ABOUT ARUN GUPTA

Arun is a technology enthusiast, avid runner, author of a best-selling book, globe trotter, a community guy, Java Champion, JavaOne Rockstar, JUG Leader, Minecraft Modder, Devoxx4Kids-er, and a Red Hatter.

Microservice using Docker stack deploy – WildFly, Java EE and Couchbase

Posted by: Arun Gupta in Enterprise Java February 6th, 2017

There is plenty of material on microservices, just google it! I gave a presentation on refactoring monolith to microservices at Devoxx Belgium a couple of years back and it has good reviews:

Refactor your Java EE application using Microservices and Contain…

This blog will show how Docker simplifies creation and shutting down of a microservice.

All code used in this blog is at github.com/arun-gupta/couchbase-javaee.

Microservice Definition using Compose

Docker 1.13 introduced a v3 of Docker Compose. The changes in the syntax are minimal but the key difference is addition of deploy attribute. This attribute allows to specify replicas, rolling update and restart policy for the container.

Our microservice will start a WldFly application server with a Java EE application pre-deployed. This application will talk to a Couchbase database to CRUD application data.

Here is the Compose definition:

```
version: '3'
services:
  web:
    image: arungupta/couchbase-javaee:travel
    environment:
      - COUCHBASE_URI=db
    ports:
      - 8080:8080
      - 9990:9990
    depends_on:
      - db
  db:
    image: arungupta/couchbase:travel
    ports:
      - 8091:8091
      - 8092:8092
      - 8093:8093
      - 11210:11210
```

In this Compose file:

1. Two services in this Compose are defined by the name
   `db`
   and
   `web`
   attributes
2. Image name for each service defined using
   `image`
   attribute
3. The
   `arungupta/couchbase:travel`
   image starts Couchbase server, configures it using Couchbase REST API, and loads
   `travel-sample`
   bucket with ~32k JSON documents.
4. The
   `arungupta/couchbase-javaee:travel`
   image starts WildFly and deploys application WAR file built from https://github.com/arun-gupta/couchbase-javaee. Clone that project if you want to build your own image.
   `envrionment`
   attribute defines environment variables accessible by the application deployed in WildFly.
   `COUCHBASE_URI`
   refers to the database service. This is used in the application code as shown at https://github.com/arun-gupta/couchbase-javaee/blob/master/src/main/java/org/couchbase/sample/javaee/Database.java.
6. Port forwarding is achieved using
   `ports`
   attribute
   `depends_on`
   attribute in Compose definition file ensures the container start up order. But application-level start up needs to be ensured by the applications running inside container. In our case, WildFly starts up rather quickly but takes a few seconds for the database to start up. This means the Java EE application deployed in WildFly is not able to communicate with the database. This outlines a best practice when building micro services applications: you must code defensively and ensure in your application initialization that the micro services you depend on have started, without assuming startup order. This is shown in the database initialization code at https://github.com/arun-gupta/couchbase-javaee/blob/master/src/main/java/org/couchbase/sample/javaee/Database.java. It performs the following checks:

VIEW ALL

JOIN US

With 1,240,600 mo unique visitors and ov 500 authors we are placed among the top related sites around. Constantly being on t lookout for partners; encourage you to join So If you have a blog unique and interesting content then you shou check out our **JCG** partners program. You can be a **guest writer** for Java Code Geeks and h your writing skills!

1. Bucket exists
2. Query service of Couchbase is up and running
3. Sample bucket is fully loaded

This application can be started using

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

command on a single host. Or a cluster of Docker engines in swarm-mode using

```
docker stack deploy
```

command.

Setup Docker Swarm-mode

Initialize Swarm mode using the following command:

```
docker swarm init
```

This starts a Swarm Manager. By default, manager node are also worker but can be configured to be manager-only.

