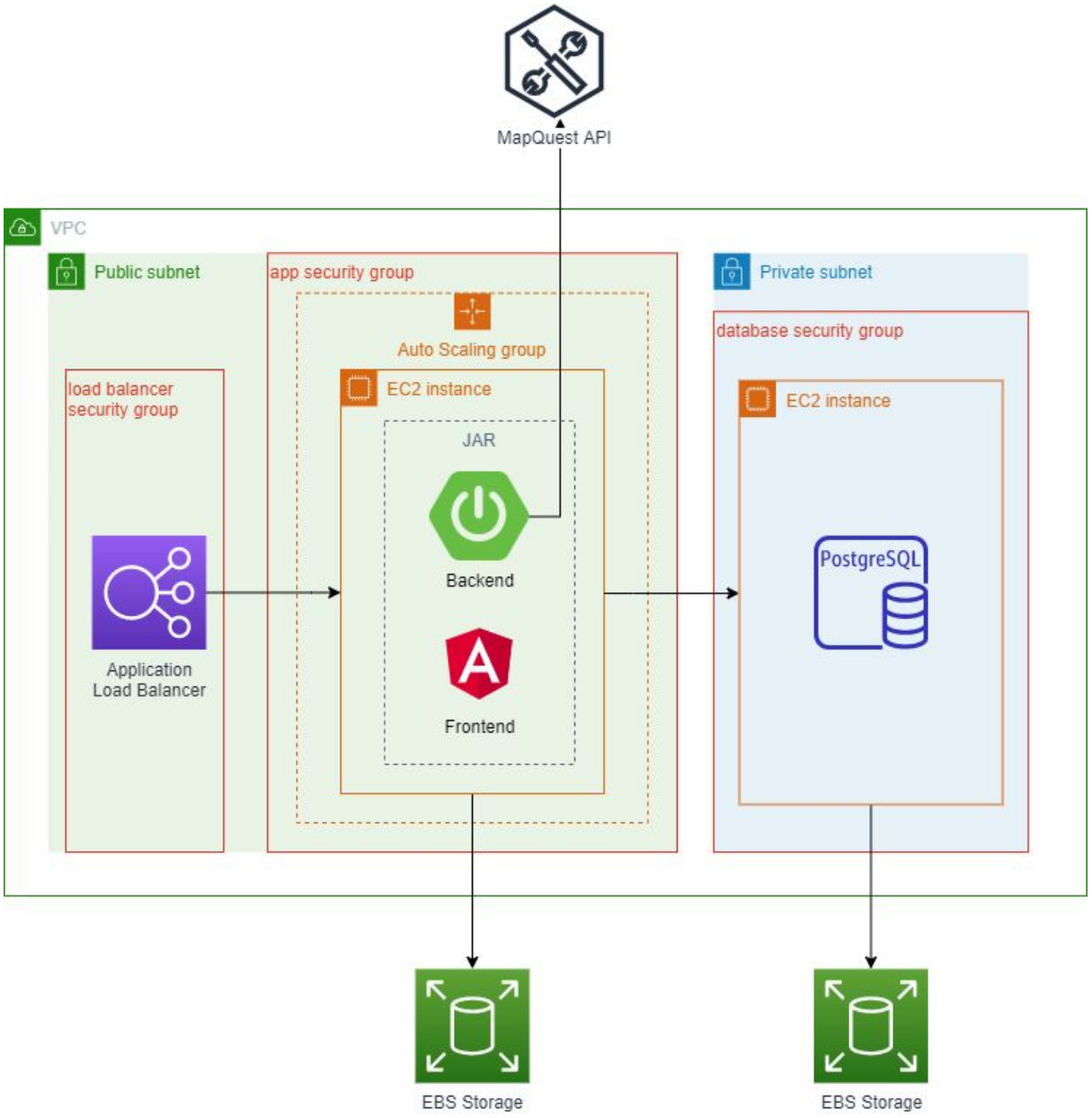
Cloud Applications Architecture - Lab 2

Aim of the lab

- Understand AWS networking concepts
- Decouple application and database layers
- Make the application layer scalable

The New Architecture



Concepts and Terms

CIDR IP Ranges

Classless Inter-Domain Routing (CIDR) is a method for allocating IP addresses and for IP routing. IP addresses are described as consisting of two groups of bits in the address: the most significant bits are the network prefix, which identifies a whole network or subnet, and the least significant set forms the host identifier, which specifies a particular interface of a host on that network. This division is used as the basis of traffic routing between IP networks and for address allocation policies.

