Cloud Application Architecture - Lab 5

Aim of the Laboratory

1. Complete previous labs
2. Recap of previous labs
3. IAM

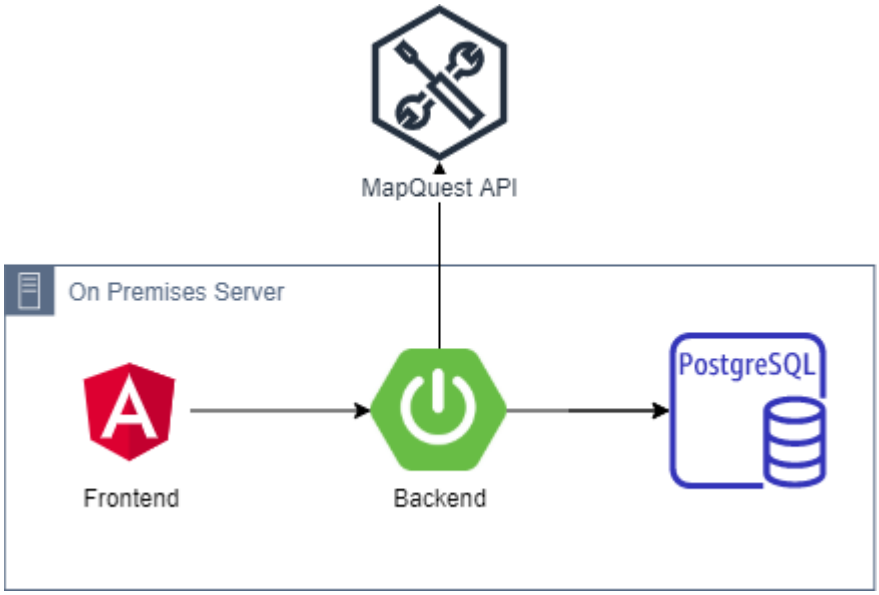
Completing Previous Labs

The first half of the lab is about getting a better understanding of the previous labs. If there are any tasks that you want to try again (or didn't manage to do them the first time) or if certain parts were not clear, now is the time to work on them.
The main points so far were:

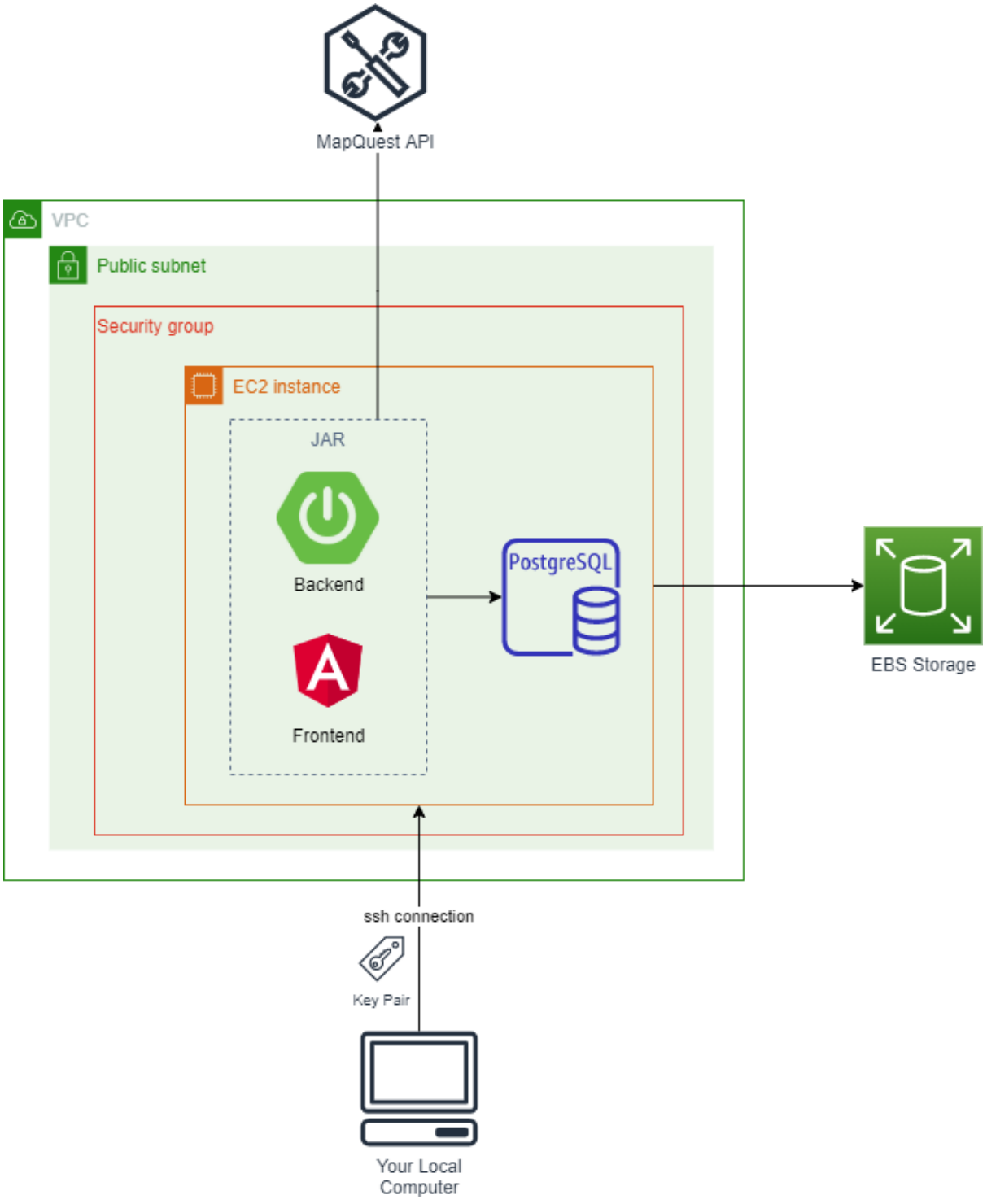
- The AWS account and how to use it (we used the web console so far)
- SSH
- EC2 instances
- VPCs, Subnets, and Security Groups
- Auto-Scaling Groups and Load Balancers
- S3 and CloudFront
- CloudFormation
- RDS and containers related services

