Aron Bahram
PM5273

Deploying a Containerized Java Spring Boot Web Application Using AWS

Technologies of Virtualized Networks and Data Centers Project

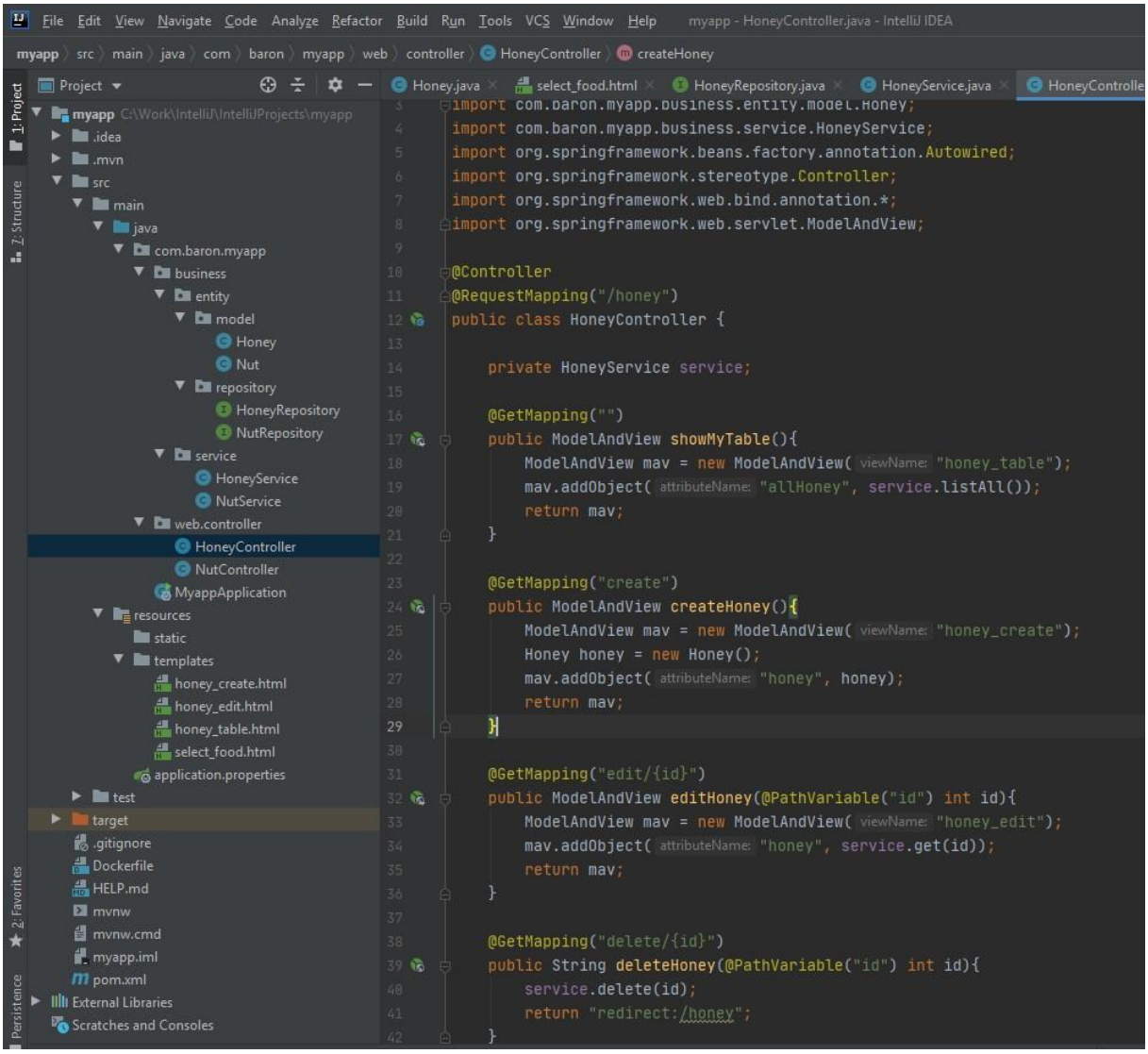
Project Steps

1. Create the Java Spring Boot web application
2. Initialize an Amazon RDS instance
3. Connect the RDS to the web application
4. Build a working Docker Image and push it to Docker Hub
5. Create an Amazon EC2 instance
6. Download and run necessary programs on the EC2 instance
7. Deploy the web application on the EC2 instance in a Docker container

