Deploy Spring Boot App on AWS – Elastic Beanstalk

bezkoder.com/deploy-spring-boot-aws-eb

bezkoder Last modified: November 16, 2021

AWS (Amazon Web Services) is one of the most widely used cloud computing platforms which provides a whole range of managed cloud services. In this tutorial, I will show you step by step to deploy Spring Boot Application with MySQL on AWS EC2 Instance using Elastic Beanstalk (for free).

Related Posts:
– Spring Boot, Spring Data JPA: Rest API example
– Spring Boot JdbcTemplate with MySQL: Rest API example
– Spring Boot Token based Authentication with Spring Security & JWT
– Dockerize with Docker Compose: Spring Boot and MySQL example

Overview

AWS provides a free tier to experience its services for free in 1 year. And you can follow the step by step in this tutorial without caring about pricing.

You need to go to Amazon AWS and sign up for AWS free tier first.
And we're gonna do these steps:

- Install Elastic Beanstalk CLI (EB CLI)
- Initialize Beanstalk for Spring Boot application
- Configure environment for MySQL in AWS RDS database
- Deploy and test the application

What is AWS Elastic Beanstalk?

Elastic Beanstalk provides way to quickly deploy and manage Go, Java, .NET, Node.js, PHP, Python, or Ruby applications in the AWS Cloud.

We don't need to to learn about the infrastructure which runs the applications. Just upload the application, Elastic Beanstalk automatically handles the details of capacity provisioning, load balancing, scaling, and application health monitoring.

When we deploy your application, Elastic Beanstalk builds the selected platform version and provisions one or more AWS resources, such as Amazon EC2 instances, to run your application.

To interact with Elastic Beanstalk, we can use the Elastic Beanstalk console, the AWS Command Line Interface (AWS CLI), or EB CLI.

In this tutorial, we choose Elastic Beanstalk Command Line Interface.

Step 0: Get Spring Boot Application to deploy on AWS

Let’s use a Spring Boot CRUD Restful Apis that works with MySQL database. You can download the source code at Github.

```
$ git clone https://github.com/bezkoder/spring-boot-data-jpa-mysql.git
```

This Spring Boot App contains a Rest service as well as MySQL database. That helps you understand how to deploy a Restful Web Service on AWS instead of simple Hello World.

The application exports APIs as following:

Methods	Urls	Actions
POST	/api/tutorials	create new Tutorial
GET	/api/tutorials	retrieve all Tutorials
GET	/api/tutorials/:id	retrieve a Tutorial by `:id`
PUT	/api/tutorials/:id	update a Tutorial by `:id`
DELETE	/api/tutorials/:id	delete a Tutorial by `:id`
DELETE	/api/tutorials	delete all Tutorials
GET	/api/tutorials/published	find all published Tutorials
GET	/api/tutorials?title=[keyword]	find all Tutorials which title contains `keyword`

The tutorial showing how to build this app can be found at:
Spring Boot, Spring Data JPA – Building Rest CRUD API example

Step 1: Install EB CLI (Elastic Beanstalk Command Line Interface)

The AWS Elastic Beanstalk Command Line Interface is a command line client provides ways to create, configure, manage, monitor Elastic Beanstalk environments.

The EB CLI requires Python version 2.7, 3.4, or later that you can download at https://www.python.org/downloads/.

Run the following command:

```
$ pip install awsebcli --upgrade
```

`--upgrade` : to upgrade any requirements that are already installed.

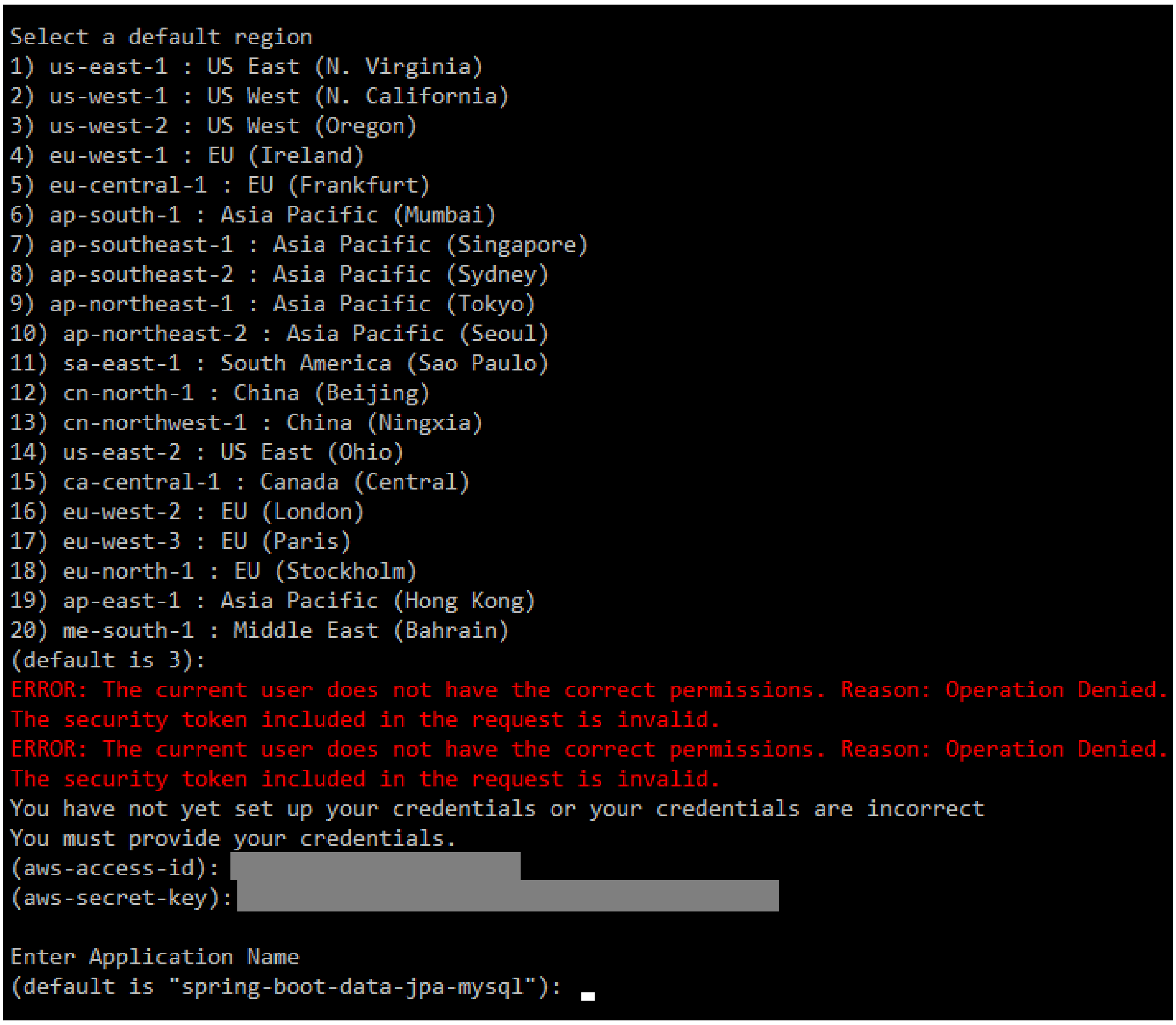
Step 2: Initialize Beanstalk project for Spring Boot application