The Road so Far

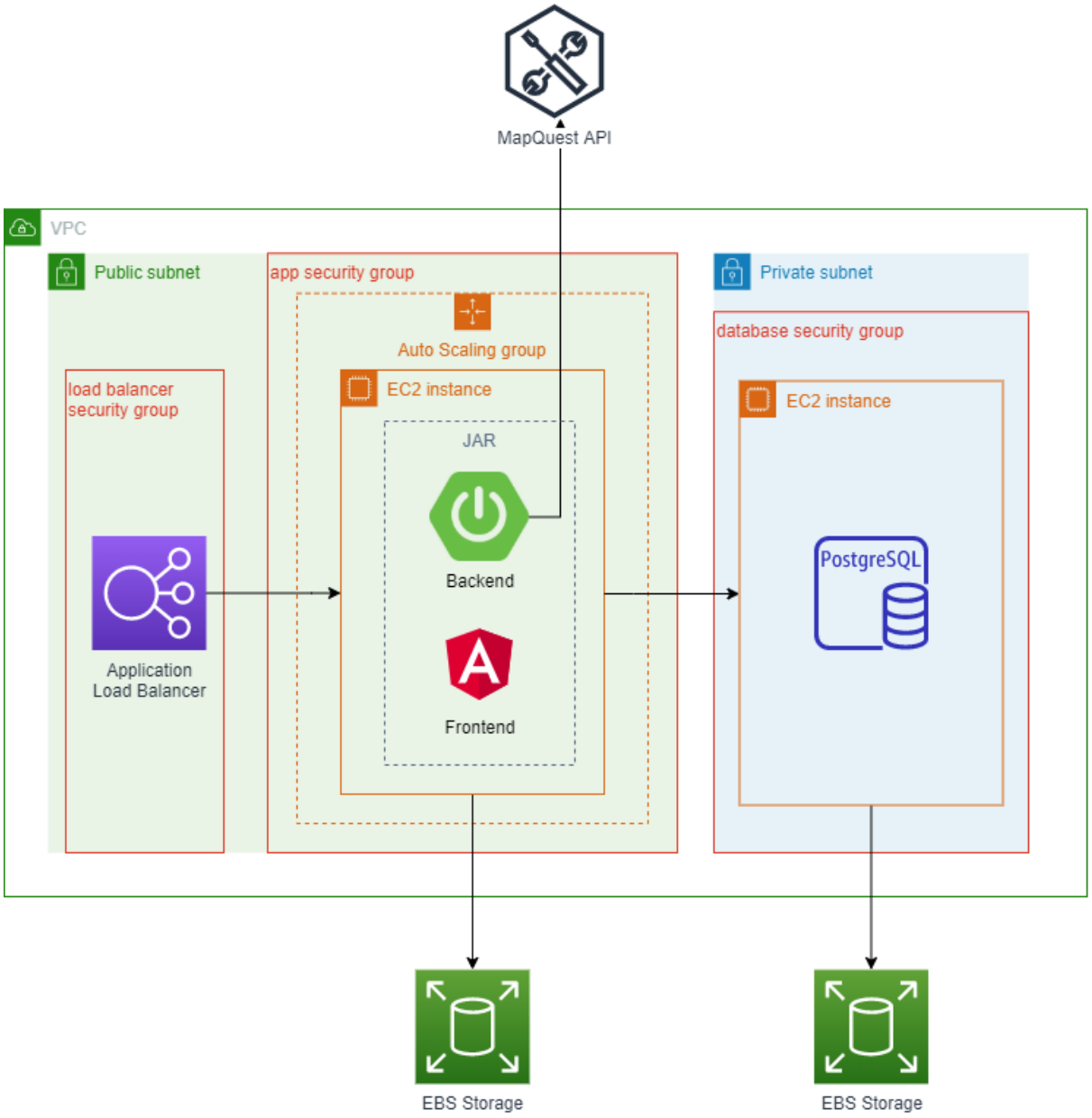
1. On-Premises



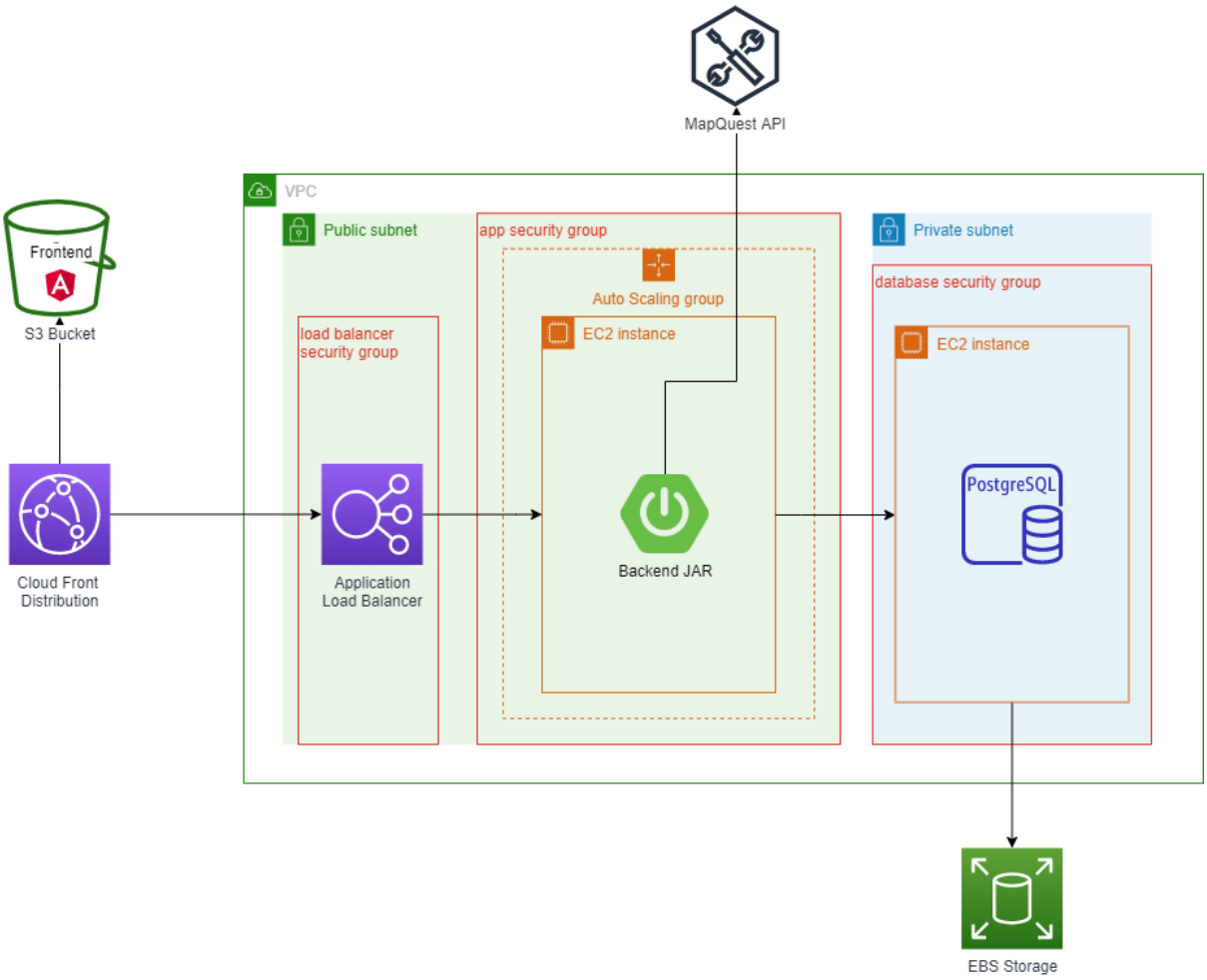
2. Lift & Shift with EC2



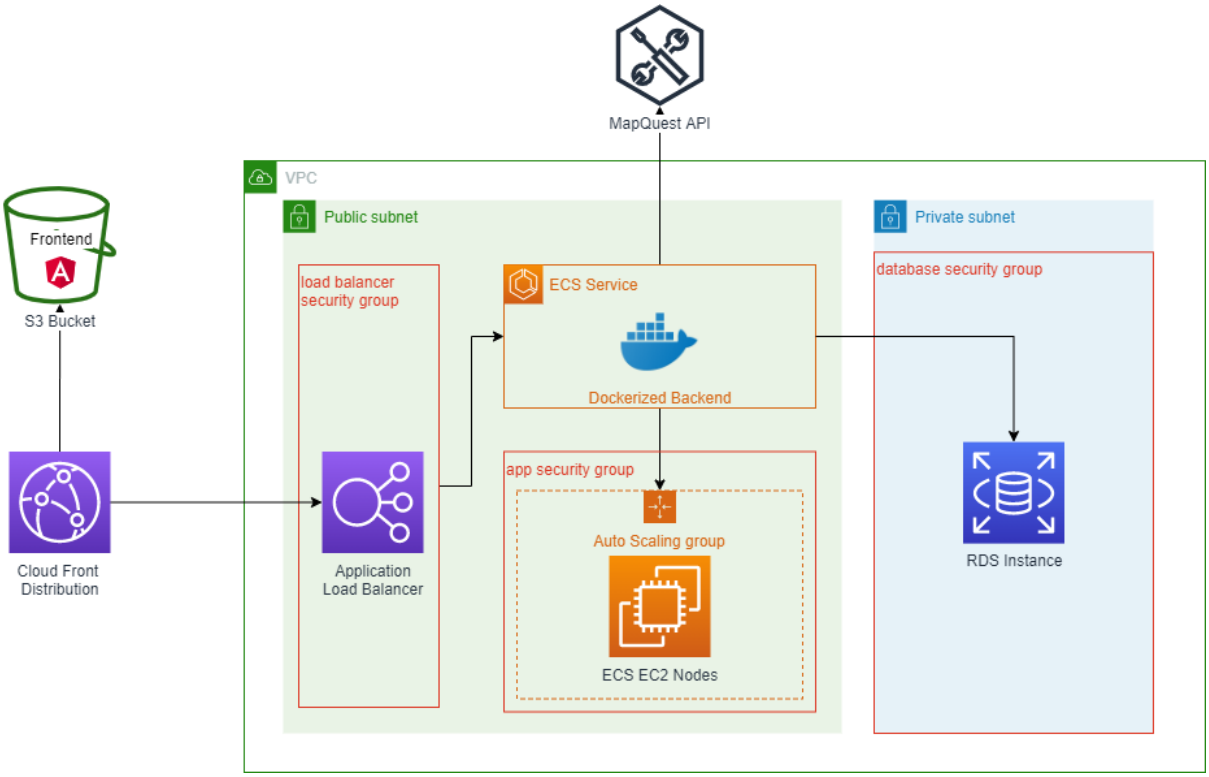
3. High Availability and Scalability



4. Frontend on Separate Hosting



5. Leveraging Managed Services and Containers



Identity & Access Management

This is the central service/place to manage everything related to access across all AWS services. This is the first service you should interact with after creating a new AWS account.
Your first account is called the root account. This is similar to Internet Explorer; you should use it only to create other 'accounts' (users). The main reason/concern is security of course - if someone gains access to your root account there is no quick way to stop him/her which could lead to considerable expenses.

Today we will assume that you are responsible for managing the users of your account. You have a new developer that joined your team and he needs **full access to EC2** and **read access to RDS**.
Your task is to create a new group called **Devs** with the right policies attached and a new user **John** for your new teammate.
Your user, *student*, might not be allowed to create IAM entities. To temporarily elevate your privileges, assume the role **IAM-Admin-Role** (the account ID is the one from the top-right menu).

Users