Create the Java Spring Boot web application

I created a Spring Boot application using the Spring Initializer in IntelliJ. With this, all dependencies were downloaded automatically, and the project was configured as a web application.

I used the MVC pattern to separate the layers in the application. Then wrote the code with a dummy database.

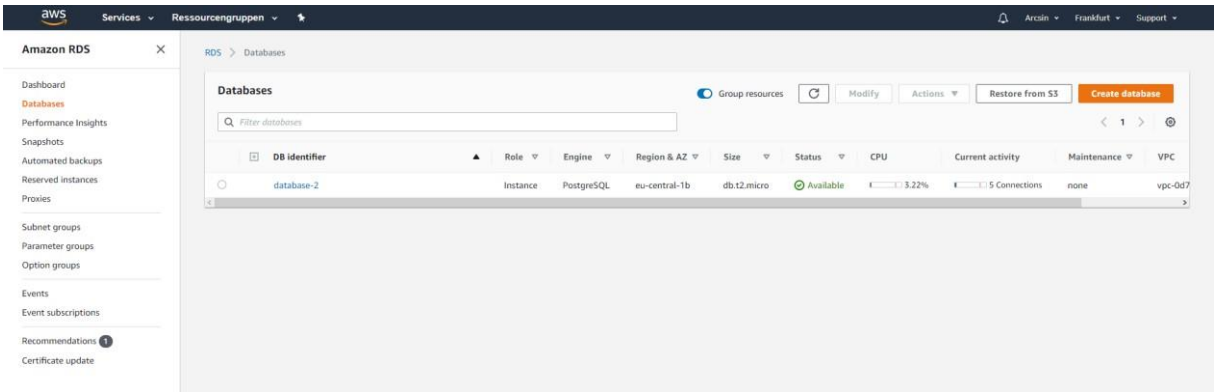


Next Step…

Aron Bahram
PM5273

Create Amazon Relational Database Service

I created a PostgreSQL base Amazon RDS instance.



