

Containers

Thursday, February 9, 2017

2:44 PM

1. https://zeroturnaround.com/rebellabs/docker-commands-and-best-practices-cheat-sheet/?utm_source=mkto&utm_medium=email&utm_campaign=content_nurture&mk_t_tok=eyJpIjoiT1dGbE56RmtNekptTVRVeilsInQiOil4S1Vzd0N5OUZmVUc3b1p5SEJHY0x6XC9mYTJ1MjBnR2dXUWF3MHFuQWRDYTRcL1NcL2NLdjJsE1cL3ZXTllqR0U5b1NpYVRGVEhvM2xlekRmWkZgdKxPcmFTUGtWYlZBb2dMRkIXVFNkN2RUVWYwOER3XC9RZlhaY1dVcmFGK3QzdW9aIn0%3D
2. <https://www.nginx.com/resources/webinars/building-applications-with-microservices-and-docker/>
3. https://www.brighttalk.com/webcast/9293/160339?utm_campaign=webcasts-search-results-feed&utm_content=Docker&utm_source=brighttalk-portal&utm_medium=web
4. https://www.brighttalk.com/webcast/14775/238505?utm_medium=web&utm_source=brighttalk-portal&utm_campaign=player-page-feed

Docker for Java Developers

1. <https://github.com/docker/labs/blob/master/swarm-mode/beginner-tutorial/README.md>
2. Docker provides PODA (Package Once Deploy Anywhere) and complements WORA (Write Once Run Anywhere) provided by Java. It also helps you reduce the impedance mismatch between dev, test, and production environment and simplifies Java application deployment. A lot of new features have been introduced in Docker 1.12 such as Swarm Mode, service discovery, load balancing, Docker for AWS, and many others. This session will walk you through some of the new features in a code-intensive session.
3. Docker gives you portable image (based on a base OS which creating the image) format in which we can run any application
4. Builds, ships & run distributed applications

5. DockerFile.txt
 - a. From ubuntu (first non comment instruction tell which is the base OS to create this docker image)
 - b. Typically last thing is the command to run ' CMD echo "Hello World"'
6. Docker Toolbox
 - a. CLI
 - b. Machine
 - c. Compose
 - d. Kitematic
 - e. ...
7. Docker Machine
 - a. Creating Docker Host on computer or cloud provider (machines that do not have native Linux capabilities). Configuring a machine as a docker engine.
 - i. Docker-machine create --driver=virtualbox myhost
 - 1) Configure docker client to talk to the host
 - 2) Create and pull images
 - 3) Upgrade docker
 - 4) Start/stop/restart docker (containers)
8. Docker Compose
 - a. Define and run multi-container applications
 - b. Define configuration in single or more files
 - i. docker-compose.yaml. It's a default one
 - ii. docker-compose.override.yaml. It's a default one
 - iii. Specify multiple files using -f
 - iv. All Paths relative to base configuration file
 - v. Great for dev, staging & CI
 - c. Docker-compose up -d (detach mode)
9. Swarm mode
 - a. Cluster of docker images
 - b. Have to enable it
 - c. Self-Organizing & Healing
 - d. **No SPOF (Single Point of Failure)**
 - e. **Scaling**
 - f. Optional Feature
 - g. Commands
 - i. Docker swarm init --listen-addr:<ip>:2377 --secret:<SECRET> (first node in Swarm is always master)
 - 1) Primary/Secondary Masters

1) Primary/Secondary masters

2) Can configure master to not run any container. Primarily for administrative purposes

ii. Docker swarm join --secret:<SECRET> <manager>:2377 (add worker)

Swarm Mode using Docker Machine

Task	Command
Create Manger	<code>docker-machine create -d virtualbox managerX</code>
Create worker	<code>docker-machine create -d virtualbox workerX</code>
h. Initialize Swarm mode	<code>docker swarm init --listen-addr <ip1> --advertise-addr <ip1></code>
Manager token	<code>docker swarm join-token manager -q</code>
Worker token	<code>docker swarm join-token worker -q</code>
Manager X join	<code>docker swarm join --token m_token --listen-addr <ipX> --advertise-addr <ipX> <ip1></code>
Worker X join	<code>docker swarm join --token w_token --listen-addr <ipX> --advertise-addr <ipX> <ip1></code>

i. It has routing mesh (can do load balancing)

