Block 1.

The design is based on the following requirement:

"Create and develop an infrastructure using an *IaC*, only *Terraform* or *Cloud Formation*. The infrastructure should create **2 EC2** instances in the same virtual cloud and subnet. Both instances would host services in the future. These services would be running in ports: 443 using **TCP**, 1337 using **TCP** and 3035 using **TCP** and **UDP**. Each instance should be accessed via SecureShell on the default port using a different key for each instance."

To meet these requirements, the architecture was design in the following way:

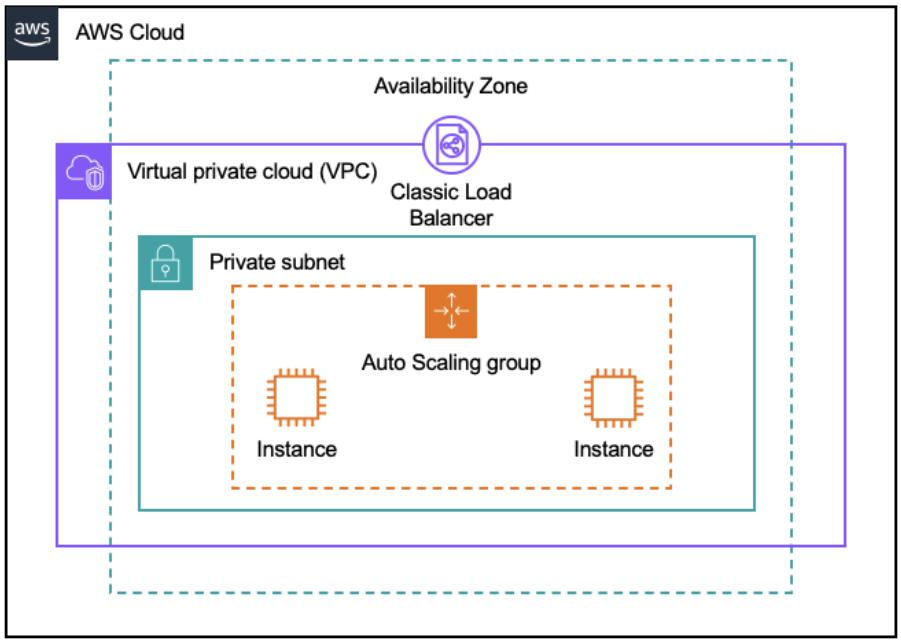
1. Ubuntu was chosen as the Linux distribution for the EC2 instances. Because Ubuntu is one of the most widely used Operating Systems in the software and development community, it is versatile and efficient.

2. A security group was design to ensure communications trough the specified ports. It was attached to the EC2 instances.

3. An Elastic Load Balancer was added to ensure distributed balancing of the workloads to instances.

4. An alarm was configured to trigger the launch of a new instance. The alarm is based on cpu utilization once it reaches the 80% cpu threshold, the autoscaling policy is triggered which launches a new instance based on the autoscaling group which has a maximum of 4 instances.

This is the main design for the requirements. The following diagram shows the architecture.



Block 2.

The design is based on the following requirement:

“Design an infrastructure to scale efficiently the deployment of the WebApp as well as the storage of the images in the S3 AWS Bucket. The design should contain all the elements. This design should be written down in the Report”.

To meet these requirements, the architecture was design in the following way:

1. It was decided to use an EKS Cluster as a solution to the application architecture design. The decision to use K8s is based on the architecture offered by this technology, allowing decoupled architecture elements and greater flexibility. Considering all the benefits of the microservices architecture over the semi-coupled architecture of virtual machines.

2. The containerization of the application allows it to be easily deployed and moved at the time of any change in the architecture.

3. A network was defined for the EKS cluster.

4. Internet access was granted to the EKS cluster to allow online access to the application.

5. All the roles and policies necessary for the management of the EKS cluster were defined, as well as a role that would allow access to the S3 bucket, which will serve as storage for the application images.

6. A bucket was created with a 30-day retention policy.

7. The EKS cluster was defined with a configuration that ensures that there is always at least one node running, and in case of high load or increasing demand, it can be increased to a maximum of two nodes.

8. A load balancer service was created in the EKS cluster for load balancing in the cluster.

9. Horizontal scaling limits were set, which means that Kubernetes will automatically adjust the number of replicas based on the load. If the load or CPU utilization exceeds 80%, Kubernetes will increase the number of replicas up to a maximum of 5. Similarly, if the load decreases and resource utilization drops below the target, Kubernetes will decrease the number of replicas to the specified minimum.