To initialize an EB CLI project, open command prompt at Spring Boot App root folder, then run the command `eb init`.

```
Select a default region
1) us-east-1 : US East (N. Virginia)
2) us-west-1 : US West (N. California)
3) us-west-2 : US West (Oregon)
4) eu-west-1 : EU (Ireland)
5) eu-central-1 : EU (Frankfurt)
6) ap-south-1 : Asia Pacific (Mumbai)
7) ap-southeast-1 : Asia Pacific (Singapore)
8) ap-southeast-2 : Asia Pacific (Sydney)
9) ap-northeast-1 : Asia Pacific (Tokyo)
10) ap-northeast-2 : Asia Pacific (Seoul)
11) sa-east-1 : South America (Sao Paulo)
12) cn-north-1 : China (Beijing)
13) cn-northwest-1 : China (Ningxia)
14) us-east-2 : US East (Ohio)
15) ca-central-1 : Canada (Central)
16) eu-west-2 : EU (London)
17) eu-west-3 : EU (Paris)
18) eu-north-1 : EU (Stockholm)
19) ap-east-1 : Asia Pacific (Hong Kong)
20) me-south-1 : Middle East (Bahrain)
(default is 3):
```

You can choose any region above, or just enter for default: `us-west-2`.

Then use your **AWS Access Id** and **AWS Secret Key**. If you don't know where they are, just go to Identity and Access Management (IAM).



```
Select a default region
1) us-east-1 : US East (N. Virginia)
2) us-west-1 : US West (N. California)
3) us-west-2 : US West (Oregon)
4) eu-west-1 : EU (Ireland)
5) eu-central-1 : EU (Frankfurt)
6) ap-south-1 : Asia Pacific (Mumbai)
7) ap-southeast-1 : Asia Pacific (Singapore)
8) ap-southeast-2 : Asia Pacific (Sydney)
9) ap-northeast-1 : Asia Pacific (Tokyo)
10) ap-northeast-2 : Asia Pacific (Seoul)
11) sa-east-1 : South America (Sao Paulo)
12) cn-north-1 : China (Beijing)
13) cn-northwest-1 : China (Ningxia)
14) us-east-2 : US East (Ohio)
15) ca-central-1 : Canada (Central)
16) eu-west-2 : EU (London)
17) eu-west-3 : EU (Paris)
18) eu-north-1 : EU (Stockholm)
19) ap-east-1 : Asia Pacific (Hong Kong)
20) me-south-1 : Middle East (Bahrain)
(default is 3):
ERROR: The current user does not have the correct permissions. Reason: Operation Denied.
The security token included in the request is invalid.
ERROR: The current user does not have the correct permissions. Reason: Operation Denied.
The security token included in the request is invalid.
You have not yet set up your credentials or your credentials are incorrect
You must provide your credentials.
(aws-access-id): [redacted]
(aws-secret-key): [redacted]

Enter Application Name
(default is "spring-boot-data-jpa-mysql"):
```

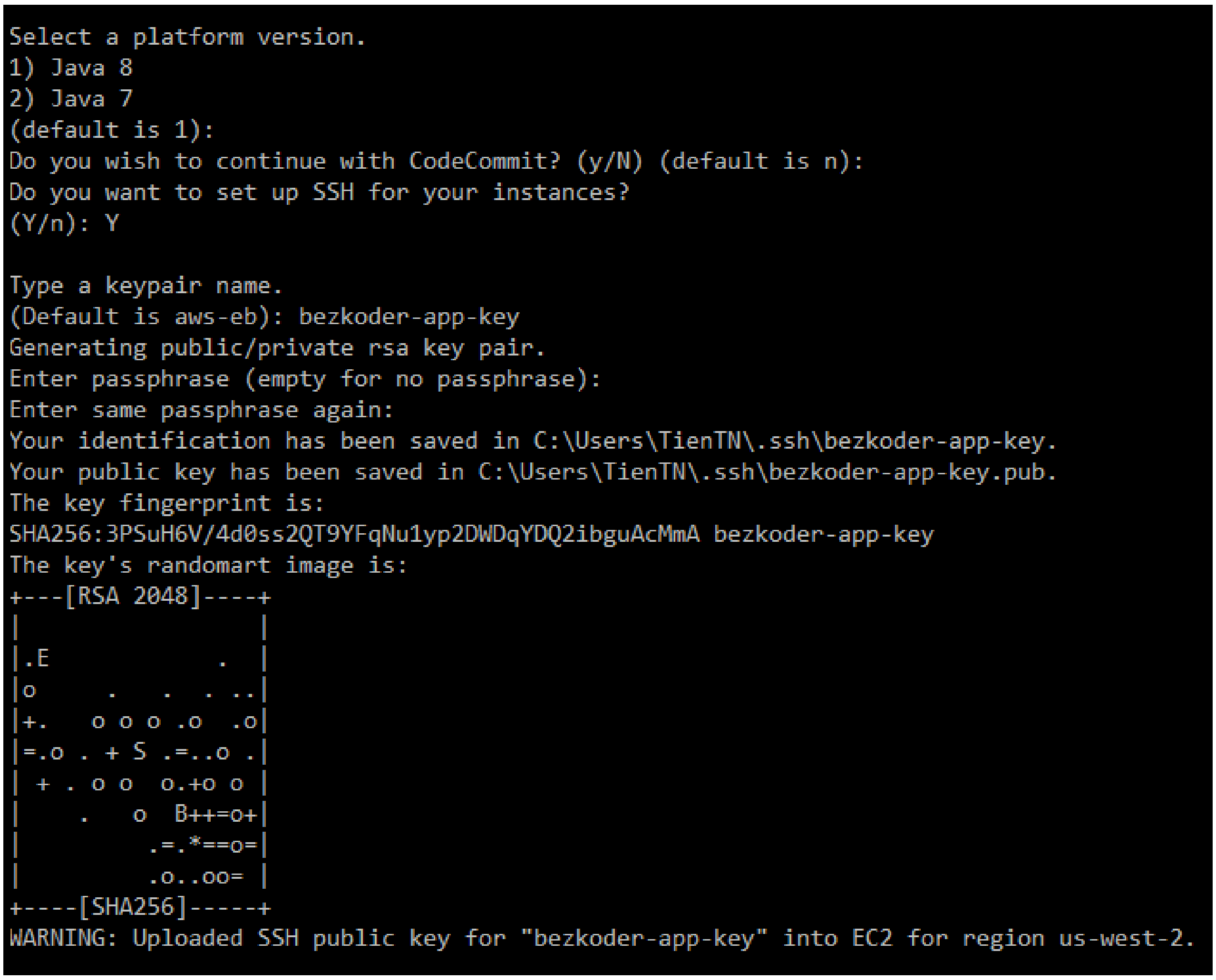
Just type your name, press **Enter** button to select the application.
Now we need to choose platform:

```
Select a platform.
1) Python (BETA)
2) Docker (BETA)
3) Ruby (BETA)
4) Go (BETA)
5) Node.js (BETA)
6) PHP (BETA)
7) Node.js
8) PHP
9) Python
10) Ruby
11) Tomcat
12) IIS
13) Docker
14) Multi-container Docker
15) GlassFish
16) Go
17) Java
18) Corretto (BETA)
19) Packer
(default is 1): 17
```

Type `17` to select Java, then `1` for **Java 8**.

Then **no** for CodeCommit, and **Yes** for SSH to proceed with SSH setup if you need to login to EC2 instances.