Whereas classful network design for IPv4 sized the network prefix as one or more 8-bit groups, resulting in the blocks of Class A, B, or C addresses, under CIDR address space is allocated to Internet service providers and end-users on any address-bit boundary. In IPv6, however, the interface identifier has a fixed size of 64 bits by convention, and smaller subnets are never allocated to end-users.

IP address construction:

Class A network: Everything before the first period indicates the network, and everything after it specifies the device within that network. Using 203.0.113.112 as an example, the network is indicated by "203" and the device by "0.113.112."

Class B network: Everything before the second period indicates the network. Again using 203.0.113.112 as an example, "203.0" indicates the network and "113.112" indicates the device within that network.

Class C network: For Class C networks, everything before the third period indicates the network. Using the same example, "203.0.113" indicates the Class C network, and "112" indicates the device.
More about CIDR: https://en.wikipedia.org/wiki/Classless_Inter-Domain_Routing

VPC

Amazon Virtual Private Cloud (Amazon VPC) lets you provision a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network that you define. You have complete control over your virtual networking environment, including selection of your own IP address range, creation of subnets, and configuration of route tables and network gateways. You can use both IPv4 and IPv6 in your VPC for secure and easy access to resources and applications.
More about VPC:
https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html

Subnets

The way IP addresses are constructed makes it relatively simple for Internet routers to find the right network to route data into. However, in a Class A network (for instance), there could be millions of connected devices, and it could take some time for the data to find the right

device. Therefore, subnetting comes in handy: subnetting narrows down the IP address to usage within a range of devices.

Because an IP address is limited to indicating the network and the device address, IP addresses cannot be used to indicate which subnet an IP packet should go to. Routers within a network use something called a subnet mask to sort data into subnetworks.
More about subnets: https://www.cloudflare.com/learning/network-layer/what-is-a-subnet/

Security Groups

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. When you launch an instance in a VPC, you can assign up to five security groups to the instance. Security groups act at the instance level, not the subnet level. Therefore, each instance in a subnet in your VPC can be assigned to a different set of security groups.
More about security groups:
https://docs.aws.amazon.com/vpc/latest/userguide/VPC_SecurityGroups.html

Load Balancer

Load balancing refers to the process of distributing a set of tasks over a set of resources (computing units), with the aim of making their overall processing more efficient.
Elastic Load Balancing, the AWS service for loading balancing supports the following types of load balancers:

- **Application** - This is an actual proxy between the internet and your application. It receives a request from a client and makes another request (with the same data) to your application. It offers tons of features and it suits very well in most cases. One important tip about it is that since the ALB creates another request, but, for some reason, you need the IP address of the original client, you can look at the request header x-forwarded-for.
- **Network** - You can look at it like at a (very sophisticated) network router. While the ALB operates at layer 7 (HTTP, WebSockets) of the OSI model, the NLB handles traffic at layer 4 (TCP/UDP) thus working with packets. You lose some features of the ALB, but gain massive performance (and scalability) and the request looks like it came directly from the original client. Also, interestingly enough, it is (slightly) cheaper than an ALB (mostly because you have to configure it more).
- There is a third option, classic, but it's deprecated.

More about load balancing:
https://docs.aws.amazon.com/AmazonECS/latest/developerguide/load-balancer-types.html#clb

Auto-Scaling Groups

An Auto Scaling group contains a collection of Amazon EC2 instances that are treated as a logical grouping for the purposes of automatic scaling and management. An Auto Scaling group also enables you to use Amazon EC2 Auto Scaling features such as health check

replacements and scaling policies. Both maintaining the number of instances in an Auto Scaling group and automatic scaling are the core functionality of the Amazon EC2 Auto Scaling service.
More about auto-scaling groups:
https://docs.aws.amazon.com/autoscaling/ec2/userguide/AutoScalingGroup.html

Lab Walkthrough

1. Understand the default VPC

Navigate to the AWS VPC service and analyze the default VPC. Answer the following questions:

- What is the IPv4 CIDR range of the VPC? ([first IP] - [last IP])
- What subnets are available by default? Are they public or private?

