Rodrigo García Carmona

Assignment 3: Deployment of a Web Application in a Cloud Environment

Due date: -
Weight: *20 % of practical assignments' score*

This assignment is *mandatory*. Note that:

- Section 5 explains what items should be delivered to the professor.
- No assignments will be accepted past the due date.

This assignment *is mandatory* and therefore *must be delivered to the professor*. This assignment will contribute to the final score of the course.

1. Introduction

The aim of this assignment is to teach how a typical public cloud environment (in this example, Amazon AWS) is managed and configured. During this assignment you'll understand how several cloud resources can be used and put this knowledge into practice by deploying the web application that you modified in the previous assignment.

1.1. Required Software Installation

Before tackling this assignment you must install the following software items:

- **MySQL Workbench:** Can be installed from the official MySQL website[1]. This is the a management console for the DBMS (DataBase Management System) that the web application is going to use to store data in the production environment.

1.2. Amazon AWS Account

To complete this assignment you will need an Amazon AWS account. You can go to `https://aws.amazon.com` and create a new account if you don't have one already. Notice that this is not the same as a shopping Amazon account; this is a different service.

You might be prompted to input a credit card for registering, but don't worry, since this assignment is only going to use free-tier resources. With that said, if you already have an Amazon AWS account, you might still want to create a new one, since some of the services are only free of charge for the first year. Also note that Amazon pricing schema can change, so be careful with the options that you choose for this assignment.

2. Initial Configuration of the AWS Console

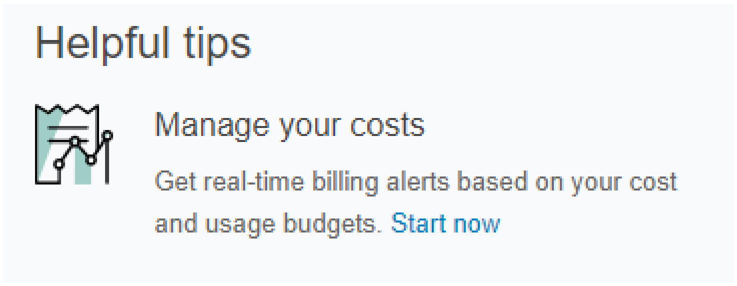
In this section you'll perform the initial configuration of your AWS account.

2.1. Log In and Set Up a Billing Alert

Go to `https://console.aws.amazon.com/` and log into the AWS Management Console. You should see a list of all the available services. If you scroll down, you will also find several tutorials that help you get started.

But first things first. Let's create a billing alert so you don't suddenly find out that Amazon has billed you hundreds of dollars because you misconfigured something. Select *Manage your costs* under the *Helpful tips* section, on the right part of the screen:

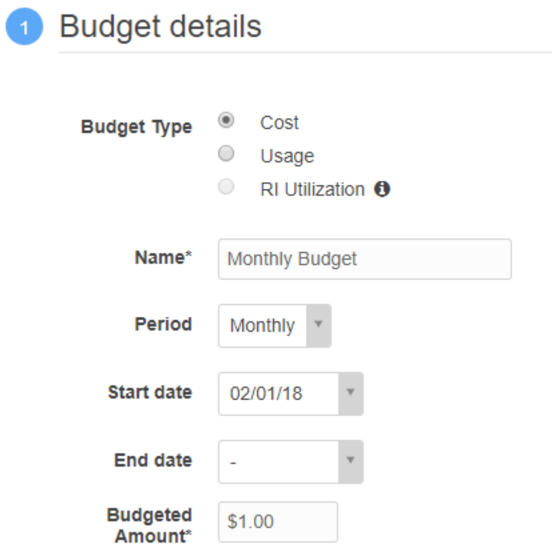
[1] `https://dev.mysql.com/downloads/`


Helpful tips

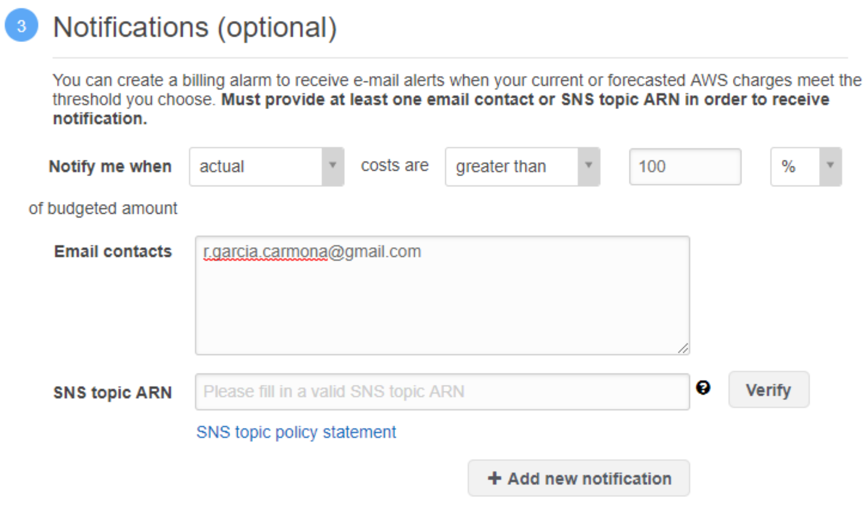
Manage your costs

Get real-time billing alerts based on your cost and usage budgets. Start now

Now, we will create a monthly budget of $1 and an alarm that will email us if that budget is surpassed. You'll need to fill a form similar to the one shown in the following images:


1 Budget details

Budget Type	[✓ Cost] Usage RI Utilization
Name*	Monthly Budget
Period	Monthly
Start date	02/01/18
End date	-
Budgeted Amount*	$1.00


3 Notifications (optional)

You can create a billing alarm to receive e-mail alerts when your current or forecasted AWS charges meet the threshold you choose. **Must provide at least one email contact or SNS topic ARN in order to receive notification.**

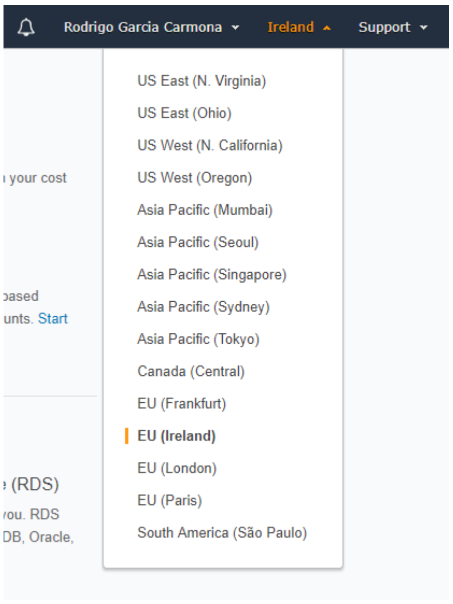
Notify me when actual costs are greater than 100 % of budgeted amount

Email contacts	r.garcia.carmona@gmail.com
SNS topic ARN	Please fill in a valid SNS topic ARN

Then, click in *Create*

2.2. Change the Region

Since you're doing this assignment in Spain, it makes sense to use a cloud region near you. Go back to the AWS Console homepage and change your region to any EU, like London, Ireland or Paris. You can find this configuration option in the right part of the top bar:



3. Cloud Resource Planning

Before doing anything else, you must first think about your web application's structure. Armed with this knowledge, you'll be able to select the best instances and cloud resources to realize it.

3.1. Web Application Components

Your web application is made up of two distinct parts:

- The **web application** itself, built using Play and following the MVC model. This corresponds to the logic layer and part of the presentation layer (the other part of the presentation layer is in the client's web browser).
- A **database** that contains the data used by your application. This corresponds to the data layer. Don't forget that the model in MVC is not part of the data layer, but how such data is seen and managed in the logic layer (review unit 1 slides).

As you can see, it's a very simple application that only has two components. A more complex application could be made up of several interconnected services, data sources and endpoints.

3.2. Cloud Resource Assignment

Think about the two aforementioned components and how they could be deployed into an IaaS (Infrastructure as a Service) cloud. Since these cloud environments provide virtual machines, it makes sense to think that each component will be deployed into a different one. So, you'll deploy two virtual machines:

- One virtual machine for the web application itself.
- One virtual machine for the database.

Since you don't want to spend any money, by looking at the AWS Free Tier[2] you find out that you'll need to use *t2.micro* instances with Linux.

When you launched the web application on your own computer, the same machine was running both the web application and the database. That's OK for testing, but it's not recommended for a production environment: you must separate each component into a different virtual machine and configure them to talk to each other.

[2] `https://aws.amazon.com/free/`

One aspect that's important to consider for cloud resources is scaling: the capability of your web application to adjust to an increased or decreased demand. Luckily for you, by using Play framework and following good programming practices, scalability is easy to implement. You only need to tell AWS which resources, and of which type, will be created or destroyed attending to the demand. This feature is out of the scope of this assignment, but should prove no problem if you choose to implement it (see section 5). It's as simple as specifying how many extra instances of the web application or the database will be needed and defining a replication strategy for the database.