10. Docker commands

a. Docker images openjdk

b. Docker history openjdk (how this image was built)

11. Dockerfile

Table 2-1. Common commands for Dockerfiles

Command	Purpose	Example
FROM	First noncomment instruction in the Dockerfile	FROM ubuntu
COPY	Copies multiple source files from the context to the filesystem of the container at the specified path	COPY .bash_profile / home
ENV	Sets the environment variable	ENV HOSTNAME=test
a. RUN	Executes a command	RUN apt-get update
CMD	Default for an executing container	CMD ["/bin/echo", "hello world"]
EXPOSE	Informs the network ports that the container will listen on	EXPOSE 8093

Any line starting with # in the Dockerfile is treated as a comment and not processed.

12. .dockerignore

13. Docker Version manager - <https://github.com/getcarina/dvm>

14. Docker Toolbox and Docker for Mac coexistence

a. You can use Docker for Mac and Docker Toolbox together on the

18. Running Docker Container

- a. Docker run hello-java
- b. The docker run command has multiple options that can be specified to customize the container. Multiple options can be combined together.

Option	Purpose
-i	Keep STDIN open even if not attached
-t	Allocate a pseudo-TTY
-d	Run container in background and print container ID
--name	Assign a name to the container
-c	Automatically remove the container when it exits
-e	Set environment variable
-P	Publish all exposed ports to random ports on the host
-p	Publish a container's port(s) to the specified host port
-m	Limit the memory

- d. The following command runs the container in the background, gives it a name, and automatically removes it when it exits:
 - i. `docker run --name hello -d --rm hello-java`

Table 2-3. Common Docker Hub commands

Command	Purpose
login	Register or log in to a Docker registry
search	Search the Docker Hub for images
pull	Pull an image or a repository from a registry
push	Push an image or a repository to a registry
logout	Log out from a Docker registry
tag	Tag an image into a repository

20. Defining the Application Using the Compose File

- The Compose file definition for our Java EE application is defined here

```

version: '2'
services:
  db: ❶
    container_name: "db" ❸
    image: arungupta/oreilly-couchbase ❷ ❹
    ports: ❸
      - 8091:8091
      - 8092:8092
      - 8093:8093
      - 11210:11210
  web: ❶
    image: arungupta/oreilly-wildfly ❷ ❺
    depends_on: ❷
      - db
    environment: ❸
      - COUCHBASE_URI=db
    ports: ❸
      - 8080:8080

```

This file is available on [GitHub](#). This Compose file has:

- ❶ Two services named db and web.
- ❷ An image name for each service defined using the image attribute.
- ❸ Port forwarding from host to container defined using the ports attribute.
- ❹ The arungupta/oreilly-couchbase image, which starts the Couchbase server and configures it using [Couchbase REST API](#). The image is built using [this Dockerfile](#).
- ❺ The arungupta/oreilly-wildfly image, which starts the WildFly application server and deploys the WAR file built from [GitHub](#). The image is built using [this Dockerfile](#).

This Java EE application uses JAX-RS to publish REST endpoints over JSON documents stored in the Couchbase server.

- ❸ The Couchbase container is given the name db using container_name attribute.
- ❷ depends_on requires the web service to start before the db service. This will create a dependency between the containers but the application-level health check is still up to the user. For example, the Couchbase database may not be completely ready before the application is deployed in WildFly. In this case, the web service needs to be restarted as `docker-compose restart web`.
- ❸ COUCHBASE_URI environment variable identifies the database container to be used by this web container. This environment variable is then used in the application to connect to the Couchbase server using the [Couchbase Java SDK API](#).

```

$ docker run --name static-site -e AUTHOR="Your Name" -d -P sequence/static-site
e61d12292d69556eabe2a44c16cbd54486b2527e2ce4f95438e504afb7b02810

```

In the above command:

21.