2. Create a private subnet

A subnet is a sub-range of IP addresses within the VPC. AWS resources can be launched into a specified subnet. Use a public subnet for resources that must be connected to the internet and use a private subnet for resources that are to remain isolated from the internet. In this task, you will create a private subnet into the default AWS VPC.

- Go to the **VPC service**
- In the left navigation pane, click *Subnets*.
- Click *Create subnet* and configure:
    - *Name tag*: Private Subnet
    - *VPC*: There should be only one, the default one
    - *Availability Zone*: Select the first AZ in the list (do not choose *No Preference*)
    - *IPv4 CIDR block*: 172.31.64.0/24

Your Default VPC now has 4 subnets (3 default public and the 1 private you created).

What is the IP range of your new subnet? (first and last IP address)

3. Configure the security groups

We need a security group for the application layer allowing app traffic and, optionally, SSH, and another security group for the database layer allowing traffic from the application layer. We will start with the application layer:

- Navigate to the **EC2** service
- Go to **Security Groups** (navigation pane on the left)
- Click Create security group and configure:
    - Security group name: **App-SG**
    - Description: Allow application traffic
    - VPC: **the default VPC**

    - **Add an inbound rule** to allow traffic on the port of the app from all sources
    - Add the tag **project: caacourse**
    - **Create** then close

Now, let's create the database security group. Follow the same steps, but name it **Database-SG** and **allow traffic for Postgres only from App-SG**.

4. Spin up the instances

We are going to use CloudFormation to provision our database and application instances. For this, open the CloudFormation service and press **Create Stack** (with new resources):

- *Template is ready* option should be selected
- In the Amazon S3 URL paste the following: https://caa-lab-templates.s3-eu-west-1.amazonaws.com/lab2.yaml
- Press next
- Give your stack a name (*Lab2* maybe?)
- Fill in the parameters so CloudFormation configures the instances exactly as we need (these are part of the template you are using). Each of them should have a description.
- For ***DbImageId***, your job is to find it. Hint: It refers to an **AMI** image ID and the owner is *026709880083*.
- Press next and create the stack.

Until the instances are created and initialized, download the template from above and take a look at it.
The app should be up and running in ~2 minutes;

Next, we want to introduce horizontal scaling (i.e. increase the number of instances running the app). The main steps to achieve this are as follows:

- Create a **Launch Template**
- Create an **Auto-Scaling Group**
- Create a **Load Balancer**

5. Create the launch template

We use launch templates to tell AWS how to launch new instances when needed.
Create a new one by right-clicking on the app instance and pressing *Create template from instance*:

- Give it a meaningful name
- Check the *Auto Scaling guidance* checkbox.
- AMI and instance type should already be set
- Select the app security group
- Remove the network interface
- In the advanced details set the *Shutdown behavior* and *Stop - Hibernate behavior* to "*Don't include in launch template".*
- *Create the launch template*

6. Create the ASG

Using the launch template, we now can create an auto-scaling group that will ensure that our required number of instances is fulfilled:

- Go to Auto Scaling group (left menu)
- Press Create
- Give it a meaningful name and select the template you just created
- On the next step, for the subnet, set at all 3 default subnets
- Skip step 3
- On step 4 set the **desired capacity to 1**, **minimum capacity to 1, and maximum capacity to 2.**
- Skip the notifications, add the usual tag, and create the ASG.
- Go to the EC2 instance list. A new instance should be initializing. Add the already existing app instance to the ASG (right-click, instance settings).

7. Create the Load Balancer

Follow the official guide to create an Application Load Balancer (without HTTPS). Remember to create a new security group allowing traffic on port 80 (don't use the one for the app). The ALB will then route the traffic on the app port (8080; set on step 4).

After it's created, attach it to the ASG by following this guide.

8. Test the app

What do you notice? Can you explain?
Look through the load balancer settings. Can you identify any settings that might help?

9. Cleanup

Delete all the resources (we will start with another CloudFormation template in the next lab):

- ALB
- ASG (this should also terminate the instances that are part of it)
- CloudFormation stack (this should terminate the DB instance)
- EC2 instances if any left