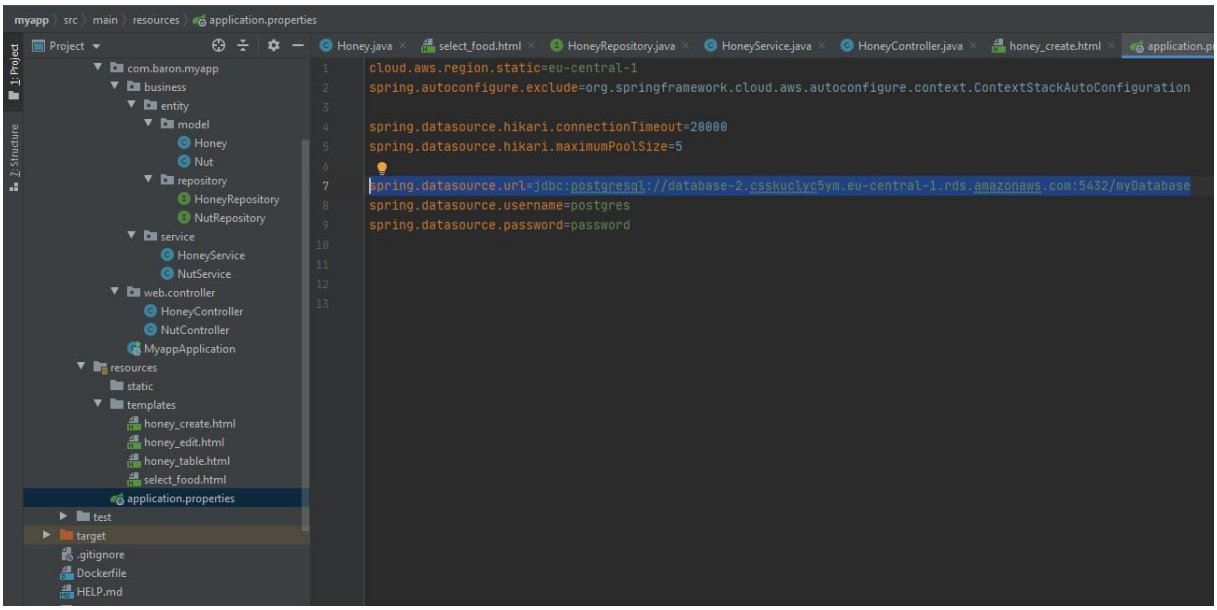
Then I connected to this Database from my Laptop using an SQL shell (psql). After successful connection, I created the “honey” table on this instance.

```
SQL Shell (psql)
Server [localhost]: database-2.csskuclyc5ym.eu-central-1.rds.amazonaws.com
Database [postgres]: myDatabase
Port [5432]:
Username [postgres]:
Passwort für Benutzer postgres:
psql (12.2, Server 11.6)
Warnung: Konsolencodeseite (850) unterscheidet sich von der Windows-
         Codeseite (1252). 8-Bit-Zeichen funktionieren möglicherweise nicht
         richtig. Einzelheiten finden Sie auf der psql-Handbuchseite unter
         »Notes for Windows users«.
SSL-Verbindung (Protokoll: TLSv1.2, Verschlüsselungsmethode: ECDHE-RSA-AES256-GCM-SHA384, Bits: 256, Komprimierung: aus)

Geben Sie »help« für Hilfe ein.

myDatabase=> \d
         Liste der Relationen
 Schema | Name  |   Typ   | Eigent³mer
--------+-------+---------+------------
 public | honey | Tabelle | postgres
 public | test  | Tabelle | postgres
(2 Zeilen)


myDatabase=> \d honey
                              Tabelle ¬public.honey%
    Spalte     |          Typ           | Sortierfolge | NULL erlaubt? | Vorgabewert
---------------+------------------------+--------------+---------------+-------------
 id            | integer                |              | not null      |
 type          | character varying(255) |              |               |
 flower        | character varying(255) |              |               |
 healthbenefit | character varying(255) |              |               |
 vitamins      | character varying(255) |              |               |
 tastiness     | integer                |              |               |
Indexe:
    "honey_pkey" PRIMARY KEY, btree (id)
```

Next Step…

Connect to the Relational Database Service

Using application.properties file I could describe the connection to the amazon RDS instance.



Next Step…

Push to Docker Hub

Creating the Dockerfile. Based on this file, the Docker Image will be created.

```
FROM openjdk:8
EXPOSE 8080
ADD target/myapp.jar myapp.jar
ENTRYPOINT ["java", "-jar", "/myapp.jar"]
```

Building the Docker Image. –t defines the tag/name of the image.

```
Administrator: Eingabeaufforderung
C:\Work\IntelliJ\IntelliJProjects\myapp>docker build -t myapp .
Sending build context to Docker daemon  51.18MB
Step 1/4 : FROM openjdk:8
8: Pulling from library/openjdk
376057ac6fa1: Pull complete
5a63a0a859d8: Pull complete
496548a8c952: Pull complete
2adae3950d4d: Pull complete
0a297eafb9ac: Pull complete
51607e1c215b: Pull complete
f7c02da203f1: Pull complete
Digest: sha256:a101b81a950114863da9db3b64e8de73a5c98641f53abbf6fe152fe45b48ba96
Status: Downloaded newer image for openjdk:8
 ---> 1077d23b5882
Step 2/4 : EXPOSE 8080
 ---> Running in 6b253cf6c48c
Removing intermediate container 6b253cf6c48c
 ---> 9159ea2598ac
Step 3/4 : ADD target/myapp.jar myapp.jar
 ---> c4e4d0709ba6
Step 4/4 : ENTRYPOINT ["java", "-jar", "/myapp.jar"]
 ---> Running in 6480fdb4a130
Removing intermediate container 6480fdb4a130
 ---> 769da98b7ad4
Successfully built 769da98b7ad4
Successfully tagged myapp:latest
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and director
ies added to build context will have '-rwxr-xr-x' permissions. It is recommended to double check and reset permissions f
or sensitive files and directories.
```