Find some information about this one-node cluster using the command

```
docker info
```

command:

```
Containers: 0
 Running: 0
 Paused: 0
 Stopped: 0
Images: 17
Server Version: 1.13.0
Storage Driver: overlay2
 Backing Filesystem: extfs
 Supports d_type: true
 Native Overlay Diff: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
 Volume: local
 Network: bridge host ipvlan macvlan null overlay
Swarm: active
 NodeID: 92mydh0e09ba5hx3wtmcmvktz
 Is Manager: true
 ClusterID: v68ikyaff7rdxpaw1j0c9i60s
 Managers: 1
 Nodes: 1
 Orchestration:
  Task History Retention Limit: 5
 Raft:
  Snapshot Interval: 10000
  Number of Old Snapshots to Retain: 0
  Heartbeat Tick: 1
  Election Tick: 3
 Dispatcher:
  Heartbeat Period: 5 seconds
 CA Configuration:
  Expiry Duration: 3 months
 Node Address: 192.168.65.2
 Manager Addresses:
  192.168.65.2:2377
Runtimes: runc
Default Runtime: runc
Init Binary: docker-init
containerd version: 03e5862ec0d8d3b3f750e19fca3ee367e13c090e
runc version: 2f7393a47307a16f8cee44a37b262e8b81021e3e
init version: 949e6fa
Security Options:
 seccomp
  Profile: default
Kernel Version: 4.9.5-moby
Operating System: Alpine Linux v3.5
OSType: linux
Architecture: x86_64
CPUs: 4
Total Memory: 1.952 GiB
Name: moby
ID: SGCM:KDRD:G3M7:PZHN:J4RL:VFFR:G2SR:EKD5:JV4J:RL3X:LF7T:XF6V
Docker Root Dir: /var/lib/docker
Debug Mode (client): false
Debug Mode (server): true
 File Descriptors: 31
 Goroutines: 124
 System Time: 2017-01-27T08:25:58.032295342Z
 EventsListeners: 1
```

```
No Proxy: *.local, 169.254/16
Username: arungupta
Registry: https://index.docker.io/v1/
Experimental: true
Insecure Registries:
 127.0.0.0/8
Live Restore Enabled: false
```

This cluster has 1 node, and that is manager.

Alternatively, a multi-host cluster can be easily setup using Docker for AWS.

Deploy Microservice

The microservice can be started as:

```
docker stack deploy --compose-file=docker-compose.yml webapp
```

This shows the output:

```
Creating network webapp_default
Creating service webapp_web
Creating service webapp_db
```

WildFly and Couchbase services are started on this node. Each service has a single container. If the Swarm mode is enabled on multiple nodes then the containers will be distributed across multiple nodes.

A new overlay network is created. This allows multiple containers on different hosts to communicate with each other.
Verify that the WildFly and Couchbase services are running using

```
docker service ls
```

:

```
ID            NAME        MODE        REPLICAS  IMAGE
a9pkiziw3vgw  webapp_db   replicated  1/1       arungupta/couchbase:travel
hr5s6ue54kwj  webapp_web  replicated  1/1       arungupta/couchbase-javaee:travel
```

Logs for the service can be seen using

```
docker service logs -f webapp_web
```

:

```
webapp_web.1.wby0b04t7bap@moby    | =======================================================================
webapp_web.1.wby0b04t7bap@moby    |
webapp_web.1.wby0b04t7bap@moby    |   JBoss Bootstrap Environment
webapp_web.1.wby0b04t7bap@moby    |
webapp_web.1.wby0b04t7bap@moby    |   JBOSS_HOME: /opt/jboss/wildfly
webapp_web.1.wby0b04t7bap@moby    |
webapp_web.1.wby0b04t7bap@moby    |   JAVA: /usr/lib/jvm/java/bin/java
webapp_web.1.wby0b04t7bap@moby    |
webapp_web.1.wby0b04t7bap@moby    |   JAVA_OPTS:  -server -Xms64m -Xmx512m -XX:MetaspaceSize=96M -XX:MaxMet
webapp_web.1.wby0b04t7bap@moby    |
webapp_web.1.wby0b04t7bap@moby    | =======================================================================

. . .

webapp_web.1.wby0b04t7bap@moby    | 23:14:15,811 INFO  [org.jboss.as.server] (ServerService Thread Pool --
webapp_web.1.wby0b04t7bap@moby    | 23:14:16,076 INFO  [org.jboss.as] (Controller Boot Thread) WFLYSRV0060:
webapp_web.1.wby0b04t7bap@moby    | 23:14:16,077 INFO  [org.jboss.as] (Controller Boot Thread) WFLYSRV0051:
webapp_web.1.wby0b04t7bap@moby    | 23:14:16,077 INFO  [org.jboss.as] (Controller Boot Thread) WFLYSRV0025:
```

Make sure to wait for the last log statement to show.

