Cloud Application Architecture - Lab 1

Contents

Aim of the Laboratory

- Set the goal for this semester
- Access your AWS account
- Familiarize yourself with the AWS console, regions, and services
- Create an EC2 instance
- Run the “legacy” online shop application in a “lift and shift” manner on EC2

The Goal

Migrate and adapt a traditional application to the AWS cloud.

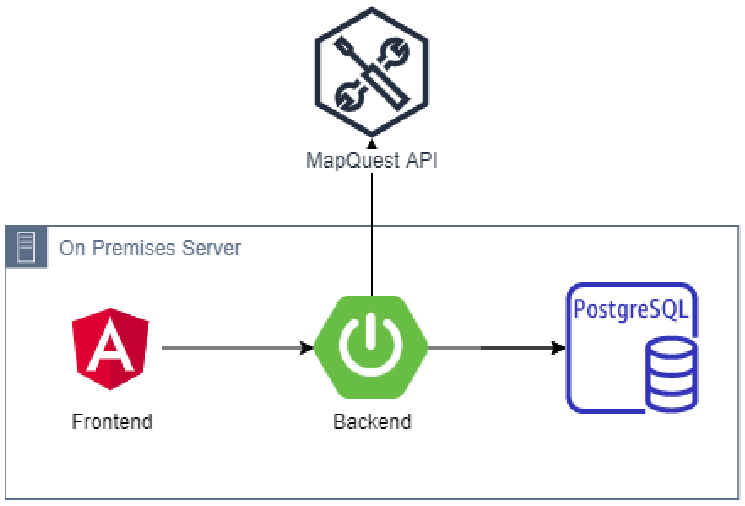


Figure 1 Initial Architecture

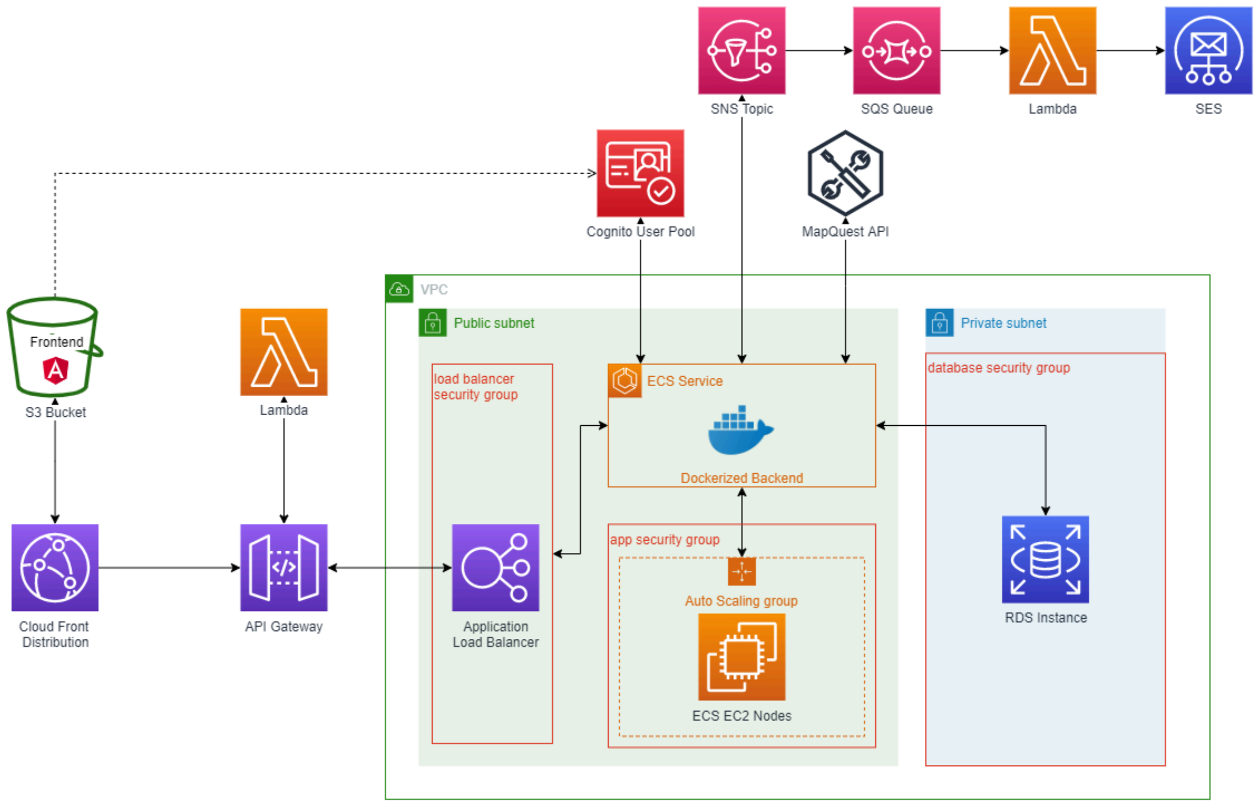


Figure 2 Final Architecture

Introduction to AWS - important concepts and terminology

AWS Intro

AWS is one of the many cloud providers available today. This wasn't always the case. Amazon launched it in 2006. The general belief is that they had lots of spare capacity after the holiday sales, but, in reality, AWS was an actual business plan. You can read more about the history of AWS here and/or watch this video presenting the history of the cloud. It took several years for any competitor to respond, which translated to an overwhelming market share owned by AWS even today and to more mature services.

It currently offers over 200 services for various use cases (hosting, AI, ML, IoT, media encoding and streaming, etc.). Some of them are very specific to a certain use case, while others are more general and most likely are the building blocks of the rest. We will focus on the latter.

AWS Regions, AZs

Amazon cloud computing resources are hosted in multiple locations worldwide. These locations comprise AWS Regions, Availability Zones, and Local Zones. Each AWS Region is a separate geographic area. Each AWS Region has multiple isolated locations known as Availability Zones.

An overview of all AWS regions is available at https://www.infrastructure.aws/.

The AWS Console

The AWS Console is a management interface that is a central entry point for all AWS services. In addition to the console, AWS cloud resources can also be managed through the AWS CLI or programmatically by using the AWS REST API/SDK.

"Online Shop" Application Overview

This semester you will be dealing with an existing Online Shop application. Throughout the laboratories, you will transform the architecture and move from a "legacy" application to a fully cloud-native architecture leveraging multiple managed services of AWS.

