

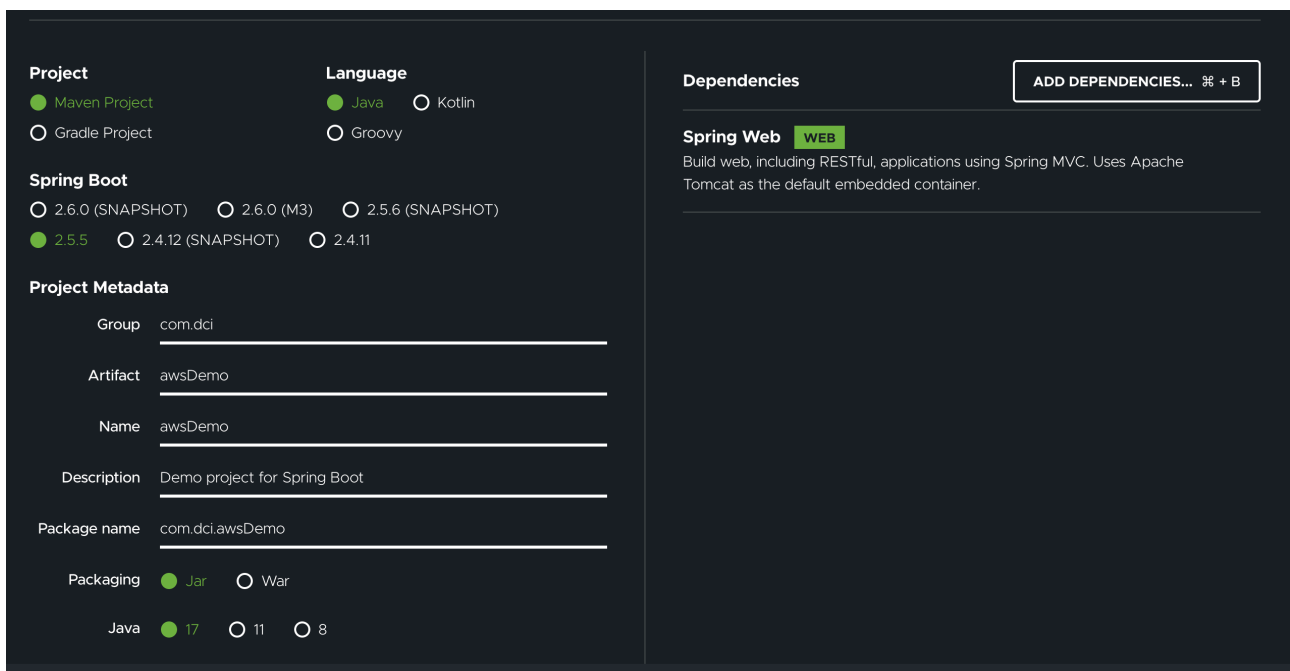
Project work (group) - 4TUs - Setup own EC2

1. Create Spring Boot "Hello world" application with one rest controller with an endpoint „/helloAws"

Use **Spring Initializr**: <https://start.spring.io/>.

Select **Maven Project**, Spring boot with version **2.5.5**, Packaging: **Jar**, **Java 17**.

Add in **Dependencies**: Spring Web.

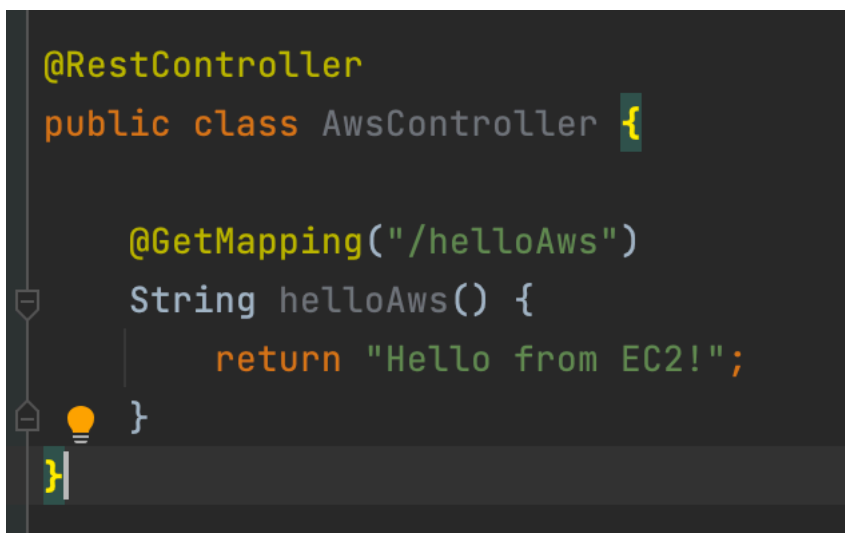


The screenshot shows the Spring Initializr web interface. On the left, under 'Project', 'Maven Project' is selected. Under 'Language', 'Java' is selected. Under 'Spring Boot', version '2.5.5' is selected. The 'Project Metadata' section contains fields for Group (com.dci), Artifact (awsDemo), Name (awsDemo), Description (Demo project for Spring Boot), and Package name (com.dci.awsDemo). Under 'Packaging', 'Jar' is selected, and under 'Java', version '17' is selected. On the right, the 'Dependencies' section shows 'Spring Web' selected with a 'WEB' tag. A button 'ADD DEPENDENCIES... + B' is visible at the top right of the dependencies section.