We also create an SSH KeyPair: `bezkoder-app-key`.



```
Select a platform version.
1) Java 8
2) Java 7
(default is 1):
Do you wish to continue with CodeCommit? (y/N) (default is n):
Do you want to set up SSH for your instances?
(Y/n): Y

Type a keypair name.
(Default is aws-eb): bezkoder-app-key
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\TienTN\.ssh\bezkoder-app-key.
Your public key has been saved in C:\Users\TienTN\.ssh\bezkoder-app-key.pub.
The key fingerprint is:
SHA256:3PSuH6V/4d0ss2QT9YFqNu1yp2DWDqYDQ2ibguAcMmA bezkoder-app-key
The key's randomart image is:
+---[RSA 2048]----+
|                 |
|.E            .  |
|o     .   .  . ..|
|+.   o o o .o  .o|
|=.o . + S .=..o .|
| + . o o  o.+o o |
|    .   o  B++=o+|
|         .=.*==o=|
|         .o..oo= |
+----[SHA256]-----+
WARNING: Uploaded SSH public key for "bezkoder-app-key" into EC2 for region us-west-2.
```

OK, done! just go to the next step.

Step 3: Configure Deployment for Spring Boot Application

Now Elastic beanstalk will automatically create a configuration file in Spring Boot project directory: **.elasticbeanstalk**/*config.yml*.

```
branch-defaults:
  master:
    environment: null
    group_suffix: null
global:
  application_name: spring-boot-data-jpa-mysql
  branch: null
  default_ec2_keyname: bezkoder-app-key
  default_platform: Java 8
  default_region: us-west-2
  include_git_submodules: true
  instance_profile: null
  platform_name: null
  platform_version: null
  profile: eb-cli
  repository: null
  sc: git
  workspace_type: Application
```

Our application will be built as **jar** package. So we add configuration for the deployment as below:

```
deploy:
  artifact: target/spring-boot-data-jpa-0.0.1-SNAPSHOT.jar
```

The `spring-boot-data-jpa` is the artifact of our application and `0.0.1-SNAPSHOT` is the version.
spring-boot-data-jpa-0.0.1-SNAPSHOT.jar file will be generated in the **target** folder when we package the Spring Boot application with the command:

```
$ mvn clean package spring-boot:repackage
```

Step 4: Configure environment for Spring Boot with MySQL on AWS

Let’s create an Elastic Beanstalk environment with the command:

```
$ eb create --single --database
```

Because an elastic beanstalk application could have multiple environments, we use `--single` for a single EC2 instance and an RDS database. If not, multiple instances will be created with a load balancer which cost our money.

```
$ eb create --single --database
Enter Environment Name
(default is spring-boot-data-jpa-mysql-dev): spring-boot-dev
Enter DNS CNAME prefix
(default is spring-boot-dev):
Would you like to enable Spot Fleet requests for this environment?
(y/N): N
```

Now type username and password for the RDS database:

```
Enter an RDS DB username (default is "ebroot"): boot
Enter an RDS DB master password:
Retype password to confirm:
```

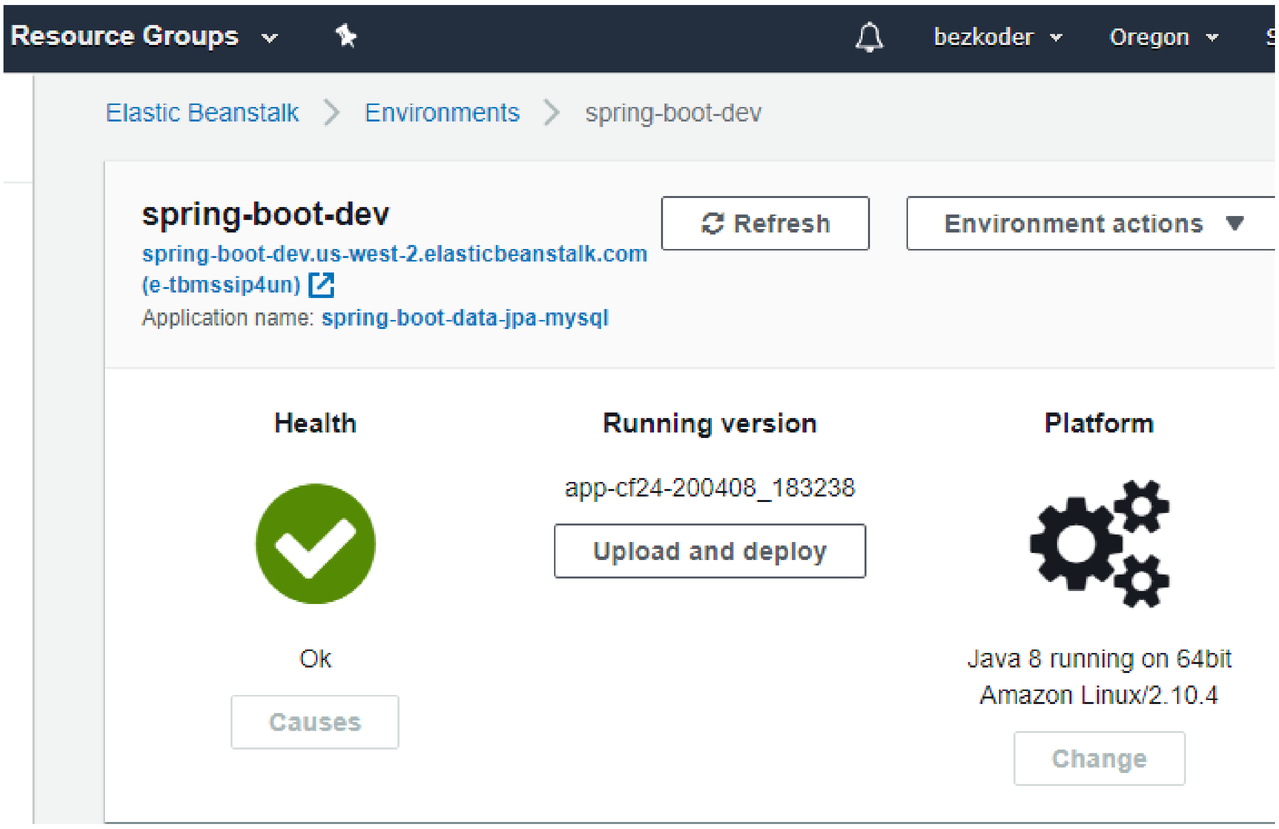
The console shows:

```
Uploading: [##################################################] 100% Done...
Environment details for: spring-boot-dev
  Application name: spring-boot-data-jpa-mysql
  Region: us-west-2
  Deployed Version: app-cf24-200408_183238
  Environment ID: e-tbmssip4un
  Platform: arn:aws:elasticbeanstalk:us-west-2::platform/Java 8 running on 64bit
Amazon Linux/2.10.4
  Tier: WebServer-Standard-1.0
  CNAME: spring-boot-dev.us-west-2.elasticbeanstalk.com
  Updated: 2020-04-08 11:33:04.032000+00:00
Printing Status:
2020-04-08 11:33:03    INFO    createEnvironment is starting.
2020-04-08 11:33:04    INFO    Using elasticbeanstalk-us-west-2-668363982883 as
Amazon S3 storage bucket for environment data.
2020-04-08 11:33:25    INFO    Created security group named: awseb-e-tbmssip4un-
stack-AWSEBSecurityGroup-T3UYE8P9CGUY
2020-04-08 11:33:40    INFO    Creating RDS database named: aazuzc8hjztxex. This
may take a few minutes.
2020-04-08 11:33:40    INFO    Created EIP: 54.185.147.178
2020-04-08 11:40:05    INFO    Created RDS database named: aazuzc8hjztxex
```