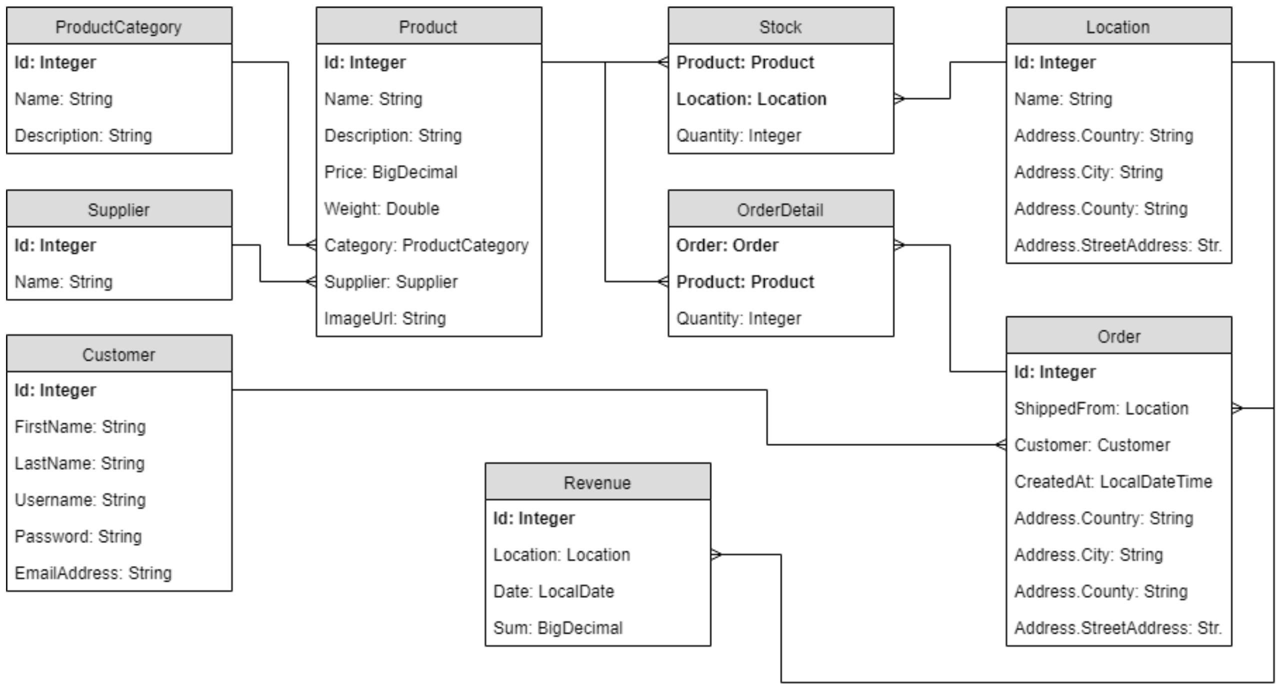
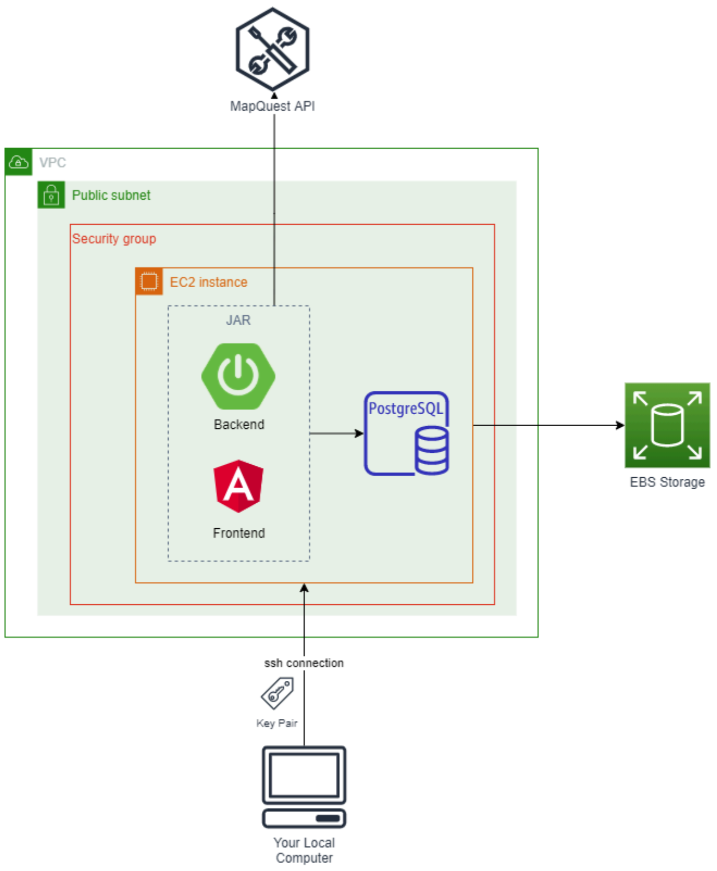


Figure 3 Data Model

The source code for the application is available here.

At the end of this lab, our application will look like this:



Lab Walkthrough

Access Your AWS Account

You should receive an email in your Outlook inbox (@stud.ubbcluj.ro) regarding your AWS account.

You can always log in at https://caa-ubb.awsapps.com/start. You will use this for all labs.

Navigate Through the AWS Services

The ***Services*** menu entry on the upper left side of the console lists all available AWS services. Have a look at the Compute, Storage, and Database sections and see if you can answer the following questions:

- What Database service does AWS offer for NoSQL databases?
- What is the S3 service?
- What service can be used to build applications using a FaaS (function as a service) paradigm?

If you cannot answer all the questions, no worries. We will review them in the latter part of the semester.

Regions

Regions serve multiple purposes. They are fundamentally different/independent clouds. They even receive new services at different rates (Ireland is the first region to receive new stuff in Europe). They are highly relevant when your application addresses the global market (e.g., Netflix) and/or when you cannot accept downtime (in extremely rare cases, an entire region might go down).

Make sure you are in the Ireland (eu-west-1) region.

Create an EC2 Instance

We will now create our first AWS virtual machine. Navigate to the EC2 service and, in the instances section, select Launch Instance.

Tags

The bigger the project/system, the more important tags get.