IAM users represent entities that interact with the AWS resources under your account. They are basically a set of credentials (username/password or access keys) that offer access to the AWS APIs/console. Each user is given a certain set of permissions adhering to the least privilege principle.
For example, in our case:

- Someone (or something) created an AWS account
- Authenticated using the root credentials
- Created an admin user (all permissions)
- Switched to that user
- Created your user and assigned the required permissions to interact with the VMs, DBs, logs, and other services covered during this training (by default a user cannot do anything)
- Sent you the username and password

Question: Do users expire? How about passwords?

Groups

Groups are what you probably expect to be - groups of users. The main benefit is that groups allow us to maintain permissions of multiple users (the members of a group). In the previous example about how your user was created, the administrator did not specify each permission individually (even though it is possible); he/she just

assigned your user to the 'Students' group from where your user inherited the permissions.
There are no pre-defined groups. In our case, the admin created the 'Students' group that you are part of.

Policies

At a high level, policies are documents (JSON) describing what an entity can or cannot do. There are 2 main types of policies (these are the most relevant for developers):

- Identity-based: they specify what the entity having it attached can do.
- Resource-based: they are attached to resources (most commonly to S3 buckets) and describe who/what has access to it and to what extent - the who/what is called the principal.

There are other policy types like policies to define permission boundaries which are like policies for policies.

Of course, you can always check the AWS documentation here. Besides some intimidating blocks of texts and diagrams, it has pretty good example policies to cover certain scenarios.

Roles

Roles do not represent an entity, but they can be assumed by an entity. You can use roles to provide temporary access to AWS resources of your account to your applications or to external users (from another AWS account or outside of AWS - e.g. identified by Google). When you create a role, you define who/what can assume it and what permissions it provides.

This is the preferred way to provide permissions to your applications. If we wouldn't have roles, we would have to provide credentials to our applications (i.e. store the username and password on the VM where the app is running).

Cleanup

Delete the following:

- ECS cluster
- ECR image
- RDS instance (no final snapshot)
- CloudFormation stack
- Any manually created EC2 instances and/or load balancers
- The IAM Users, Groups, Policies, and Roles