Access Microservice

Get 10 airlines from the microservice:

```
curl -v http://localhost:8080/airlines/resources/airline
```

This shows the results as:

```
*   Trying ::1...
* Connected to localhost (::1) port 8080 (#0)
> GET /airlines/resources/airline HTTP/1.1
> Host: localhost:8080
> User-Agent: curl/7.43.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Connection: keep-alive
< X-Powered-By: Undertow/1
< Server: WildFly/10
< Content-Type: application/octet-stream
< Content-Length: 1402
< Date: Fri, 03 Feb 2017 17:02:45 GMT
<
* Connection #0 to host localhost left intact
[{"travel-sample":{"country":"United States","iata":"Q5","callsign":"MILE-AIR","name":"40-Mile Air","icao":
```

Docker for Java Developers workshop is a self-paced hands-on lab and allows you to get started with Docker easily.

Get a single resource:

```
curl -v http://localhost:8080/airlines/resources/airline/137
```

Create a new resource:

```
curl -v -H "Content-Type: application/json" -X POST -d
'{"country":"France","iata":"A5","callsign":"AIRLINAIR","name":"Airlinair","icao":"RLA","type":"airline"}'
http://localhost:8080/airlines/resources/airline
```

Update a resource:

```
curl -v -H "Content-Type: application/json" -X PUT -d
'{"country":"France","iata":"A5","callsign":"AIRLINAIR","name":"Airlin
Air","icao":"RLA","type":"airline","id": "19810"}' http://localhost:8080/airlines/resources/airline/19810
```

Delete a resource:

```
curl -v -X DELETE http://localhost:8080/airlines/resources/airline/19810
```

Detailed output from each of these commands is at github.com/arun-gupta/couchbase-javaee.

Delete Microservice

The microservice can be removed using the command

```
docker stack rm webapp
```

:

```
Removing service webapp_web
Removing service webapp_db
Removing network webapp_default
```

Want to get started with Couchbase? Look at Couchbase Starter Kits.

Want to learn more about running Couchbase in containers?

- Couchbase on Containers
- Couchbase Forums
- Couchbase Developer Portal
- @couchhasedev and @couchbase

Reference: Microservice using Docker stack deploy – WildFly, Java EE and Couchbase from our JCG partner Arun Gupta at the Miles to go 3.0 … blog.

Tagged with: COUCHBASE JBOSS WILDFLY MICROSERVICES and many more ....

Email address:

Your email address

Sign up

"Emails are the best way to communicate with our customers."
- MATT JUSZCAK Co-Owner, Turnstyle Cycle
BE A MARKETER, LIKE MATT
Constant Contact Email Marketing makes it easy.
Constant Contact
TRY EMAIL FREE

LEAVE A REPLY

Your email address will not be published. Required fields are marked *

Name *

Email *

Website

✓ Sign me up for the newsletter!

Receive Email Notifications?

no, do not subscribe | instantly

Or, you can subscribe without commenting.

Post Comment

KNOWLEDGE BASE

- Courses
- Examples
- Resources
- Tutorials
- Whitepapers

PARTNERS

- Mkyong

HALL OF FAME

- "Android Full Application Tutorial" series
- 11 Online Learning websites that you should check out
- Advantages and Disadvantages of Cloud Computing – Cloud computing pros and cons
- Android Google Maps Tutorial
- Android JSON Parsing with Gson Tutorial
- Android Location Based Services Application – GPS location

ABOUT JAVA CODE GEEKS

JCGs (Java Code Geeks) is an independent online community focused on creating the ultimate Java to Java developers resource center; targeted at the technical architect, technical team lead (senior developer), project manager and junior developers alike. JCGs serve the Java, SOA, Agile and Telecom communities with daily news written by domain experts, articles, tutorials, reviews, announcements, code snippets and open source projects.

DISCLAIMER

THE CODE GEEKS NETWORK

- .NET Code Geeks
- Java Code Geeks
- System Code Geeks
- Web Code Geeks

- Android Quick Preferences Tutorial
- Difference between Comparator and Comparable in Java
- GWT 2 Spring 3 JPA 2 Hibernate 3.5 Tutorial
- Java Best Practices – Vector vs ArrayList vs HashSet