- `-d` will create a container with the process detached from our terminal
- `-P` will publish all the exposed container ports to random ports on the Docker host
- `-e` is how you pass environment variables to the container
- `--name` allows you to specify a container name
- `AUTHOR` is the environment variable name and `Your Name` is the value that you can pass

22. Docker containers will consume disk-space and if you want to determine how much disk-space each of the containers on your computer use, issue the following command:

- a. `sudo docker ps -a -s`

23. **Binding Volumes**

- Three directories in a Docker container bound to three directories in the host.
 - Launch the Docker container with the command below.
The `-v` option tells Docker that we want to make the contents of a directory in the host available at a certain path in the Docker container file-system.
The `-d` option runs the container in the background and the terminal prompt will be available as soon as the id of the newly launched Docker container has been printed.
 - `sudo docker run -d -v ~/mule-root/apps:/opt/mule/apps -v ~/mule-root/conf:/opt/mule/conf -v ~/mule-root/logs:/opt/mule/logs codingtony/mule`

24. **Direct Access to Docker Container Files**

- Start a new Docker container: `sudo docker run -d codingtony/mule`
- Find the id of the newly launched Docker container: `sudo docker ps`
- Examine low-level information about the newly launched Docker container: `sudo docker inspect [container id or name here]`

```
[{
  "AppArmorProfile": "",
  "Args": [],
  "Config": {
    ...
  },
  "Created": "2015-01-12T07:58:47.913905369Z",
  "Driver": "aufs",
  "ExecDriver": "native-0.2",
  "HostConfig": {
    ...
  },
  "HostnamePath": "/var/lib/docker/containers/68b40def7ad6a7f819bd654d5627ad1c3a0f40c84e0fb0f875760f1bd6790eef",
  "HostsPath": "/var/lib/docker/containers/68b40def7ad6a7f819bd654d5627ad1c3a0f40c84e0fb0f875760f1bd6790eef",
  "Id": "68b40def7ad6a7f819bd654d5627ad1c3a0f40c84e0fb0f875760f1bd6790eef",
  "Image": "bcd0f37d48d4501ad64bae941d95446b157a6f15e31251e26918dbac542d731f",
  "MountLabel": "",
  "Name": "/thirsty_darwin",
```



```

    "NetworkSettings": {
      ...
    },
    "Path": "/opt/mule/bin/mule",
    "ProcessLabel": "",
    "ResolvConfPath": "/var/lib/docker/containers/68b40def7ad6a7f819bd654d5627ad1c3a0f40c84e0fb0f875760f1bd6
    "State": {
      ...
    },
    "Volumes": {},
    "VolumesRW": {}
  }
}

```

- Locate the “Driver” node (highlighted in the above output) and ensure that its value is “aufs”.
If it is not, you may need to modify the directory paths below replacing “aufs” with the value of this node. Personally I have only seen the “aufs” value at this node so anything else is uncharted territory to me.
- Copy the long hexadecimal value that can be found at the “Id” node (also highlighted in the above output).
This is the long id of the Docker container.
- In a terminal window, issue the following command, inserting the long id of your container where noted:
 - `sudo ls -al /var/lib/docker/aufs/mnt/[long container id here]`
- You are now looking at the root of the volume used by the Docker container you just launched.
 - In the same terminal window, issue the following command:
 - `sudo ls -al /var/lib/docker/aufs/mnt/[long container id here]/opt`
 - ◆ The output from this command should look like this:

```

total 12
drwxr-xr-x  4 root root 4096 jan 12 15:58 .
drwxr-xr-x 75 root root 4096 jan 12 15:58 ..
lrwxrwxrwx  1 root root   26 aug 10 04:19 mule -> /opt/mule-standalone-3.5.0
drwxr-xr-x 17  409  409 4096 jan 12 15:58 mule-standalone-3.5.0