Let's open AWS Console with the command:

```
$eb console
```

You can see our Elastic Beanstalk environment:



spring-boot-dev
spring-boot-dev.us-west-2.elasticbeanstalk.com
(e-tbmssip4un)
Application name: spring-boot-data-jpa-mysql

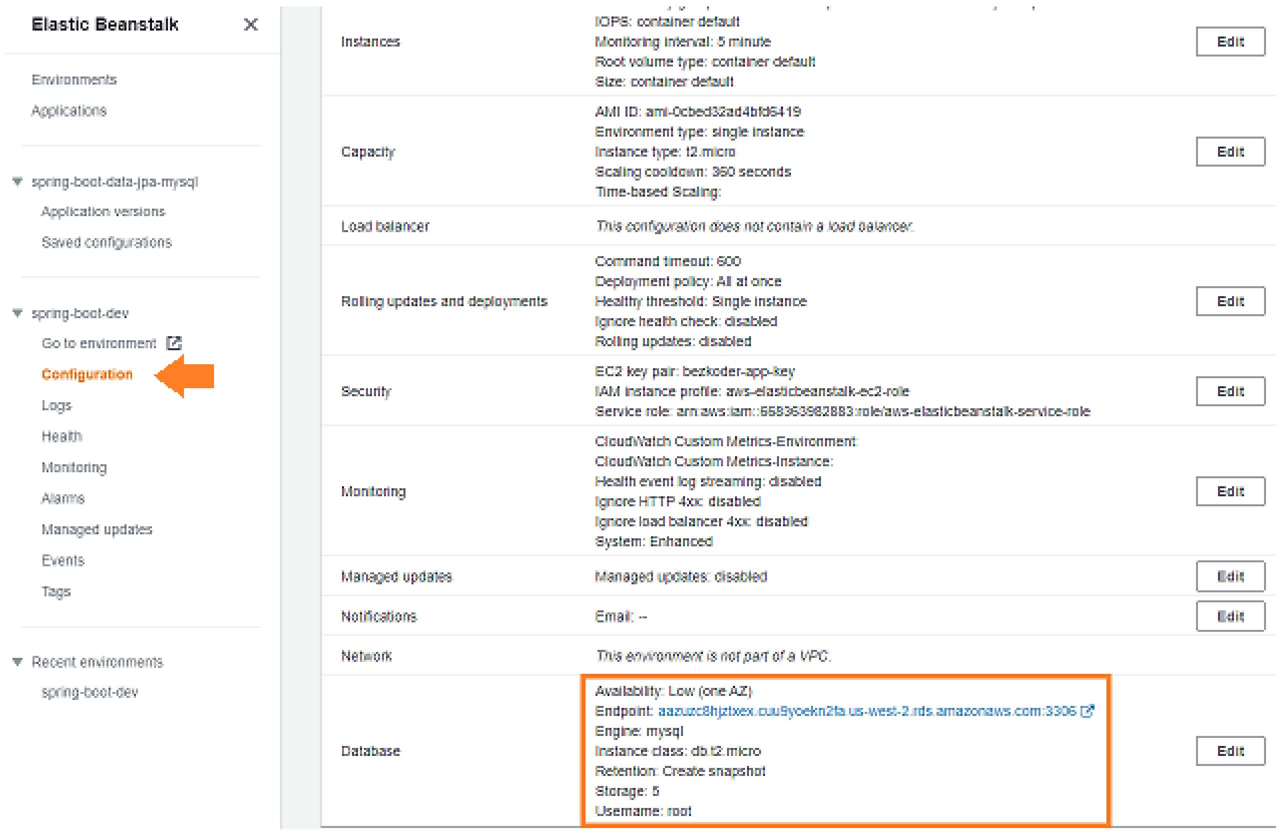
Health
Ok

Running version
app-cf24-200408_183238

Platform
Java 8 running on 64bit Amazon Linux/2.10.4

It can change to **Warning** or **Severe**. Don't worry, we need to set some configuration.

Click Configuration tab on the left side, the database shows the information as following:



Instances	IOPS: container default Monitoring interval: 5 minute Root volume type: container default Size: container default	Edit
Capacity	AMI ID: ami-0cbed32ad4bfd6419 Environment type: single instance Instance type: t2.micro Scaling cooldown: 360 seconds Time-based Scaling:	Edit
Load balancer	*This configuration does not contain a load balancer.*	
Rolling updates and deployments	Command timeout: 600 Deployment policy: All at once Healthy threshold: Single instance Ignore health check: disabled Rolling updates: disabled	Edit
Security	EC2 key pair: bezkoder-app-key IAM instance profile: aws-elasticbeanstalk-ec2-role Service role: arn:aws:iam::658363982883:role/aws-elasticbeanstalk-service-role	Edit
Monitoring	CloudWatch Custom Metrics-Environment: CloudWatch Custom Metrics-Instance: Health event log streaming: disabled Ignore HTTP 4xx: disabled Ignore load balancer 4xx: disabled System: Enhanced	Edit
Managed updates	Managed updates: disabled	Edit
Notifications	Email: --	Edit
Network	*This environment is not part of a VPC.*	
Database	Availability: Low (one AZ) Endpoint: aazuzc8hjztxex.cuu9yoekn2fa.us-west-2.rds.amazonaws.com:3306 Engine: mysql Instance class: db.t2.micro Retention: Create snapshot Storage: 5 Username: root	Edit

By default, Spring Boot applications will listen on port `8080`. Elastic Beanstalk assumes that the application will listen on port `5000`. So we will change the port the Spring Boot application listens on.

Let's use `eb setenv` for the work:

```
$ eb setenv SERVER_PORT=5000
```

Let's use `eb setenv` with *endpoint* and port in the image above, *username* and *password* we've set before:

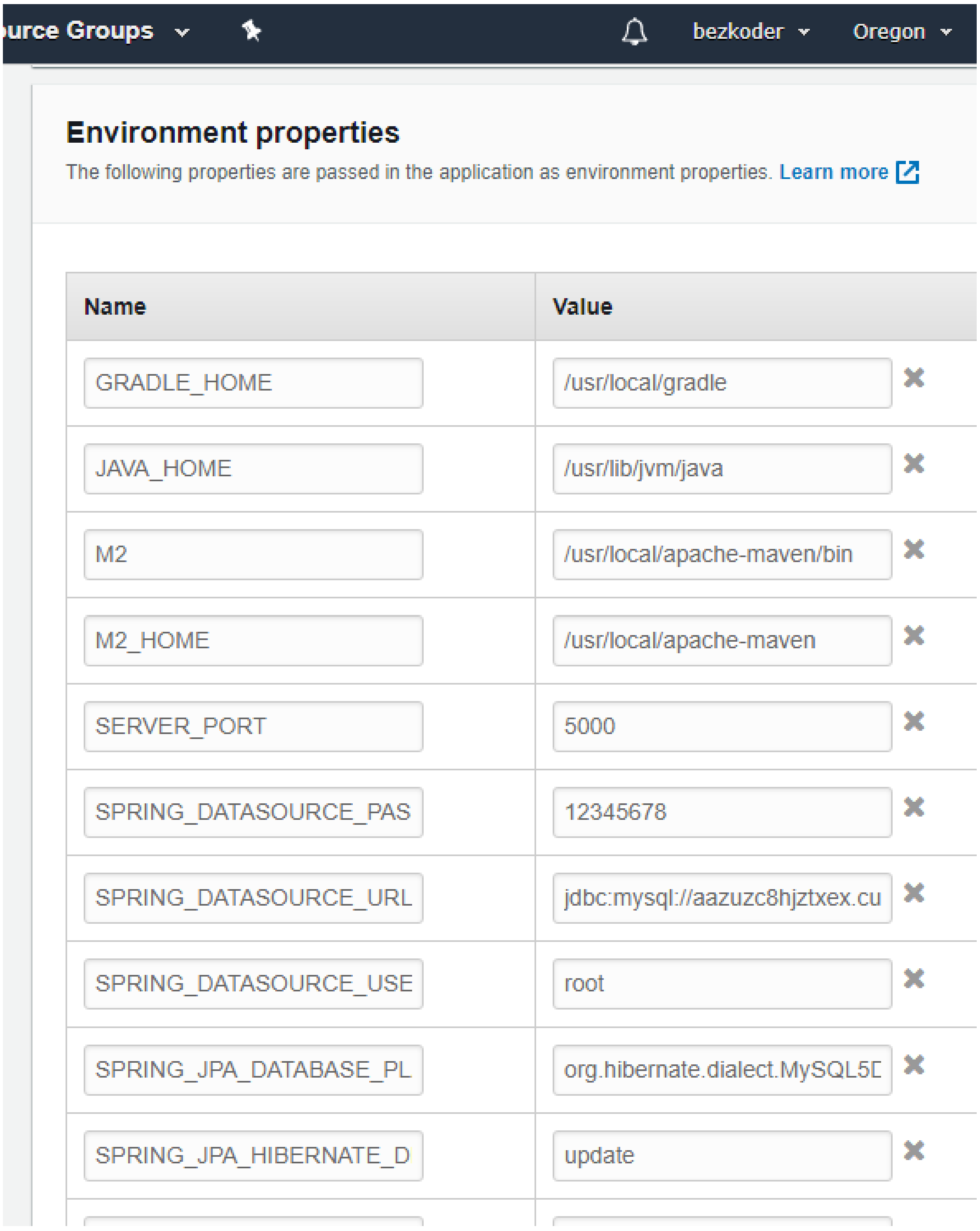
```
SPRING_DATASOURCE_URL=jdbc:mysql:///ebdb
SPRING_DATASOURCE_USERNAME=
SPRING_DATASOURCE_PASSWORD=
SPRING_JPA_HIBERNATE_DDL_AUTO=update
SPRING_JPA_DATABASE_PLATFORM=org.hibernate.dialect.MySQL5Dialect

$ eb setenv SPRING_DATASOURCE_URL=jdbc:mysql://aazuzc8hjztxex.cuu9yoekn2fa.us-
west-2.rds.amazonaws.com:3306/ebdb SPRING_DATASOURCE_USERNAME=root
SPRING_DATASOURCE_PASSWORD=12345678 SPRING_JPA_HIBERNATE_DDL_AUTO=update
SPRING_JPA_DATABASE_PLATFORM=org.hibernate.dialect.MySQL5Dialect
```

The console shows:

```
2020-04-08 12:03:03    INFO    Environment update is starting.
2020-04-08 12:03:11    INFO    Updating environment spring-boot-dev's
configuration settings.
2020-04-08 12:04:33    INFO    Successfully deployed new configuration to
environment.
```

Now open AWS Console, in **Configuration** tab, click on **Edit** button in **Software** Category, we can see all of **Environment properties** here:



Environment properties

The following properties are passed in the application as environment properties. Learn more

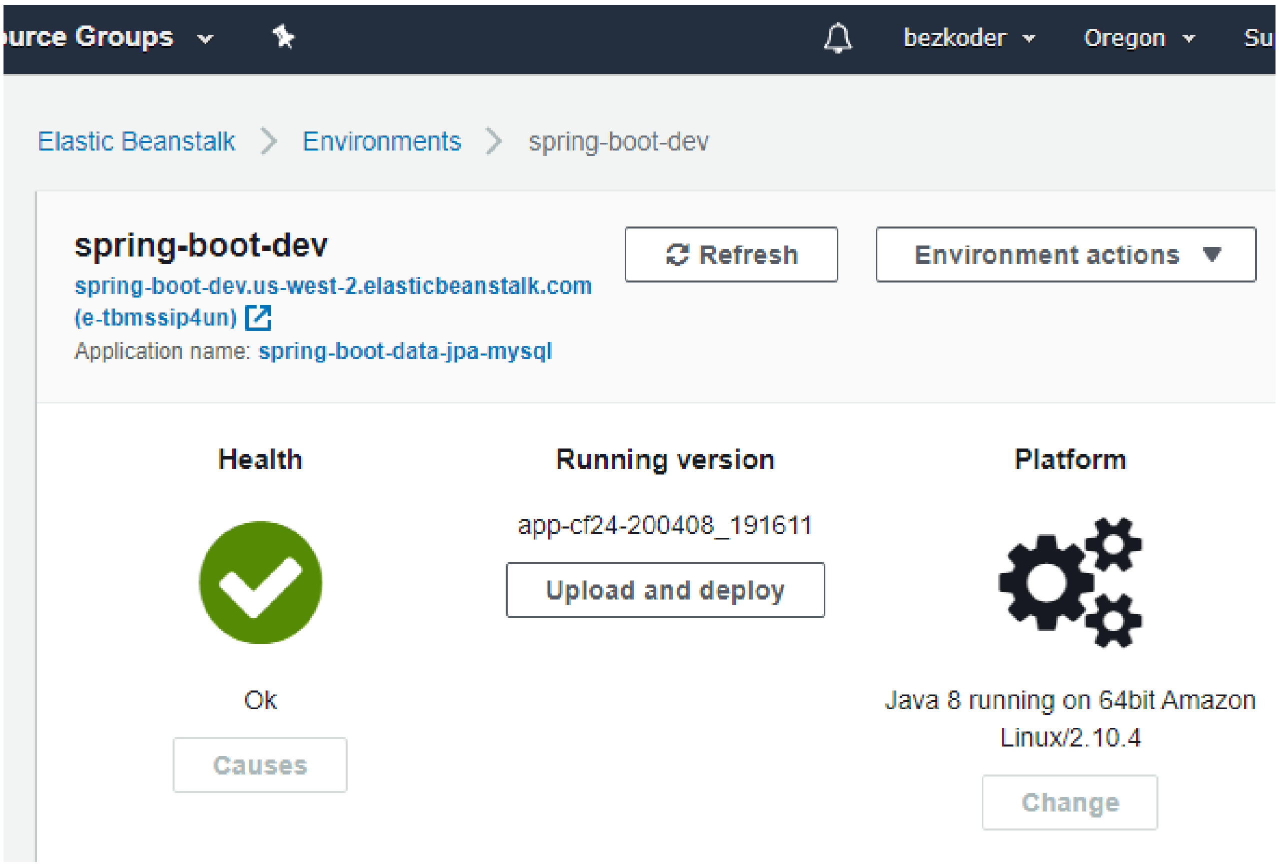
Name	Value
GRADLE_HOME	/usr/local/gradle
JAVA_HOME	/usr/lib/jvm/java
M2	/usr/local/apache-maven/bin
M2_HOME	/usr/local/apache-maven
SERVER_PORT	5000
SPRING_DATASOURCE_PAS	12345678
SPRING_DATASOURCE_URL	jdbc:mysql://aazuzc8hjztxex.cu
SPRING_DATASOURCE_USE	root
SPRING_JPA_DATABASE_PL	org.hibernate.dialect.MySQL5D
SPRING_JPA_HIBERNATE_D	update