Download the project as a .zip, extract it and import into your IDE as Maven Project.

Make sure that your project settings has proper Java version set (of course you need to download Java 17 first if you don't have it yet).

Add a Rest Controller with an endpoint „**helloAws**”:



```
@RestController
public class AwsController {

    @GetMapping("/helloAws")
    String helloAws() {
        return "Hello from EC2!";
    }
}
```

Run locally your application and test it via browser entering url:

localhost:8080/helloAws

You should see a response: „Hello from EC2”.

2. Prepare the final jar file with Maven

Run **mvn clean install** from the console, or if you use IntelliJ you can do it from **IntelliJ Maven -> Lifecycle -> Install**

This command will create a fat jar. Now, you need to find the **target** folder - the jar will be placed there. You are going to need .jar file in the step 5.

3. Launch an EC2 Instance

Now go to AWS console and log in.

Go to **EC2** dashboard by selecting EC2 from services.

Click orange button „**Launch Instances**”.

Select **Amazon Linux 2 AMI (HVM), SSD Volume Type** - ami-07df274a488ca9195 (64-bit x86) / ami-07d397b752b4016f8 (64-bit Arm)

At the end of Instance Creation creator will ask you if you want to use existing key pair or create a new one. If you know where is your .pem file, then you can select existing one. If not, create a new pair and download the file.

Make sure you keep downloaded key pair .pem file handy.

4. Allow Inbound traffic from your computer

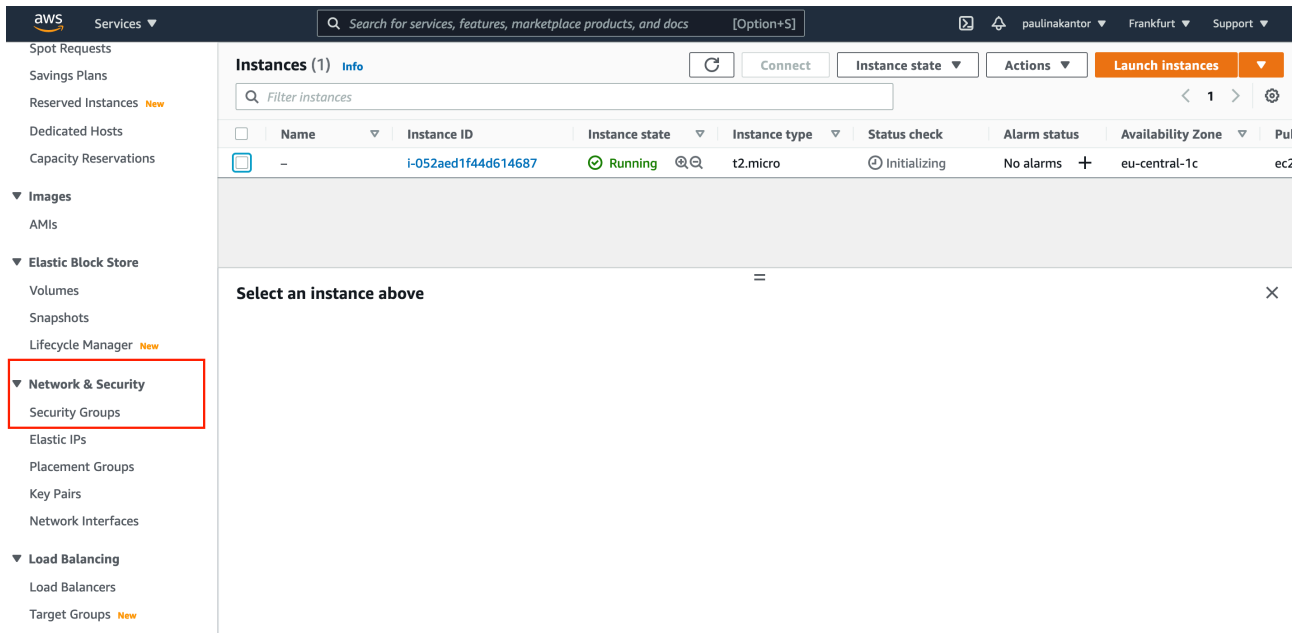
You need to allow inbound traffic from your computer's IP to be able to SSH/SCP to your EC2 instance.