```

25. Entering a Docker Container

- The `exec` Docker command is used to run a command, *bash* in this case, in a running Docker container. The `-i` flags tell Docker to keep the input stream open while the command is being executed. In this example, it allows us to enter commands into the bash shell running inside the Docker container. The `-t` flag cause Docker to allocate a text terminal to which the output from the command execution is printed.
 - `sudo docker exec -i -t [container id or name here] bash`
 - Note the prompt, which should change to `[user]@[Docker container id]`. In my case it looks like this:
 - `root@3ea374a280da:/#`
 - `sudo docker inspect [container id or name here] | grep IPAddress`

- `sudo docker inspect [container id or name here] | grep IP Address`
(getting ip address of the container)

26. Accessing Endpoints

- `sudo docker run -d -p 8181:8181 -v ~/mule-root/apps:/opt/mule/apps -v ~/mule-root/conf:/opt/mule/conf -v ~/mule-root/logs:/opt/mule/logs codingtony/mule`
- Using the `-p` flag, we have seen that we can expose a service in a Docker container so that it becomes accessible from outside of the host computer

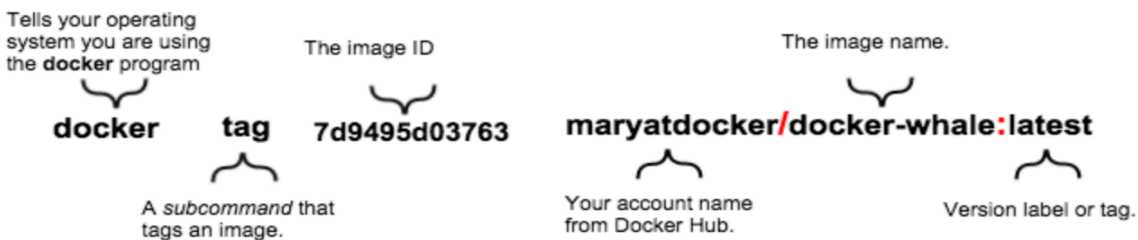
27. Dockerfile for Mule

- FROM codingtony/java
- MAINTAINER tony.bussieres@ticksmith.com
- RUN `wget https://repository.mulesoft.org/nexus/content/repositories/releases/org/mule/distributions/mule-standalone/3.5.0/mule-standalone-3.5.0.tar.gz`
- RUN `cd /opt && tar xvf ~/mule-standalone-3.5.0.tar.gz`
- RUN `echo "4a94356f7401ac8be30a992a414ca9b9/mule-standalone-3.5.0.tar.gz" | md5sum -c`
- RUN `rm ~/mule-standalone-3.5.0.tar.gz`
- RUN `ln -s /opt/mule-standalone-3.5.0/opt/mule`
- CMD `["/opt/mule/bin/mule"]`

Important Commands

- `docker ps -a -q`
 - lookout for all running containers and return their Id's
- `docker rm -f $(docker ps -a -q)`
 - Remove all running containers
- `docker run --name java-8u91 -d java:8u91-jdk`
 - Running a docker image in a detached mode
- `docker inspect 618fe3c045c3`
 - Inspect a particular container
- `sudo docker exec -i -t 618fe3c045c3 bash`
 - Run command from docker host
- `docker ps -a`
 - List of all containers
- `docker build -t java-8u91-jdk .`
 - Build the docker image. Make sure Dockerfile exists in current dir as shown by `'.'`
- `docker ps -a -s`
 - List of all containers and their sizes
- `docker images`
 - List of all docker images
- `docker rmi -f $(docker images -a -q)`

- a. To clear all images
- 11. `docker volume rm $(docker volume ls -q)`
 - a. To clear all Volumes
- 12. `docker network rm $(docker network ls | tail -n+2 | awk '{if($2 !~ /bridge|none|host/){print $1}}')`
 - a. To clear Networks
- 13. `sudo service docker restart`
- 14. `docker tag 7d9495d03763 maryatdocker/docker-whale:latest`

15. 