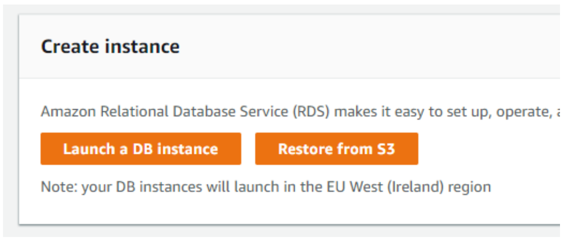
4. Cloud Deployment

In the previous section you planned your deployment with an IaaS in mind, however, AWS offers many PaaS (Platform as a Service) features that can make you like easier. Instead of having to configure a whole Linux system, it'll be better if you can use a preconfigured instance designed for a database or a web application. This way, you'll only need to configure the application itself.

4.1. Database Deployment

AWS provides a service called RDS (Relation Database Service) that is perfect for deploying your database.

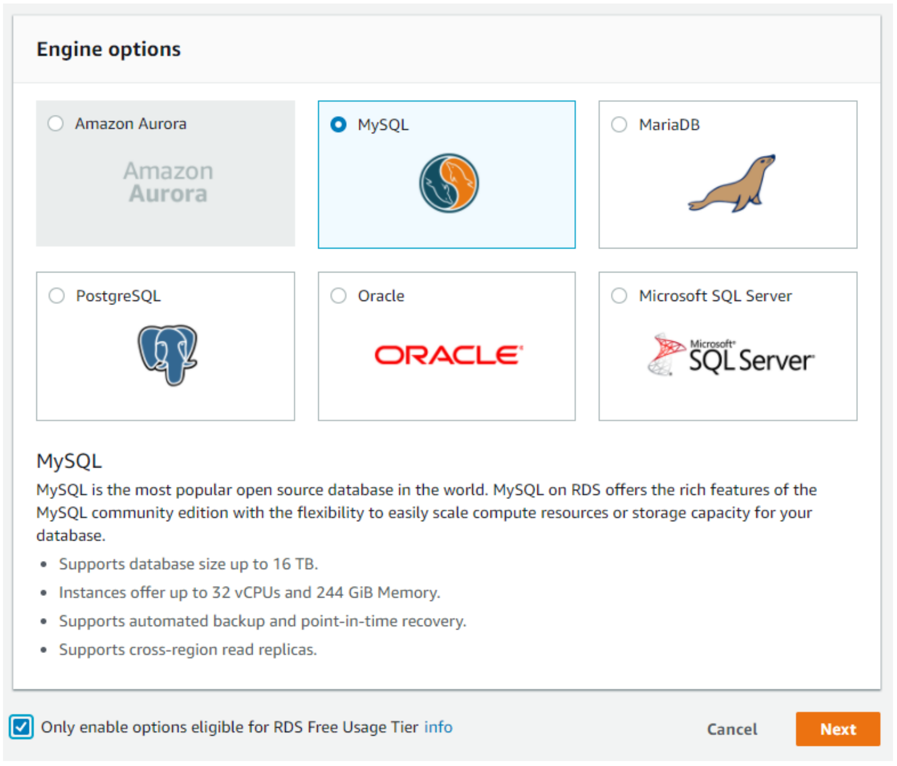
In the main page of the AWS console, select *Relational Database Service* under the *Database* section of the services list. Now, find out the *Create instance* section and click in the *Launch a DB instance* button:


Create instance

Amazon Relational Database Service (RDS) makes it easy to set up, operate, a

Note: your DB instances will launch in the EU West (Ireland) region

In the next page you must select the engine. Click in a box near the bottom that says "Only enable options eligible for the RDS Free Usage Tier". Of the offered engine options, choose MySQL.


Engine options

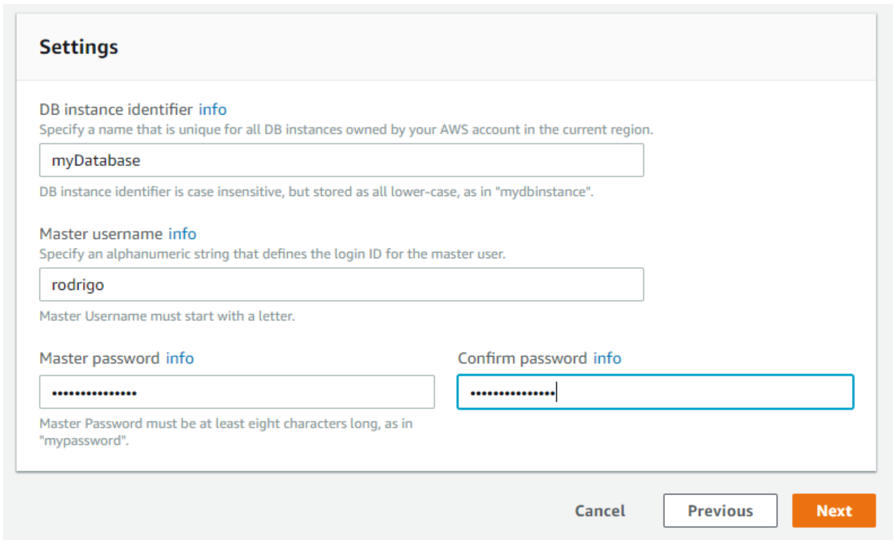
MySQL

MySQL is the most popular open source database in the world. MySQL on RDS offers the rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.