First, locate your Security Group id assigned to your EC2 instance.

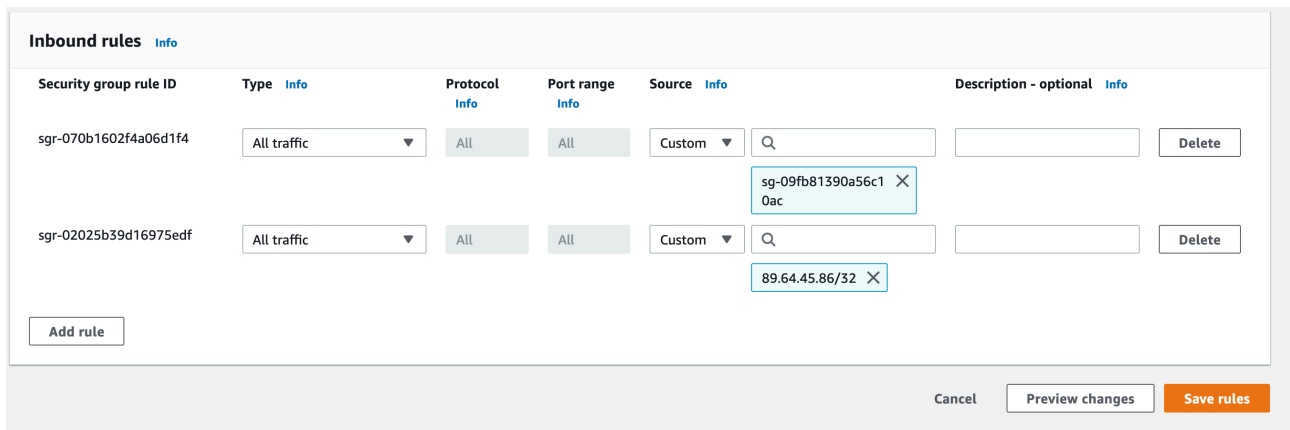
The screenshot shows the AWS Management Console interface. On the left, there is a navigation menu with categories like Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, Elastic Block Store, Network & Security, Load Balancing, and Auto Scaling. The main content area displays the 'Instances (1/1)' page. A table lists the instance 'i-052aed1f44d614687' with a status of 'Running'. Below this, the 'Instance: i-052aed1f44d614687' details page is open, showing tabs for Details, Security, Networking, Storage, Status checks, Monitoring, and Tags. The 'Networking' tab is active, showing a 'Run Reachability Analyzer' button and a 'Network interfaces (1)' section. A table below lists the network interfaces, with a red box highlighting the 'Security groups' column, showing the security group 'sg-09fb81390a56c10ac'.

ID	Subnet ID	Delete on terminate	Source / destination...	Security groups	Interface type
ipc-0aba8b5fa4...	subnet-0e55b67...	enabled	enabled	sg-09fb81390a56c10ac (def...)	Network interface

Then go to Security Groups from EC2 dashboard left panel:



Select proper Security Group. Go to Inbound Rules and click „Edit Inbound Rules”. Add the following rule by selecting in **Source** -> „My IP”. Then click Save:



5. Copy jar file to AWS EC2

Following steps will vary depending on the OS you are using. Here I presented the way if you use Linux/Mac. If you use Windows you need to download SCP client first.

Then copy your **jar** and **.pem** file with key pair to be in the same folder.

In folder above run Terminal/CLI.

First of all, set proper permissions of your private key (.pem) file so that only you can read it. If you don't set the permission then you can not connect to your instance using key pair.