- 16. `docker login`
 - a. Login for docker HUB
- 17. `docker push maryatdocker/docker-whale`
 - a. Push to docker HUB
- 18. `docker exec -it c55ffef28b70 /bin/bash`
 - a. c55ffef28b70 is the container Id

Mule Containerization

1. Create a folder 'mule-ee' to hold key files
 - a. Copy 'mule-ee-distribution-standalone-3.8.3.zip' artifact in 'mule-ee'
 - b. create 'conf' folder
 - i. Copy license-mule.lic in 'conf'
 - ii. Copy any other props you app might need
 - iii. Optional: provide wrapper.conf
 - c. Dockerfile
 - d. build.sh
2. Run './build.sh' from 'mule-ee' folder
 - a. It will create the image
3. `docker tag e3a623d8252f bulandmalik/mule_ee_3.8.3:latest`
 - a. Tag it to be pushed to Docker Hub
4. `docker login`
5. `Docker push bulandmalik/mule_ee_3.8.3:latest`
6. Running it
 - a. `docker run -t -i --name='mule-ee-node1' bulandmalik/mule_ee_3.8.3`
 - b. Run it by creating symbolic links
 - i. `sudo docker run -t -i --name='mule-ee-node1' -d -v ~/work/study/docker/mule/mule-ee/apps:/opt/mule-ee-3.8.3/apps -v ~/work/study/docker/mule/mule-`

```
ee/conf:/opt/mule-ee-3.8.3/conf -v ~/work/study/docker/mule-  
ee-3.8.3/mule-ee/logs:/opt/mule-ee-3.8.3/logs  
bulandmalik/mule_ee_3.8.3
```

ii. ~~sudo docker run \$(docker cp ~/work/study/docker/mule/mule-
ee/conf mule-ee-node1:/opt/mule-ee-3.8.3/conf) -t -i --
name='mule-ee-node1' -d -v ~/work/study/docker/mule/mule-
ee/apps:/opt/mule-ee-3.8.3/apps -v
~/work/study/docker/mule/mule-ee/logs:/opt/mule-ee-3.8.3/logs
mule_ee_3.8.3~~

iii. sudo docker run -t -i --name='mule-ee-node1' -d -v
~/work/study/docker/mule/mule-ee/apps:/opt/mule-
ee-3.8.3/apps -v ~/work/study/docker/mule/mule-
ee/logs:/opt/mule-ee-3.8.3/logs mule_ee_3.8.3

iv. sudo docker run -t -i -p 9761:9761 --name='mule-ee-node1' -
d -v ~/work/study/docker/mule/mule-ee/apps:/opt/mule-
ee-3.8.3/apps -v ~/work/study/docker/mule/mule-
ee/logs:/opt/mule-ee-3.8.3/logs mule_ee_3.8.3

1) The added ag -p 9761:9761 makes the service exposed at port 9761 in
the Docker container available at port 9761 in the host.

7. docker exec -it 7fe479b81335 /bin/bash

8. docker inspect cad206e3a4c9 | grep Status

9. docker inspect cad206e3a4c9 | grep Driver

10. docker logs mule-ee-node1

a. To check the logs to see what went wrong

11. docker rm \$(docker ps -a -q)

a. Remove all stopped containers

12. Curl Commands

a. curl --header "Authorization:

Intuit_IAM_Authentication intuit_token_type=IAM-

Ticket,intuit_appid=Intuit.cto.SUBS_Service,intuit_app_secret=prep

rd8ZR0vV9GP7IcrGFSMlsNZ5SRm45WWRUUum,intuit_userid=

123146275296154,intuit_token=V1-145-a3td5pfzbbjpyodr38lndq"

<https://accounts->

[e2e.platform.intuit.net:443/v1/realms/123146275295789](https://accounts-e2e.platform.intuit.net:443/v1/realms/123146275295789)