Running the Docker image. With –p I defined the port 8080 on my host machine to connect to the port 8080 on the container.

```
C:\Work\IntelliJ\IntelliJProjects\myapp>docker run -p 8080:8080 myapp

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.3.0.RELEASE)

2020-05-25 13:14:45.962  INFO 1 --- [           main] com.baron.myapp.MyappApplication         : Starting MyappApplicati
on v0.0.1-SNAPSHOT on 62001f2b0bdb with PID 1 (/myapp.jar started by root in /)
2020-05-25 13:14:45.968  INFO 1 --- [           main] com.baron.myapp.MyappApplication         : No active profile set,
falling back to default profiles: default
2020-05-25 13:14:51.956  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Multiple Spring Data mo
dules found, entering strict repository configuration mode!
2020-05-25 13:14:51.959  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Da
ta JDBC repositories in DEFAULT mode.
2020-05-25 13:14:52.104  INFO 1 --- [           main] .RepositoryConfigurationExtensionSupport : Spring Data JDBC - Coul
d not safely identify store assignment for repository candidate interface com.baron.myapp.business.entity.repository.Hor
eyRepository. If you want this repository to be a JDBC repository, consider annotating your entities with one of these a
nnotations: org.springframework.data.relational.core.mapping.Table.
```

I can display the images created with this command:

```
C:\Work\IntelliJ\IntelliJProjects\myapp>docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
myapp               latest              769da98b7ad4        51 minutes ago      561MB
openjdk             8                   1077d23b5882        9 days ago          510MB
```

The openjdk image is downloaded with the myapp image since myapp depends on it. The dependency was defined in the Dockerfile (FROM openjdk:8). With the docker ps command I can check the running containers.