- Supports database size up to 16 TB.
- Instances offer up to 32 vCPUs and 244 GiB Memory.
- Supports automated backup and point-in-time recovery.
- Supports cross-region read replicas.

☑ Only enable options eligible for RDS Free Usage Tier info

Now you need to configure several settings. Since you selected the free tier, most parameters will already be set for you, but you still need to choose the database instance ID, its username and its password. Anything you choose will be OK, but remember the instance ID, user and password, since you'll need them later. Then, click next:


Settings

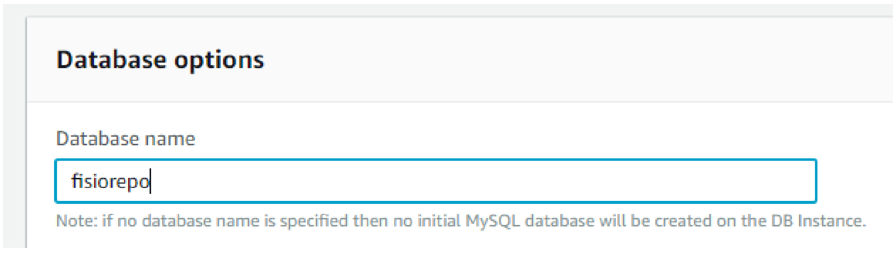
DB instance identifier info
Specify a name that is unique for all DB instances owned by your AWS account in the current region.
myDatabase
DB instance identifier is case insensitive, but stored as all lower-case, as in "mydbinstance".

Master username info
Specify an alphanumeric string that defines the login ID for the master user.
rodrigo
Master Username must start with a letter.

Master password info
•••••••••••••••
Master Password must be at least eight characters long, as in "mypassword".

Confirm password info
•••••••••••••••

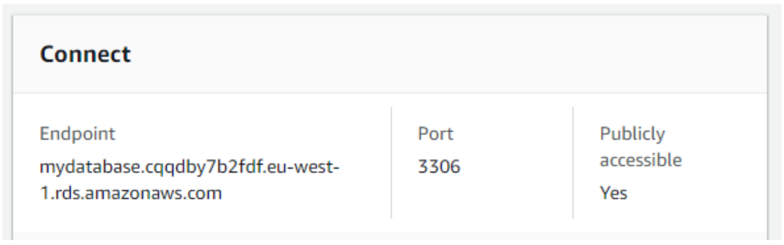
In the next page, you'll fine tune several aspects of the database, look at them and try to think what they represent, but for this assignment you should only chose the database name (in the *Database options section*. This name is important, and is not the same as the instance ID, so remember it for later.


Database options

Database name
fisiorepo
Note: if no database name is specified then no initial MySQL database will be created on the DB Instance.

Now, you can click *Launch DB Instance*. Congratulations, your database is set to be launched! It'll take a while, though. You can now click in *View My Instance* to check the instance details. You can also reach this page from the *Amazon RDS* section in the AWS console. Take your time to explore the database status and options.

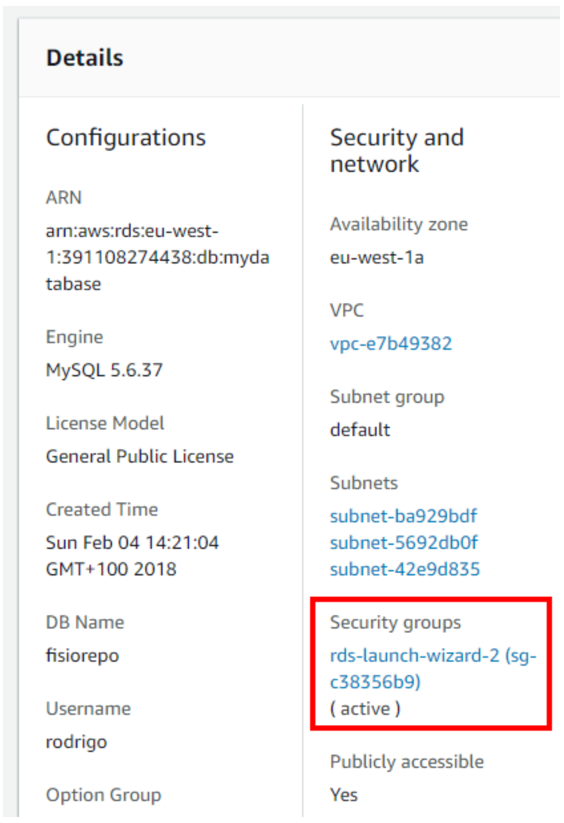
It's specially important that you find out, in this page, the **endpoint** used to connect with your database, since you'll need it later. It should be something like this:


Connect

Endpoint	Port	Publicly accessible
mydatabase.cqqdby7b2fdf.eu-west-1.rds.amazonaws.com	3306	Yes

4.2. Database Access Configuration

By default, RDS databases accept connections from any computer outside AWS (like your own), but can't be accessed by other machines inside the Amazon cloud. Since your web application needs to use your database and it's going to be deployed in Amazon AWS, you need to change this access configuration. In the instance details page for your database (the one you've been looking at in the previous section), find the security group the database belongs to and click on it:



Details

Configurations

ARN
arn:aws:rds:eu-west-1:391108274438:db:mydatabase

Engine
MySQL 5.6.37

License Model
General Public License

Created Time
Sun Feb 04 14:21:04 GMT+100 2018

DB Name
fisiorepo

Username
rodrigo

Option Group

Security and network

Availability zone
eu-west-1a

VPC
vpc-e7b49382

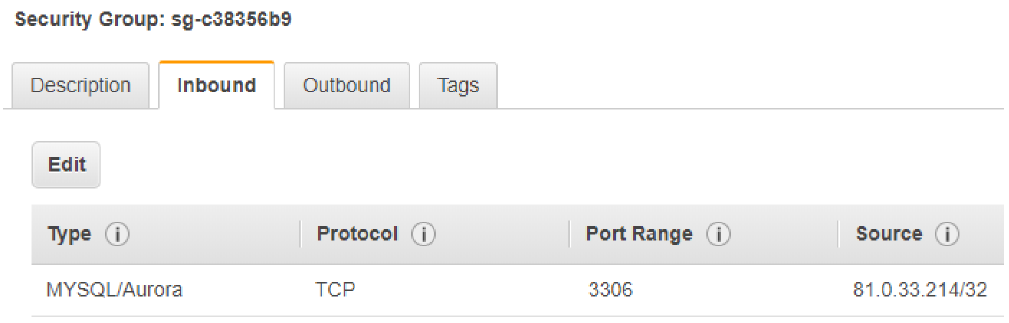
Subnet group
default

Subnets
subnet-ba929bdf
subnet-5692db0f
subnet-42e9d835

Security groups
rds-launch-wizard-2 (sg-c38356b9)
(active)

Publicly accessible
Yes

Then, look at the lower part of the page and click in the *Inbound* tab:

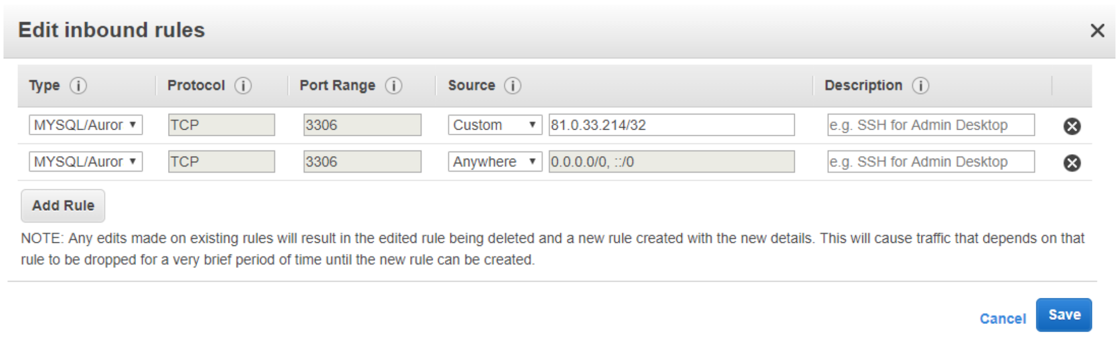


Security Group: sg-c38356b9

Type	Protocol	Port Range	Source
MYSQL/Aurora	TCP	3306	81.0.33.214/32

Here, you will add a new rule to allow connections from your web application. Click in *Edit* and, in the next screen, add a new rule with the following characteristics:

1. MySQL/Aurora type
2. TCP protocol
3. Port 3306
4. Connections from anywhere



Edit inbound rules

Type	Protocol	Port Range	Source		Description
MYSQL/Auror	TCP	3306	Custom	81.0.33.214/32	e.g. SSH for Admin Desktop
MYSQL/Auror	TCP	3306	Anywhere	0.0.0.0/0, ::/0	e.g. SSH for Admin Desktop

NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created.

