# CSCI4145/5409: Network & Security

**Reminder: This is an individual assignment. You are not allowed to collaborate with anyone else when completing this assignment. You can borrow code and configuration snippets from internet sources that are not from students in this class, however that code must be cited and include comments for how you have modified the original code.**

## Introduction

This assignment will measure your understanding of some of the network and security mechanisms of our cloud provider AWS. This assignment assures us that you have attended the tutorials and learned about AWS VPC, and RDS, or that you have found some other way to learn these services.

## Learning Outcomes

- Learn the importance and role of Virtual Private Clouds in creating secure architectures that maximize the principle of least privilege (i.e. if a server does not need to be publicly reachable or have its traffic go across the Internet, then it should be behind some protective measure).
- Learn how to implement a VPC on AWS.
- Apply your learning by implementing one **public** facing service inside the VPC that is accessible to the public internet.
- Apply your learning by implementing one **private** service inside the VPC that is **not** accessible to the public internet, used by the public facing service.
- More experience working with AWS libraries that allow you to perform AWS operations.
- More experience building REST APIs and working with arrays in JSON.

By now you have all learned that cloud computing can be tricky and complicated. You have learned that provisioning IT resources correctly is complicated, and a small mistake can lead to hours of debugging and searching for answers. I **highly** recommend that you start this assignment early, and that you follow my tutorial so that you understand how to properly provision everything. **This architecture will be too complex for me or the TAs to debug with you individually if you make mistakes. Proceed carefully and meticulously.**

## Requirements

You will build a web application with any language or framework you like, **deployed on an EC2 instance behind a Virtual Private Cloud (VPC)**.

Your application running on EC2 will be public facing (accessible through a Public IP you assign to it on launch, or an elastic IP you create for it), and listening to **POST** requests to /store-products and **GET** requests to /list-products.

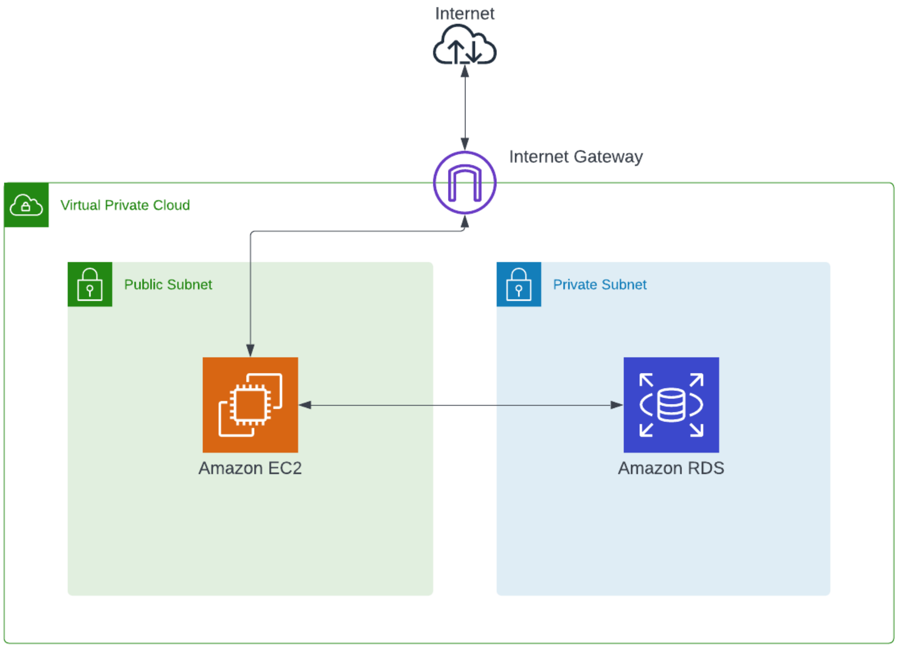
/store-products will:

- Receive and parse a JSON body.
- Connect to an AWS RDS database server running on a private subnet inside your VPC.
- Insert one record into the **products** table in the database *for each item* in the products array in the JSON body.
- Return a status 200 code if everything works, or the appropriate HTML status code if there is an error or invalid input.

/list-products will:

- Connect to the AWS RDS database running on the private subnet inside your VPC.
- Query the **products** table and return a list of all products with status 200 code.

When you are finished the system will look and function like this:



## Your RDS database:

You may choose whichever type of database you are comfortable with; I recommend Aurora because this database is covered in the tutorial. I do not recommend SQL Server or Oracle as these DBs will devour your credits.