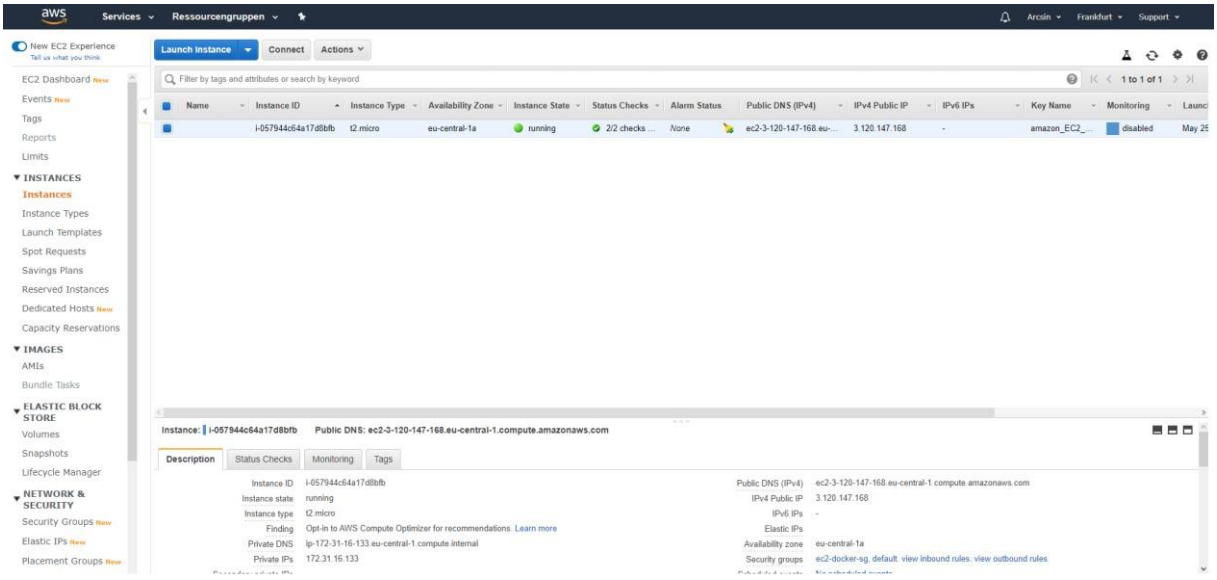
Now I want to push the Image to Docker Hub, so I create a tag for my image, and then push it.

```
C:\Work\IntelliJ\IntelliJProjects\myapp>docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
myapp               latest              769da98b7ad4        51 minutes ago      561MB
openjdk             8                   1077d23b5882        9 days ago          510MB

C:\Work\IntelliJ\IntelliJProjects\myapp>docker tag 769da98b7ad4 barcsin/myapp:myapp

C:\Work\IntelliJ\IntelliJProjects\myapp>docker push barcsin/myapp
The push refers to repository [docker.io/barcsin/myapp]
342f34a76fe4: Pushed
7d0784c2c563: Mounted from library/openjdk
715d0a3d2cc2: Mounted from library/openjdk
155d997ed77c: Mounted from library/openjdk
88cfc2fcd059: Mounted from library/openjdk
760e8d95cf58: Mounted from library/openjdk
7cc1c2d7e744: Mounted from library/openjdk
8c02234b8605: Mounted from library/openjdk
myapp: digest: sha256:42a381259bc7adf575538aca7d337a41161bd6394b0d5e5cd96005f57996775d size: 2006
```

Next Step…

Create Amazon EC2 instance

I created an Amazon EC2 instance. Mostly with the default values. I set it up so that it responds to SSH, HTTP and HTTPs requests.



I created a private key to be able to SSH into the server. Once created, I then connected to the EC2 instance using SSH with this private key.

```
C:\Users\BAHRAMA\Downloads>ssh -i "amazon_EC2_key.pem" ec2-user@ec2-3-120-147-168.eu-central-1.compute.amazonaws.com
The authenticity of host 'ec2-3-120-147-168.eu-central-1.compute.amazonaws.com (3.120.147.168)' can't be established.
ECDSA key fingerprint is SHA256:Y/EgREiIhOxP1r+2jp9gWVCmwhBZoWVD2vUIaQBxygQ.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-3-120-147-168.eu-central-1.compute.amazonaws.com,3.120.147.168' (ECDSA) to the list of k
nown hosts.

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
1 package(s) needed for security, out of 16 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-16-133 ~]$
```

Another option was to use PUTTY but I opted for a simple Shell connection.

Next Step…

Configure EC2 instance

To configure the EC2 instance, I had to first update it. This was done with the following command:

„sudo yum update”

```
ec2-user@ip-172-31-16-133:~
  Verifying  : aws-cfn-bootstrap-1.4-31.amzn2.noarch                                   25/32
  Verifying  : rpm-build-libs-4.11.3-40.amzn2.0.3.x86_64                               26/32
  Verifying  : rpm-plugin-systemd-inhibit-4.11.3-40.amzn2.0.3.x86_64                   27/32
  Verifying  : gdisk-0.8.6-5.amzn2.0.2.x86_64                                          28/32
  Verifying  : rpm-4.11.3-40.amzn2.0.3.x86_64                                          29/32
  Verifying  : libdrm-2.4.83-2.amzn2.0.2.x86_64                                        30/32
  Verifying  : selinux-policy-targeted-3.13.1-192.amzn2.6.noarch                       31/32
  Verifying  : python2-rpm-4.11.3-40.amzn2.0.3.x86_64                                  32/32

Installed:
  kernel.x86_64 0:4.14.177-139.254.amzn2

Dependency Installed:
  libpng.x86_64 2:1.5.13-7.amzn2.0.2

Updated:
  aws-cfn-bootstrap.noarch 0:1.4-32.amzn2.0.1          awscli.noarch 0:1.16.300-1.amzn2.0.2
  freetype.x86_64 0:2.8-14.amzn2                       gdisk.x86_64 0:0.8.10-3.amzn2
  glib2.x86_64 0:2.56.1-5.amzn2.0.1                    libdrm.x86_64 0:2.4.97-2.amzn2
  microcode_ctl.x86_64 2:2.1-47.amzn2.0.6              python2-rpm.x86_64 0:4.11.3-40.amzn2.0.4
  rpm.x86_64 0:4.11.3-40.amzn2.0.4                     rpm-build-libs.x86_64 0:4.11.3-40.amzn2.0.4
  rpm-libs.x86_64 0:4.11.3-40.amzn2.0.4                rpm-plugin-systemd-inhibit.x86_64 0:4.11.3-40.amzn2.0.4
  selinux-policy.noarch 0:3.13.1-192.amzn2.6.1         selinux-policy-targeted.noarch 0:3.13.1-192.amzn2.6.1
  yum.noarch 0:3.4.3-158.amzn2.0.4

Complete!
```

I then had to install docker with the command:

Sudo yum install docker

No need to install java, since its already present in our container.

```
ec2-user@ip-172-31-16-133:~
(2/5): pigz-2.3.4-1.amzn2.0.1.x86_64.rpm                                   |  81 kB  00:00:00
(3/5): containerd-1.3.2-1.amzn2.x86_64.rpm                                 |  24 MB  00:00:00
(4/5): docker-19.03.6ce-3.amzn2.x86_64.rpm                                 |  37 MB  00:00:00
(5/5): runc-1.0.0-0.1.20200204.gitdc9208a.amzn2.x86_64.rpm                 | 2.6 MB  00:00:00
--------------------------------------------------------------------------------------------------
Total                                                           70 MB/s |  65 MB  00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : runc-1.0.0-0.1.20200204.gitdc9208a.amzn2.x86_64                          1/5
  Installing : containerd-1.3.2-1.amzn2.x86_64                                          2/5
  Installing : libcgroup-0.41-21.amzn2.x86_64                                           3/5
  Installing : pigz-2.3.4-1.amzn2.0.1.x86_64                                            4/5
  Installing : docker-19.03.6ce-3.amzn2.x86_64                                          5/5
  Verifying  : containerd-1.3.2-1.amzn2.x86_64                                          1/5
  Verifying  : pigz-2.3.4-1.amzn2.0.1.x86_64                                            2/5
  Verifying  : libcgroup-0.41-21.amzn2.x86_64                                           3/5
  Verifying  : runc-1.0.0-0.1.20200204.gitdc9208a.amzn2.x86_64                          4/5
  Verifying  : docker-19.03.6ce-3.amzn2.x86_64                                          5/5

Installed:
  docker.x86_64 0:19.03.6ce-3.amzn2

Dependency Installed:
  containerd.x86_64 0:1.3.2-1.amzn2          libcgroup.x86_64 0:0.41-21.amzn2  pigz.x86_64 0:2.3.4-1.amzn2.0.1
  runc.x86_64 0:1.0.0-0.1.20200204.gitdc9208a.amzn2

Complete!
```

Next Step…

Deploy the Jar in Docker

Once I finally had a fully configured and running EC2 instance, installed with a Docker engine, I could pull my Docker Image from Docker Hub. If I only used the run command it would've automatically pulled down the image.

```
[ec2-user@ip-172-31-16-133 ~]$ sudo docker pull barcsin/myapp:myapp
myapp: Pulling from barcsin/myapp
376057ac6fa1: Pull complete
5a63a0a859d8: Pull complete
496548a8c952: Pull complete
2adae3950d4d: Pull complete
0a297eafb9ac: Pull complete
51607e1c215b: Pull complete
f7c02da203f1: Pull complete
d5fe4dfea8c0: Pull complete
Digest: sha256:42a381259bc7adf575538aca7d337a41161bd6394b0d5e5cd96005f57996775d
Status: Downloaded newer image for barcsin/myapp:myapp
docker.io/barcsin/myapp:myapp
[ec2-user@ip-172-31-16-133 ~]$
```

