Checkout.com solution

Introduction

For the challenge presented I decided to focus on the hosting side and managing the entire system, both infrastructure and application deployment.

I also tried to automate as much as possible and make the solution independent of any manual configuration as this will be prone to errors in the long run.

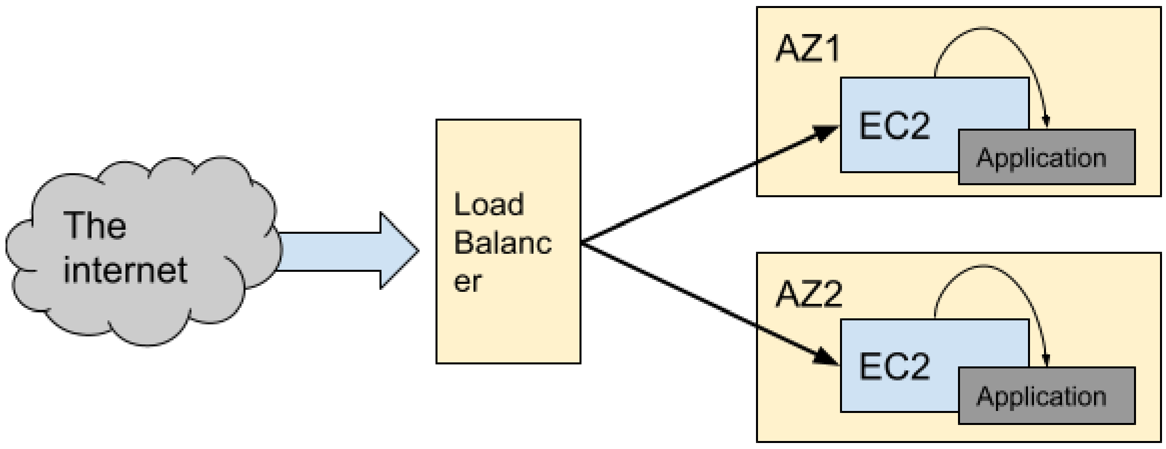
Scaling

To scale the solution in this exercise and add more servers running more containers we can simply increment the ***instance_count*** variable on the **terraform.tfvars** file and run ***terraform apply*** again. This will simply add more EC2 instances running the application behind the loadbalancer and the traffic will be shared across the pool of instances.

Instances will be created in different "Availability Zones" (AZ) which also makes the solution more resilient as the instances are spread across different AZ.

This is not ideal though, as scaling this way may take more time than desired, and it's not automated, human intervention is required to scale the solution. This could be a possible scenario for an application that we know will have more demand in the near future, and that demand will keep constant. It's not a solution for an application that requires scaling and downsizing on a regular basis.

The scenario is illustrated on the diagram below. EC2 instances are created in different AZs and the load balancer is responsible for distributing the incoming traffic across the different regions.


The internet
Load Balancer
AZ1
EC2
Application
AZ2
EC2
Application