm cluster. VM1: sudo docker node ls

3. At VM1, create a docker compose YAML file that specifies a service called *web* that will use two replicas and will export port 8080 to the host (`https://docs.docker.com/get-started/part3/#docker-composeyml`).

4. At VM1, the master node, launch the docker stack with the command *docker stack deploy*. Note that the compose configuration file only needs to be present at the master node. However, since we are not using a docker registry for the tomcat docker image, this image must be present (built) on both VMs.

5. Check that the docker containers are running on both VMs (swarm nodes) with the *docker ps* command.

6. Access the tomcat service from the hosts (*10.0.0.101:8080/sample*) and (*10.0.0.102:8080/sample*)

Learning outcomes Experiment linux Docker containers configuration and deployment. Experiment Docker cluster deployment and configuration. Revise Docker configuration parameters and deployment/management commands.

A criação do swarm é para poder distribuir por diferentes nodes, serviços de uma aplicação

>>> vim docker-compose.yml

```
version: '3'
services:
  web:
image: my/tomcat
  deploy:
    replicas: 2
    build: .
    ports:
  - "8080:8080"
```

>>> sudo docker-compose up -d

HOST: scp -r docker_env vagrant@10.0.0.102:

VM2: sudo docker build -t my/tomcat .