For EC2 (and not only), AWS uses tags to store the resource's name. So, besides the name (suggestion: *caa-lab1*), add the tag project: caacourse to both the instance and the EBS volume. This tag will help us track the resources and costs generated by the lab. We will use this tag for all resources from now on.

AMIs

An Amazon Machine Image (AMI) provides the information required to launch an instance, including the operating system and additional software. You must specify an AMI when you launch an instance. When you need multiple instances with the same configuration, you can launch multiple instances from a single AMI.

For the AMI (Amazon Machine Image), use Amazon Linux 2023 (should be the default). This AWS-optimized Linux flavor can be part of the AWS free tier. It is configured by AWS and comes with several packages pre-installed, such as the AWS CLI.

Instance Type

Amazon EC2 provides many instance types optimized for different use cases. Instance types comprise varying combinations of CPU, memory, storage, and networking capacity, giving you the flexibility to choose the appropriate mix of resources for your applications.

Select t2.micro (or t3.micro if not available). This provides a machine with 1 vCPU and 1 GB of memory, which is more than enough for our needs. It is also a burstable machine type.

Key Pair

To connect to the instance, you will need to create an SSH key pair. Select Create new key pair, provide a name, download the file, and keep it safe.

Network settings

We will connect to this instance via SSH, so it's important to leave port 22 open for incoming connections. However, you should not do this in a production environment as it exposes your instance to security threats.

Also, since the application will run on plain HTTP, allow incoming connections on that. (you might have to press Edit)

Storage

The default configuration should include an 8GB EBS volume (network storage) — no need to change anything here.

Advanced details

Leave the default values and scroll down to the User Data section. Here, you can define an automatic script. We will use it to install some prerequisites on the machine (such as the Java runtime environment, etc.). You can find the script here.

Connect to your instance via SSH

Note

The main cloud providers, including AWS, provide a simpler way to access our instances directly from the browser. In the case of EC2 instances, after selecting the instance, there should be the Connect button which will open a shell connected to the instance in a new browser tab. However, since not all providers have this option, practicing the universally accepted way (with ssh) might be useful in the future.

By default, the instance receives a public IP address and a DNS name (based on the IP address). These should be visible on the "Description" tab of the EC2 service. Copy them, as you will need them for the SSH connection.

You will need an SSH client to connect to your instance.

a) Linux

Use the command-line ssh client. On Linux, you must also ensure the private key file is not publicly viewable on your computer. Use "chmod" for this

```
chmod 400 my-key-pair.pem
```

b) Windows

Either use putty (https://www.putty.org/) or the command line client that comes with git bash (https://gitforwindows.org/). Newer builds of Windows 10 also come with a built-in SSH client (OpenSSH) (more info here).

To use putty, take a look at the official AWS guide.

For using the command line tool:

```
ssh -i /path/my-key-pair.pem ec2-user@<public_ip_address>
```

Confirm that you trust the authenticity of the host (type yes).

Legacy Application “Lift-and-shift”

Now that you have connected to your own EC2 instance, it is time to run the Online Shop on the VM.

Download the release from GitHub and run it:

```
curl -L -O
https://github.com/CAA-Course/app/releases/download/1.0/shop-1.0.jar
```

Start the application

```
sudo java -Dspring.profiles.active=with-form,local -Dserver.port=80 -jar
shop-1.0.jar
```

Note: Sudo is required for running applications on ports below 1024 (these are called privileged ports). The optimal approach would be to install a web server/reverse proxy such as Apache httpd or Nginx that will serve our application running on other ports such as 3000 or 8080.

Your application should now be accessible at the above-mentioned public IP address (don’t forget about the port). Try it out!

The default credentials are

- User: **user**
- Password: **storepass**

Hint: you can stop the app by pressing Ctrl+C. Another useful tip when working with Linux is how to exit vim – press Esc and type “:wq” (w = write/save, q = quit).

Bonus Task

Configure the app to run as a Linux service. In our case, this is nice to have since we will stop the instances at the end of each lab. In practice, this is mandatory to avoid downtime as much as possible.

Hint: One way is using *systemd* and *systemctl*.

Cleanup

In the cloud world, you pay for what you use, so make sure you terminate the EC2 instance before the end of the laboratory.

From the AWS Console, select the instance and go to Actions. In the Instance state menu, press Terminate instance.