Then all I had to do is create a container from the image and run it. But I encountered a lengthy error message, which basically stated this:

```
org.postgresql.util.PSQLException: The connection attempt failed.
```

I had to add both amazon instances, meaning the EC2, running my Application, and the RDS, running my database, to the same network to fix this issue.

After that the web application ran successfully.

```
[ec2-user@ip-172-31-16-133 ~]$ sudo docker run -p 80:8080 barcsin/myapp:myapp

 .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.3.0.RELEASE)

2020-05-25 17:58:58.967  INFO 1 --- [           main] com.baron.myapp.MyappApplication        : Starting MyappApplication v0.0.1-SNAPSHOT on 494acd0f2d11 with PID 1 (/myapp.jar started by root in /)
2020-05-25 17:58:58.976  INFO 1 --- [           main] com.baron.myapp.MyappApplication        : No active profile set, falling back to default profiles: default
2020-05-25 17:59:02.147  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Multiple Spring Data modules found, entering strict repository configuration mode!
2020-05-25 17:59:02.149  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JDBC repositories in DEFAULT mode.
2020-05-25 17:59:02.276  INFO 1 --- [           main] .RepositoryConfigurationExtensionSupport : Spring Data JDBC - Could not safely identify store assignment for repository candidate interface com.baron.myapp.business.entity.repository.
HoneyRepository. If you want this repository to be a JDBC repository, consider annotating your entities with one of these annotations: org.springframework.data.relational.core.mapping.Table.
2020-05-25 17:59:02.277  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 100ms. Found 0 JDBC repository interfaces.
2020-05-25 17:59:02.306  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Multiple Spring Data modules found, entering strict repository configuration mode!
2020-05-25 17:59:02.308  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFERRED mode.
2020-05-25 17:59:02.408  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 89ms. Found 1 JPA repository interfaces.
2020-05-25 17:59:03.545  INFO 1 --- [           main] trationDelegate$BeanPostProcessorChecker : Bean '(inner bean)#6d763516' of type [com.amazonaws.auth.EC2ContainerCredentialsProviderWrapper] is not eligible for getting processed by al
l BeanPostProcessors (for example: not eligible for auto-proxying)
2020-05-25 17:59:03.574  INFO 1 --- [           main] trationDelegate$BeanPostProcessorChecker : Bean '(inner bean)#dd8ba08' of type [com.amazonaws.auth.profile.ProfileCredentialsProvider] is not eligible for getting processed by all Bea
nPostProcessors (for example: not eligible for auto-proxying)
2020-05-25 17:59:03.577  INFO 1 --- [           main] trationDelegate$BeanPostProcessorChecker : Bean 'credentialsProvider' of type [org.springframework.cloud.aws.core.credentials.CredentialsProviderFactoryBean] is not eligible for getti
ng processed by all BeanPostProcessors (for example: not eligible for auto-proxying)
2020-05-25 17:59:03.581  INFO 1 --- [           main] trationDelegate$BeanPostProcessorChecker : Bean 'credentialsProvider' of type [com.amazonaws.auth.AWSCredentialsProviderChain] is not eligible for getting processed by all BeanPostPro
cessors (for example: not eligible for auto-proxying)
2020-05-25 17:59:03.621  WARN 1 --- [           main] com.amazonaws.util.EC2MetadataUtils      : Unable to retrieve the requested metadata (/latest/user-data/). The requested metadata is not found at http://169.254.169.254/latest/user-da
ta/
```

```
2020-05-25 17:59:14.931  INFO 1 --- [           main] com.baron.myapp.MyappApplication        : Started MyappApplication in 17.378 seconds (JVM running for 18.725)


2020-05-25 17:59:49.650  INFO 1 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
2020-05-25 17:59:49.654  INFO 1 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2020-05-25 17:59:49.681  INFO 1 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 22 ms
```

It was finally then time to see if it works. This was it's IP address:

IPv4: 3.120.147.168

The server was responding to HTTP requests.