When you have finished, click *Save*.

4.2.1. *Optional:* Check if the Database is Working

If you want to check that everything's working as intended, you can do one (or both) of the following two things:

- Run in your own computer a version of the web application configured to work with the remote database (see section 4.3.1).
- Use MySQL Workbench to access your remote database. Please, note that some of the university's network forbid these types of connection, so you might only be able of doing this check from your home or other network. MySQL Workbench is very easy to use, so we won't explain how to use this software here. Just remember to use the connection parameters that you just configured.

4.3. Web Application Configuration

AWS provides a PaaS called Elastic Beanstalk. Elastic Beanstalk has some very interesting features, like auto-scaling, but the most relevant to us is that we can deploy a web application into it instead of having to configure a complete virtual machine. Elastic Beanstalk accept many container and applicaton types, and one of them are Java Applications.

Therefore, before deploying you application into Elastic Beanstalk you must configure it so the web application:

1. Uses the AWS database that you've just created.
2. Is exported as a Java Application compatible with Elastic Beanstalk.

4.3.1. Database configuration

Using a text editor (not a word processor) open the file "conf/application.conf" of your web application and search for the following lines:

```
# Amazon RDS parameters
# default.url = "jdbc:mysql://DBURL/fisiorepo?characterEncoding=UTF-8"
# default.driver = com.mysql.jdbc.Driver
# default.username = fisiorepo
# default.password = PASSWORD

## H2 Database parameters
 default.driver = org.h2.Driver
## Choose one of the following two
## In memory database
# default.url = "jdbc:h2:mem:play"
## Local database mimicking MySQL
 default.url = "jdbc:h2:file:~/db/db;MODE=MYSQL"
```

The "#" character at the beginning of a line is a comment symbol. You have to configure the web application so it uses the RDS database instead of the in-memory one, so you need to comment the H2 Database lines and uncomment the appropriate RDS lines.

On top of that, you must also write the configuration parameters of your database:

1. In **default.url**, change the text "DBURL" for your database's endpoint.
2. In **default.url**, change the text "fisiorepo" for your database's name (not the instance ID).
3. In **default.username**, put your database's username.
4. In **default.password**, put your database's password.

It should look like this. I've split the default url in several lines so it can be shown properly in this document, you **should not** do that in your file:

```
# Amazon RDS parameters
default.url =
 "jdbc:mysql://mydatabase.cqqdby7b2fdf.eu-west-1.rds.amazonaws.com/
 fisiorepo?characterEncoding=UTF-8"
default.driver = com.mysql.jdbc.Driver
default.username = rodrigo
default.password = masterbiomedica

## H2 Database parameters
# default.driver = org.h2.Driver
## Choose one of the following two
## In memory database
# default.url = "jdbc:h2:mem:play"
## Local database mimicking MySQL
# default.url = "jdbc:h2:file:~/db/db;MODE=MYSQL"
```

4.3.2. Secret Key for Cookies Configuration

You will continue editing the "conf/application.conf" file. Look for the following lines:

```
# Needed for deployment in Amazon Beanstalk
play.http.secret.key = "SECRETKEY"
play.filters.hosts {
   allowed = ["."]
}
```

And change "SECRETKEY" for the secret key that's going to be used in the cookies produced by your web application. This is an important security measure to avoid cookie forgery. To create a secret key, open a terminal (*PowerShell* in Windows, *Terminal* in Mac OS) and navigate to the project folder. Inside that folder, type the following command:

```
sbt playGenerateSecret
```

This command will produce a secret key that you put in the configuration file.

4.3.3. Export as Java Application

Configuring a web application to be exported as an Elastic Beanstalk has many implications: several configuration files needs to be modified and taken into account. But, luckily for you, this job has already been done in your web application, so you only need to do the final steps.