Step 5: Deploy the Spring Boot application to AWS

Now we just run the command for deploying the Spring Boot application to the instances in the Elastic Beanstalk environment.

```
$ eb deploy
Uploading: [##################################################] 100% Done...
2020-04-08 12:16:42    INFO    Environment update is starting.
2020-04-08 12:16:46    INFO    Deploying new version to instance(s).
2020-04-08 12:17:05    INFO    New application version was deployed to running EC2
instances.
2020-04-08 12:17:05    INFO    Environment update completed successfully.
```

Check the AWS Console:



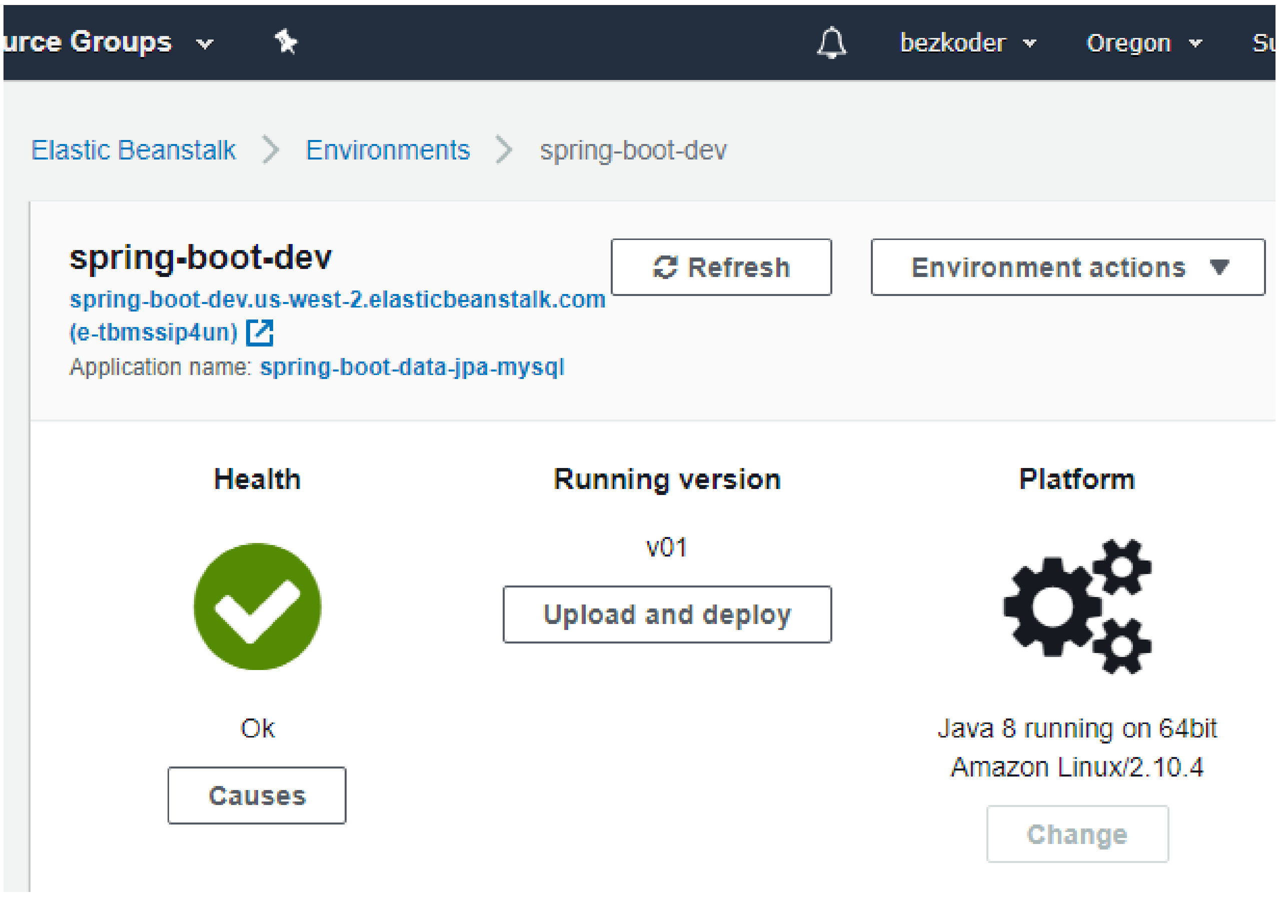
spring-boot-dev
spring-boot-dev.us-west-2.elasticbeanstalk.com (e-tbmssip4un)
Application name: spring-boot-data-jpa-mysql

Health: Ok
Running version: app-cf24-200408_191611
Platform: Java 8 running on 64bit Amazon Linux/2.10.4

If you want to deploy with a label version, just use: `eb deploy --label [your-version]`.
If the label has already been used, the EB CLI redeploys the previous version with that label.

For example:

```
eb deploy --label v01
Uploading: [##################################################] 100% Done...
2020-04-08 12:22:42    INFO    Environment update is starting.
2020-04-08 12:22:46    INFO    Deploying new version to instance(s).
2020-04-08 12:23:04    INFO    New application version was deployed to running EC2
instances.
2020-04-08 12:23:04    INFO    Environment update completed successfully.
```



spring-boot-dev

spring-boot-dev.us-west-2.elasticbeanstalk.com (e-tbmssip4un)