Nicht sicher | ec2-3-120-147-168.eu-central-1.compute.amazonaws.com/honey

Honey

Choose a food... | Create New Honey | Extra Functionality

Interact	Id	Type	Flower	Health benefits	Vitamins	Tastiness
Edit Delete	1	Manuka Honey	Leptospermum scoparium	antibacterial activity, stimulates the formation of new blood cells, potential wound-healing	B1, B2, B3, B5, and B6	7
Edit Delete	2	Alfalfa Honey	Medicago sativa	prebiotic effects, improved digestion, can treat anemia diabetes and fever	-	9
Edit Delete	3	Eucalyptus Honey	Eucalyptus rostrata	powerful antioxidant and anti-inflammatory agent	sodium, potassium, manganese, magnesium, iron, copper, and zinc	6
Edit Delete	4	Acacia Honey	Acacia flowers	hepatoprotective (liver) and nephroprotective (kidney) effects, wound healing	A, C, and E, flavonoids, and essential fatty and amino acids	10
Edit Delete	5	Buckwheat Honey	Fagopyrum esculentum	can kill notorious pathogens, high bactericidal properties, protection of DNA from chemical or oxidative stress	-	8
Edit Delete	6	Clover Honey	Trifollum species	antioxidant and antimicrobial activity	-	5
Edit Delete	7	Sage Honey	Sage flower	antibacterial, antioxidant, expectorant, and digestive	-	9
Edit Delete	8	Lavender Honey	Lavender flower	antifungal activity	vitamin C, catalase, and flavonoids	9
Edit Delete	9	Rosemary Honey	Rosmarinus officinalis	rich in kaempferol, an antioxidant, natural moisturizing agent	-	7
Edit Delete	10	Jamun Honey	Syzygium cumini	antihypoxic, treatment for Fournier gangrene, promotes wound healing	-	8

Next Step…

Choose a food...
Choose a food...
Honey
Nuts
Berries

Create New Honey | Extra Functionality

Health benefits

vity, stimulates the formation of new blood cells, potential wound-healing

Create a new honey type:

Type: unkown

Flower: a pink one

Health benefits: a lot

Vitamins: Vitamin C

Tastiness: 10

Save

Edit Delete	9	Rosemary Honey	Rosmarinus officinalis	rich in kaempferol, an antioxidant, natural moisturizing agent	-	7
Edit Delete	10	Jamun Honey	Syzygium cumini	antihypoxic, treatment for Fournier gangrene, promotes wound healing	-	8
Edit Delete	11	unkown	a pink one	a lot	Vitamin C	10

Edit honey:

Id: 11

Type: we now know it

Flower: more like red

Health benefits: a lot

Vitamins: Vitamin C

Tastiness: 9

Save

Edit Delete	9	Rosemary Honey	Rosmarinus officinalis	rich in kaempferol, an antioxidant, natural moisturizing agent	-	7
Edit Delete	10	Jamun Honey	Syzygium cumini	antihypoxic, treatment for Fournier gangrene, promotes wound healing	-	8
Edit Delete	11	we now know it	more like red	a lot	Vitamin C	9

Edit Delete	9	Rosemary Honey	Rosmarinus officinalis	rich in kaempferol, an antioxidant, natural moisturizing agent	-	7
Edit Delete	10	Jamun Honey	Syzygium cumini	antihypoxic, treatment for Fournier gangrene, promotes wound healing	-	8

As seen above, the application works as expected. You can perform CRUD operations on the database, that is located on another VM in AWS.