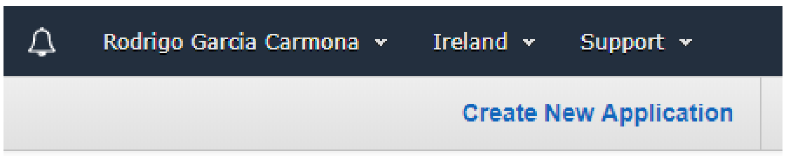
Open a terminal and navigate to the project folder. Inside that folder, type the following commands:

```
sbt clean
sbt compile
sbt dist
```

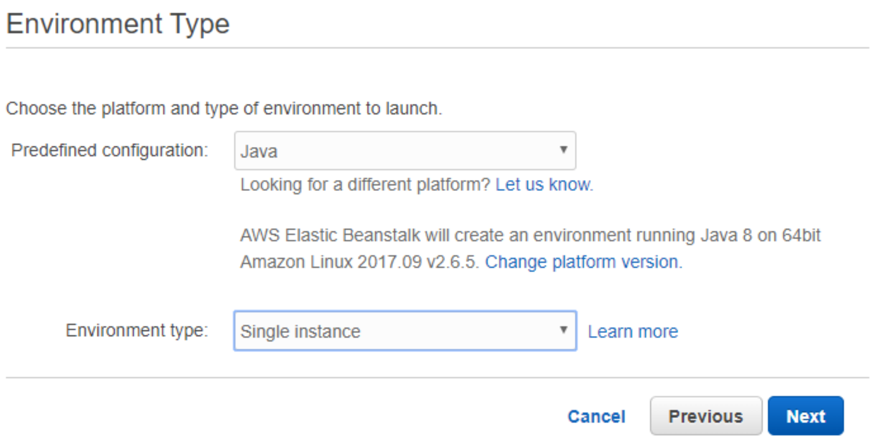
The result of this command is a ZIP file that should be in the "target/universal" folder. You'll upload this ZIP file to AWS.

4.4. Web Application Deployment

In the main page of the AWS console, select *Elastic Beanstalk* under the *Compute* section of the services list. Now, click *Create New Application* in the top right corner of the screen:



Give the application a name and a description (any will do) and click *Next*. Now, choose the *Create Web Server* option. In the next screen select the Java platform and a single instance environment. Then, click *Next*:



Environment Type

Choose the platform and type of environment to launch.

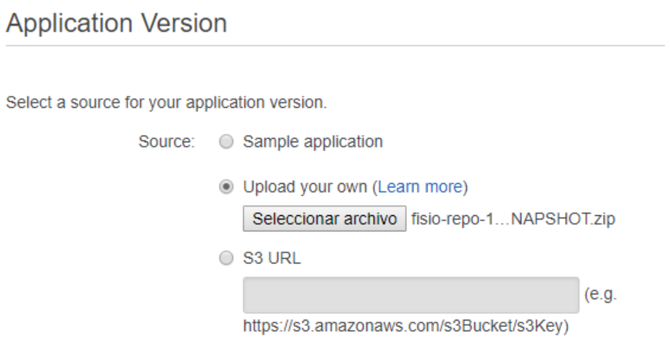
Predefined configuration: Java

Looking for a different platform? Let us know.

AWS Elastic Beanstalk will create an environment running Java 8 on 64bit Amazon Linux 2017.09 v2.6.5. Change platform version.

Environment type: Single instance

In the next screen, in *Application version*, click in *Upload your own* and select the ZIP file that you just created. Don't modify any deployment preferences and click *Next*:



Application Version

Select a source for your application version.

Source: ( ) Sample application

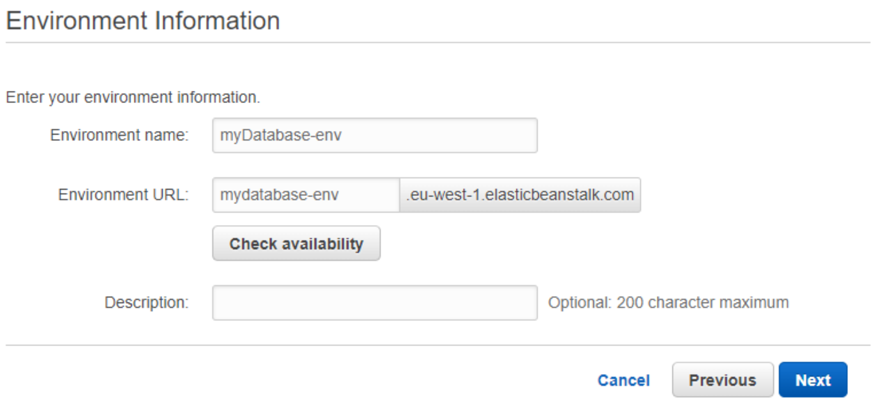
(●) Upload your own (Learn more)

Seleccionar archivo fisio-repo-1...NAPSHOT.zip

( ) S3 URL

(e.g. https://s3.amazonaws.com/s3Bucket/s3Key)