Your database must contain a single table named **products** that has the following schema:

```
CREATE TABLE products (
     name varchar(100),
     price varchar(100),
     availability boolean
)
```

You will build the web application with any language or framework you like. I will do the same thing! Your application will “introduce” itself to mine by sending a POST request with some JSON to a URL I provide to you that begins a chain of events.

1. My app will POST to your app’s /store-products route with some JSON that gives you data to store in your RDS database.
2. Your app will retrieve the data from the POST and store the data on the database.
3. Your app will return a 200 status code with a JSON message.
4. My app will POST to your app’s /list-products route.
5. Your app will query the products data from the database and return the list of products with 200 status code.

## JSON to send to My App's /start

```
{
   "banner": "<Replace with your Banner ID, e.g., B00123456>",
   "ip": "<Replace with the IPv4 of your EC2 instance, e.g.
   74.23.154.12>",
   "ec2_instance_id": "<Replace with EC2 instance ID>",
   "rds_instance_id": "<Replace with your RDS DB identifier>",
   "aws_access_key_id": "<Replace with your AWS account’s access key
   id>",
   "aws_secret_access_key": "<Replace with your AWS account’s secret
   access key>",
   "aws_session_token_id": "<Replace with your AWS account’s session
   token>"
}
```

## JSON My App sends to Your App's /store-products

When you POST to my app's /start endpoint with valid JSON my app will immediately interact with yours by sending a POST with the following JSON to your app's /store-products endpoint:

```
{
   "products": [
      {
         "name": "<a name>",
         "price": "<a price>",
         "availability": <true or false>
      },
      {
         "name": "<yet another name>",
         "price": "<yet another price>",
         "availability": <true or false>
      },
      ...
   ]
}
```

Note the following:

- <>'s are placeholders intended to mean you replace this with some other text.
- … indicates that the "products" object is an array that can have any length, including being empty!

### Your App’s Response to /store-products POSTs:

When my app posts the JSON above to your /store-products endpoint, after you save the data in your DB, you must return a 200 status code and the following JSON:

```
{
   "message": "Success."
}
```

### Your App’s Response to /list-products GET:

When my app posts the JSON above to your /list-products endpoint, after you retrieve the data from your DB, you must return a 200 status code with the list of products in JSON:

```
{
   "products": [
      {
         "name": "<a name>",
         "price": "<a price>",
         "availability": <true or false>
      },
      {
         "name": "<yet another name>",
         "price": "<yet another price>",
         "availability": <true or false>
      },
      ...
   ]
}
```

## Marking Rubric

In this class I'm not very concerned about the quality of the code you write, if you write bad quality code it is you that will suffer in maintaining and supporting it. I care that you can meet the learning objectives defined at the top of this document, and I can verify this by simply verifying the correct behaviour of your app's interaction with mine.

Your submission will be marked by the app that I will write, my app will:

- Listen for requests to /start and initiate the check process.
- The check process:
  1. Records the IP you send to /start in DynamoDB.
  2. Uses your provided credentials to retrieve your EC2, RDS and VPC configurations.
  3. Sends a POST request to your IP's /store-products API.
  4. Sends a GET request to your IP's /list-products method.

Your mark is entirely based on the success of steps 1 - 4:

- **Your EC2 should only be open to HTTP, HTTPS and SSH: 18%**
- **Your RDS should be in a private subnet: 17%**
- **Your RDS instance should only be open to EC2 instance: 18%**
- **Your EC2 and RDS should be in the same VPC: 17%**
- **Your app's response to /store-products should be successful: 15%**
- **Your app's response to /list-products should give the products information: 15%**

I care about learning. If it takes you multiple attempts to learn the outcomes of this assignment, that's fine with me because in the end you learned! **Therefore, I will build my app such that only the performance of your highest scoring call to /start counts towards grading.**

Because your mark is entirely results based it makes sense for you to spend time testing to ensure it works! I again recommend that you use a tool like Postman to test your app's behaviour.

## How To Submit

Create a folder in your repository labeled **A3**. Put all your source code in this folder and push it to your individual repository on GitLab **before the assignment deadline.**

I will publish the IP of my running app a few days before the assignment deadline, I will leave it running. You must build, provision, and execute your app such that it makes its call to /start **before the assignment deadline.**