Application name: spring-boot-data-jpa-mysql

Health: Ok

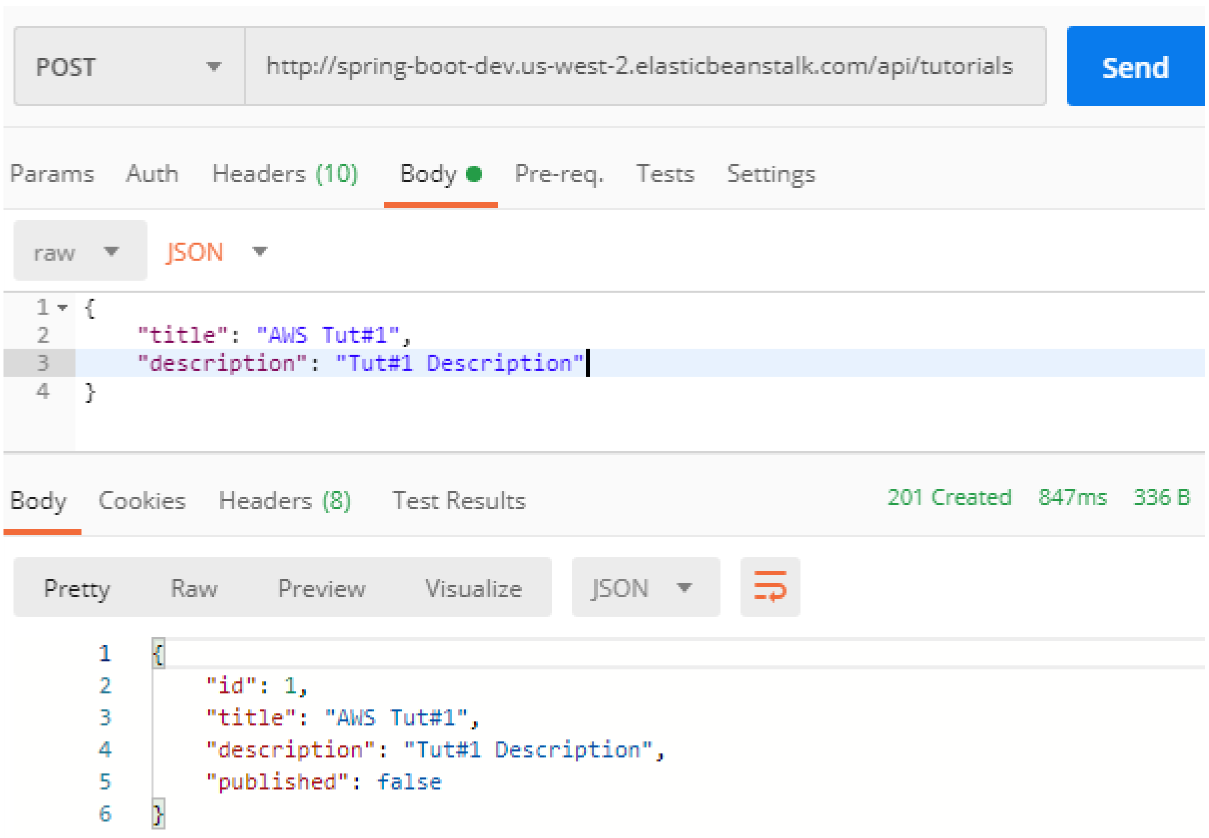
Running version: v01

Platform: Java 8 running on 64bit Amazon Linux/2.10.4

Step 6: Test the application

This is the final section of the tutorial, we're gonna use Postman to test the Spring Boot Rest Apis which was deployed on AWS.

– Create an object:



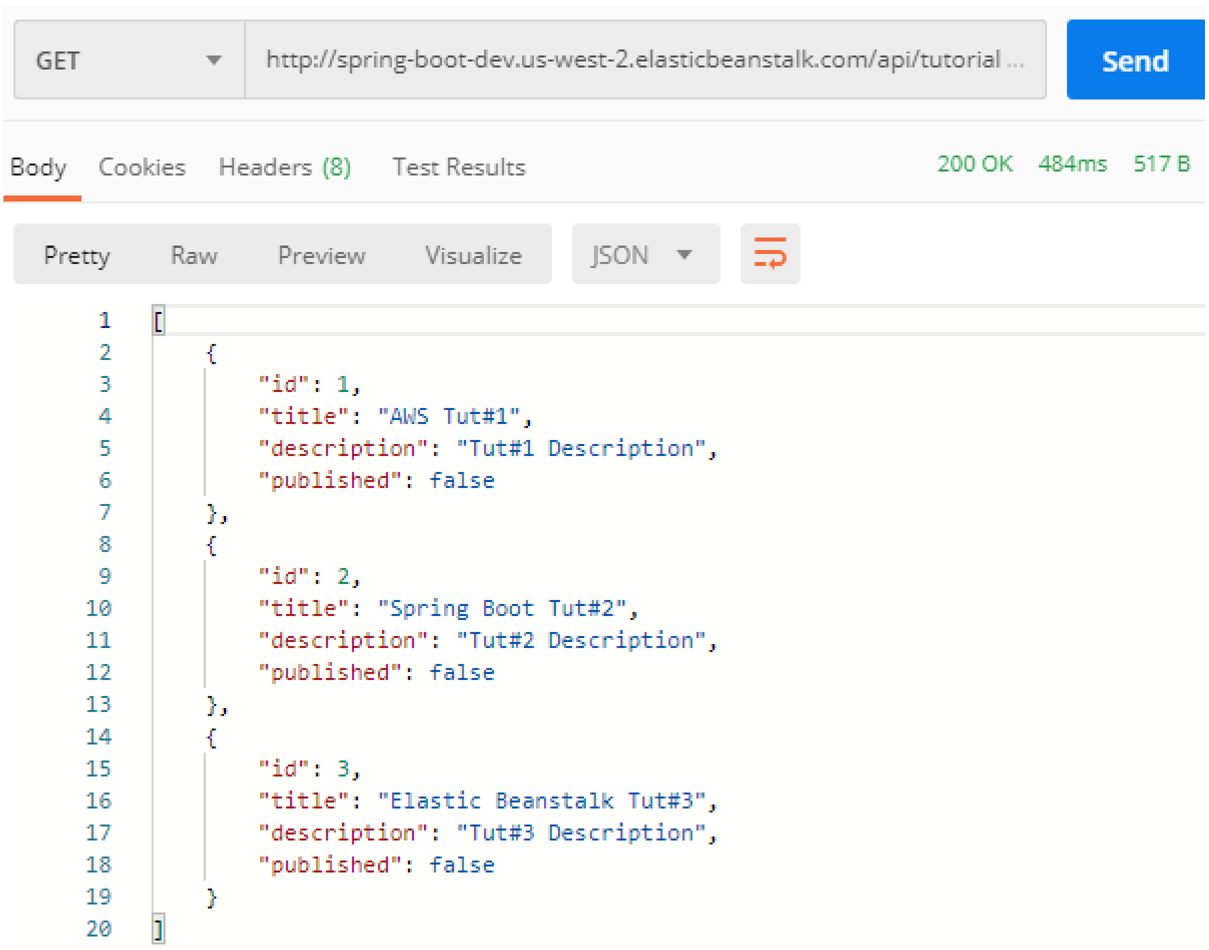
http://spring-boot-dev.us-west-2.elasticbeanstalk.com/api/tutorials

```
{
    "title": "AWS Tut#1",
    "description": "Tut#1 Description"
}
```

201 Created 847ms 336 B

```
{
    "id": 1,
    "title": "AWS Tut#1",
    "description": "Tut#1 Description",
    "published": false
}
```

– Retrieve all objects:

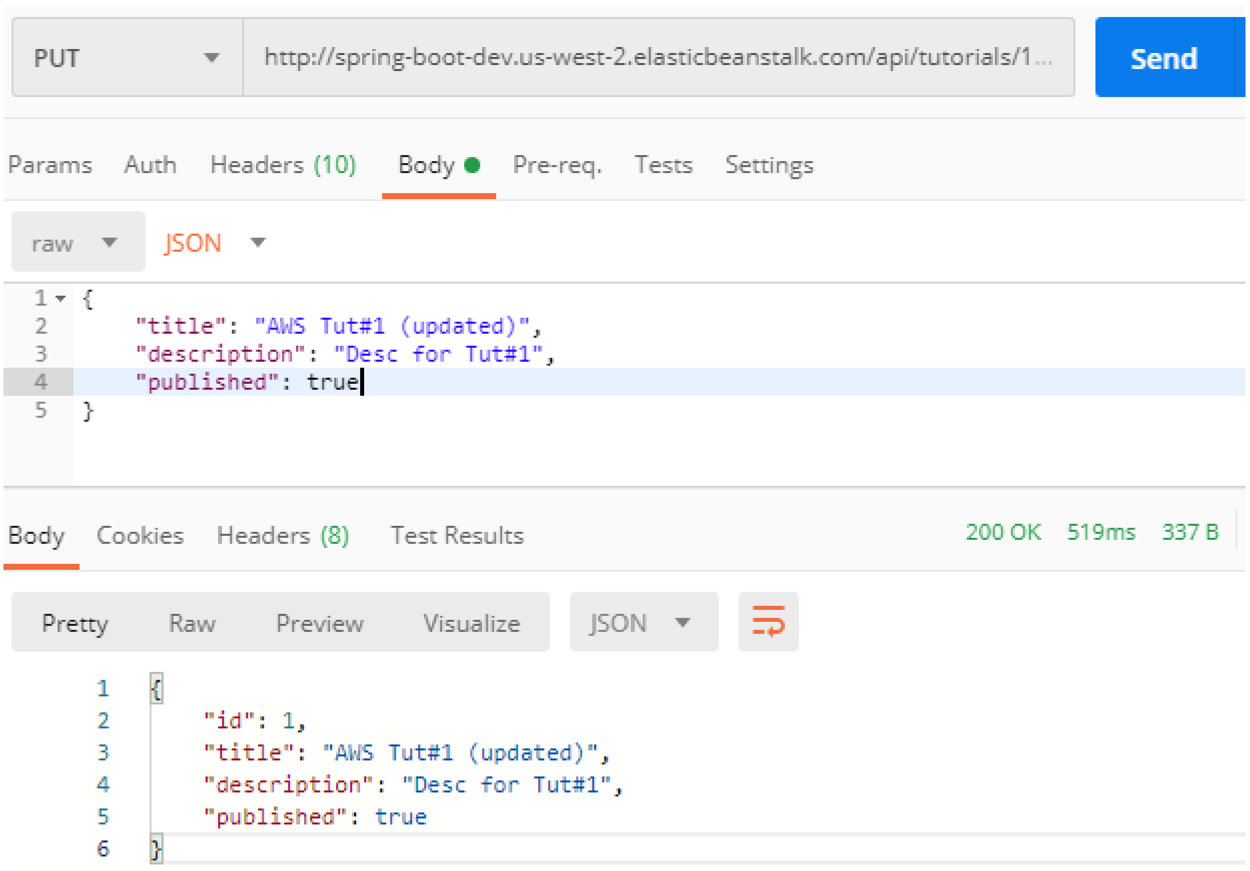


http://spring-boot-dev.us-west-2.elasticbeanstalk.com/api/tutorial ...

200 OK 484ms 517 B

```
[
    {
        "id": 1,
        "title": "AWS Tut#1",
        "description": "Tut#1 Description",
        "published": false
    },
    {
        "id": 2,
        "title": "Spring Boot Tut#2",
        "description": "Tut#2 Description",
        "published": false
    },
    {
        "id": 3,
        "title": "Elastic Beanstalk Tut#3",
        "description": "Tut#3 Description",
        "published": false
    }
]
```

– Update an object:

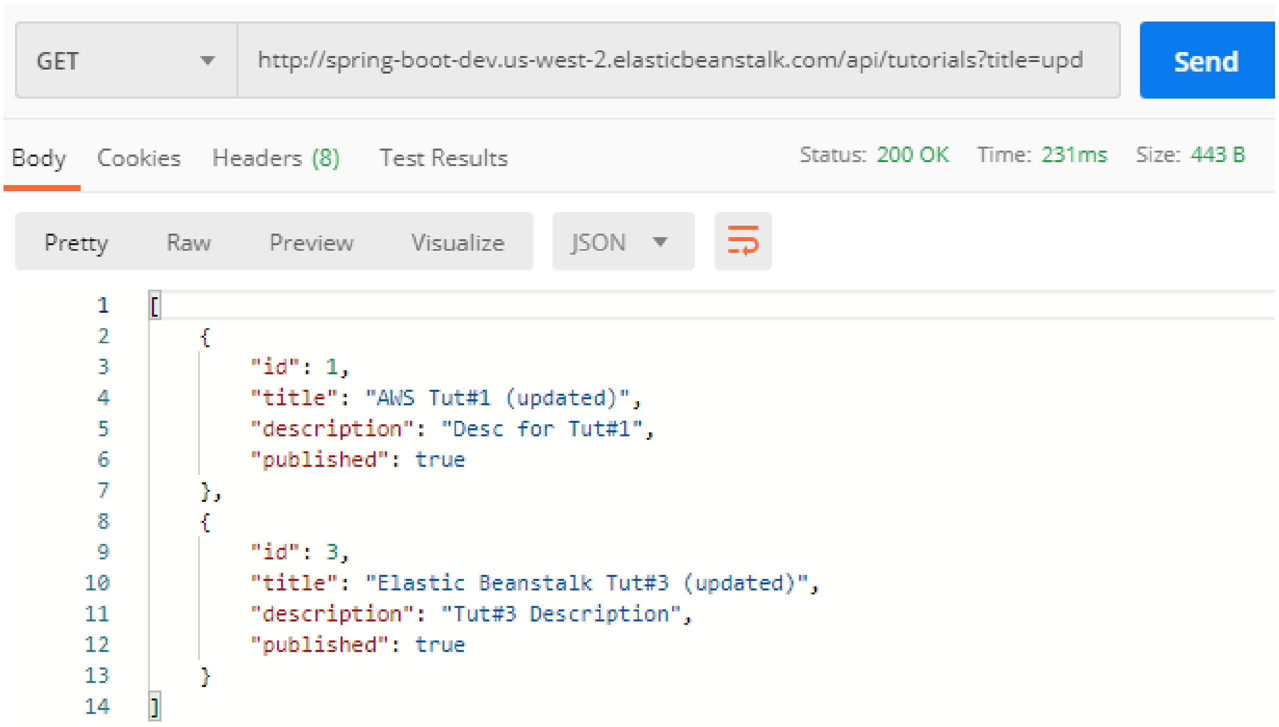

PUT
http://spring-boot-dev.us-west-2.elasticbeanstalk.com/api/tutorials/1...
Send
Params Auth Headers (10) Body Pre-req. Tests Settings
raw JSON

```
{
    "title": "AWS Tut#1 (updated)",
    "description": "Desc for Tut#1",
    "published": true
}
```

Body Cookies Headers (8) Test Results
200 OK 519ms 337 B
Pretty Raw Preview Visualize JSON

```
{
    "id": 1,
    "title": "AWS Tut#1 (updated)",
    "description": "Desc for Tut#1",
    "published": true
}
```

– Find objects by field:


GET
http://spring-boot-dev.us-west-2.elasticbeanstalk.com/api/tutorials?title=upd
Send
Body Cookies Headers (8) Test Results
Status: 200 OK Time: 231ms Size: 443 B
Pretty Raw Preview Visualize JSON

```
[
    {
        "id": 1,
        "title": "AWS Tut#1 (updated)",
        "description": "Desc for Tut#1",
        "published": true
    },
    {
        "id": 3,
        "title": "Elastic Beanstalk Tut#3 (updated)",
        "description": "Tut#3 Description",
        "published": true
    }
]
```

Conclusion

Today we've done many things for deploying Spring Boot Application on AWS EC2 Instance using Elastic Beanstalk. Now you can:

- Install EB CLI and setup AWS Beanstalk environment
- Create database, configure connection properties and set environment variables
- Build and deploy Spring Boot application on AWS
- Test the application

You may need to handle Exception with:
Spring Boot @ControllerAdvice & @ExceptionHandler example

Or: Spring Boot Pagination and Sorting example

Dockerize with Docker Compose: Spring Boot and MySQL example

Further Reading