Now choose and environment name and URL. This is specially important, since it's the URL that you're going to use to access your web application. Click *Next*:



Environment Information

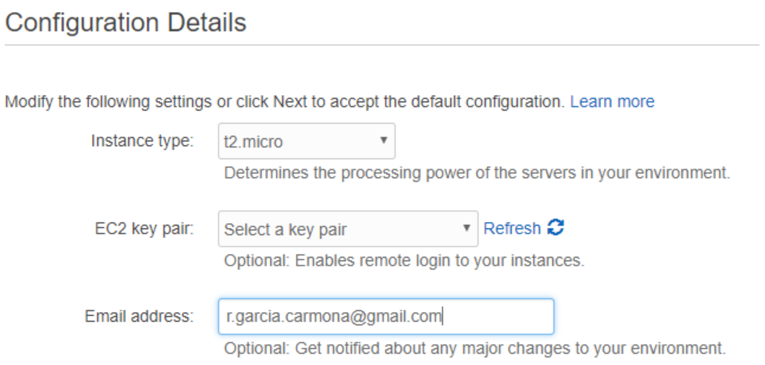
Enter your environment information.

Environment name: myDatabase-env

Environment URL: mydatabase-env .eu-west-1.elasticbeanstalk.com

Description: Optional: 200 character maximum

In the following screen, don't create a new RDS instance, since you already have one, and click *Next*. Now, you will need to select an instance type (select *t2.micro*, since it's the best you can use in the free tier) and put an email address which will be notified of any changes in the status of the database. Click *Next* when you're ready.


Configuration Details

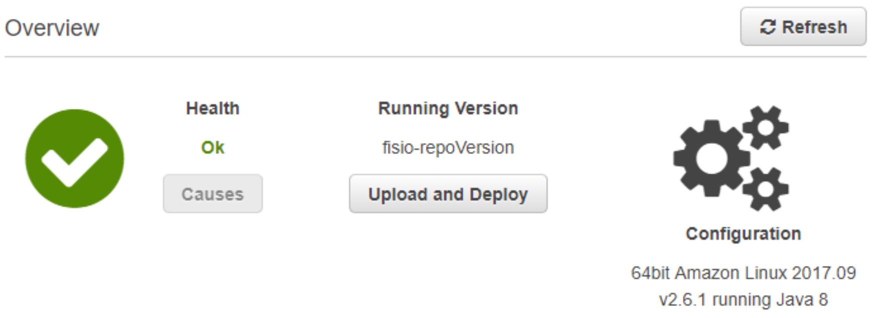
Modify the following settings or click Next to accept the default configuration. Learn more

Instance type: t2.micro
Determines the processing power of the servers in your environment.

EC2 key pair: Select a key pair
Optional: Enables remote login to your instances.

Email address: r.garcia.carmona@gmail.com
Optional: Get notified about any major changes to your environment.

Don't specify any environment tags and click *Next*. Again, don't change any of the permissions and click *Next*. In the next screen you'll be able to review all the relevant information of your instance. When you're sure that everything's as it should, click *Launch* and go make a coffee, since you'll need to wait a couple of minutes until your instance is ready. If everything went right, you'll see a beautiful big green OK symbol:


Overview

Health: Ok

Running Version: fisio-repoVersion

Configuration: 64bit Amazon Linux 2017.09 v2.6.1 running Java 8

Finally, you can access your web application deployed in the cloud! From any computer. Just go to an URL that would look like the following:

```
http://YOUR_ENVIRONMENT_NAME.YOUR_REGION.elasticbeanstalk.com/
```

For instance, mine is:

```
http://fisio-repo.eu-west-1.elasticbeanstalk.com/
```

Note that you don't need to access port 9000, since the application has been configured to use port 80 (the HTTP standard port) when deployed in production. Access the endpoints that you created for the new element of the model in the previous assignment, test some thing here and there, and you're done! Congratulations!

5. Delivery

To pass this assignment you must send the URL of the cloud deployment of your web application to the professor's email (r.garcia.carmona@gmail.com). That's enough. Of course, keep in mind that the application that you must have deployed is the one that you created for the previous assignment, with the extra endpoints and model.

6. If you are bored...

If you're bored and have already finished the assignment, you can try any of the following advanced tasks:

1. Deploy more copies of your Elastic Beanstalk application.

2. Update the already deployed version of your Elastic Beanstalk application with a new version with some changes.

3. Try to manage a load balancing and auto-scaling deployment of your Elastic Beanstalk application.
4. Create a proper security group, in which only the chosen Elastic Beanstalk instances can access the RDS database.
5. Play a little bit with the RDS database and rollback to a previous state.
6. Change the backup schema of your RDS database.