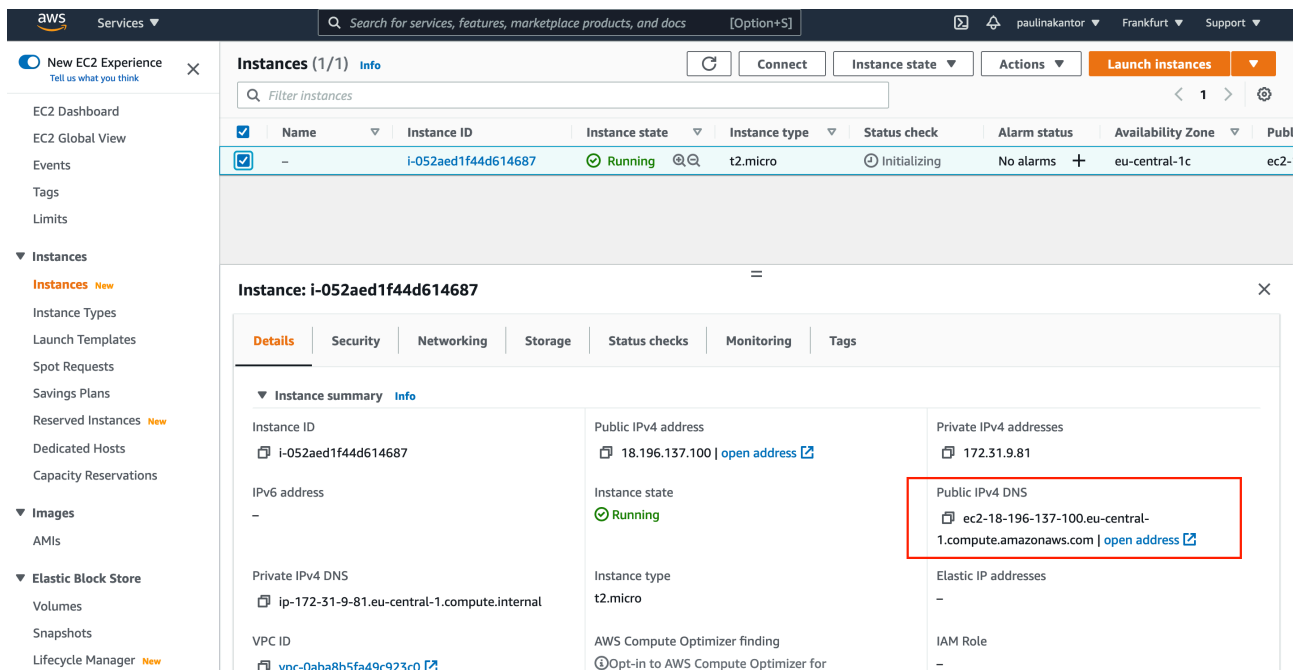
Run following command:

```
chmod 400 keyPairForAwsDemo.pem
```

Then, you can just copy .jar file into your EC2 instance home directory using following command:

```
scp -i ./keyPairForAwsDemo.pem ./awsDemo-0.0.1-SNAPSHOT.jar ec2-user@ec2-18-196-137-100.eu-central-1.compute.amazonaws.com:~
```

Of course remember to use **YOUR** names for .jar file and locate **public IPv4 DNS** address for your EC2. You can find it in EC2 dashboard:



The screenshot displays the AWS Management Console interface for an EC2 instance. The instance is named 'i-052aed1f44d614687' and is in a 'Running' state. The 'Public IPv4 DNS' address is highlighted with a red box, showing 'ec2-18-196-137-100.eu-central-1.compute.amazonaws.com'.

Instance ID	Public IPv4 address	Private IPv4 addresses
i-052aed1f44d614687	18.196.137.100 open address	172.31.9.81

Instance state	Public IPv4 DNS
Running	ec2-18-196-137-100.eu-central-1.compute.amazonaws.com open address

6. SSH into the EC2 instance and Install Java 17

Now you need to install Java to be able to run your Spring Boot application.

First, SSH to your EC2 instance:

```
ssh -i "keyPairForAwsDemo.pem" ec2-user@ec2-18-196-137-100.eu-central-1.compute.amazonaws.com
```

Then, use wget and download Java 17:

```
wget https://download.oracle.com/java/17/latest/jdk-17_linux-x64_bin.rpm
```

And last step: install Java 17:

```
sudo rpm -Uvh jdk-17_linux-x64_bin.rpm
```

7. Run the Spring Boot application on EC2

Now, whilst being still SSH to your instance, let's run our application by executing a command:

```
java -jar awsDemo-0.0.1-SNAPSHOT.jar
```

Let's test our application in the browser by entering given url:

```
http://ec2-18-196-137-100.eu-central-1.compute.amazonaws.com:8080/helloAws
```

It should return an error - we need to go to the next step.

8. Allow port 8080 on your Instance security group

Again, go to **Security Groups**, locate the proper one and add one more **Inbound Rule** allowing communication on port 8080:

EC2 > Security Groups > sg-09fb81390a56c10ac - default > Edit inbound rules

Edit inbound rules [Info](#)

Inbound rules control the incoming traffic that's allowed to reach the instance.

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info	Description - optional Info	
sg-070b1602f4a06d1f4	All traffic	All	All	Custom		Delete
sg-02025b39d16975edf	All traffic	All	All	Custom		Delete
sg-0f8195089a997d7d5	Custom TCP	TCP	8080	Custom		Delete

[Add rule](#)

Cancel [Preview changes](#) [Save rules](#)

9. Test your application again.

Now you can test given url again:

```
http://ec2-18-196-137-100.eu-central-1.compute.amazonaws.com:8080/helloAws
```

And you should see the same effect as by entering via localhost:8080/helloAws, so seeing „Hello from EC2” in your browser.