AWS - Capstone Project

Amir Ali, Stanislaw Matuszynski, Mateusz Kierznowski
April 29, 2022

Abstract

This document summarises the work of our team on creating AWS architecture - Capstone Project using MySQL Database, Balance Loader, Auto Scaling, specifying IAM controls, and setting VPC and HTTP protocols.

1. Overview:

- Deploy a PHP application that runs on an Amazon Elastic Compute Cloud (Amazon EC2) instance.
- Create a database instance that the PHP application can query.
- Create a MySQL database from a structured query language (SQL) dump file.
- Update application parameters in an AWS Systems Manager Parameter Store.
- Secure the application to prevent public access to backend systems.

2. SQL Database

For the purpose of this project Amazon Relational Database Service: MySql Database has been chosen. The reason beyond is that Amazon RDS supports an array of database engines to store and organize data. It also provides methods that help with relational database management tasks, such as data migration, backup, and recovery. We select the Multi-AZ DB instance in order to provide a backup option while instance when primary DB encounters an error/bug. The class for the database is t3.micro (as provided in the task). With additional function enabled: Auto Scaling enables the Aurora DB cluster to handle sudden increases in connectivity. Additionally selected DB has no public access.

3. Application Load Balancer

In our architecture, we implement Application Load Balancer which helps with distributing network traffic such that queries/data flows across multiple servers. So it ensures any server will not be overloaded. In our application load balancers catch information from the internet, en distributes data into our 2 different web services.

The role of a load balancer is sometimes likened to that of a traffic cop, as it is meant to systematically route requests to the right locations at any given moment, thereby preventing costly bottlenecks and unforeseen incidents. Load balancers should

ultimately deliver the performance and security necessary for sustaining complex IT environments, as well as the intricate workflows occurring within them.[1]

4. Auto Scaling group

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 to use of 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.

An additional feature to the architecture is the Auto Scaling group with adds another EC2 instance while loading increases. In provided architecture, the Auto Scaling group has been attached to web services.

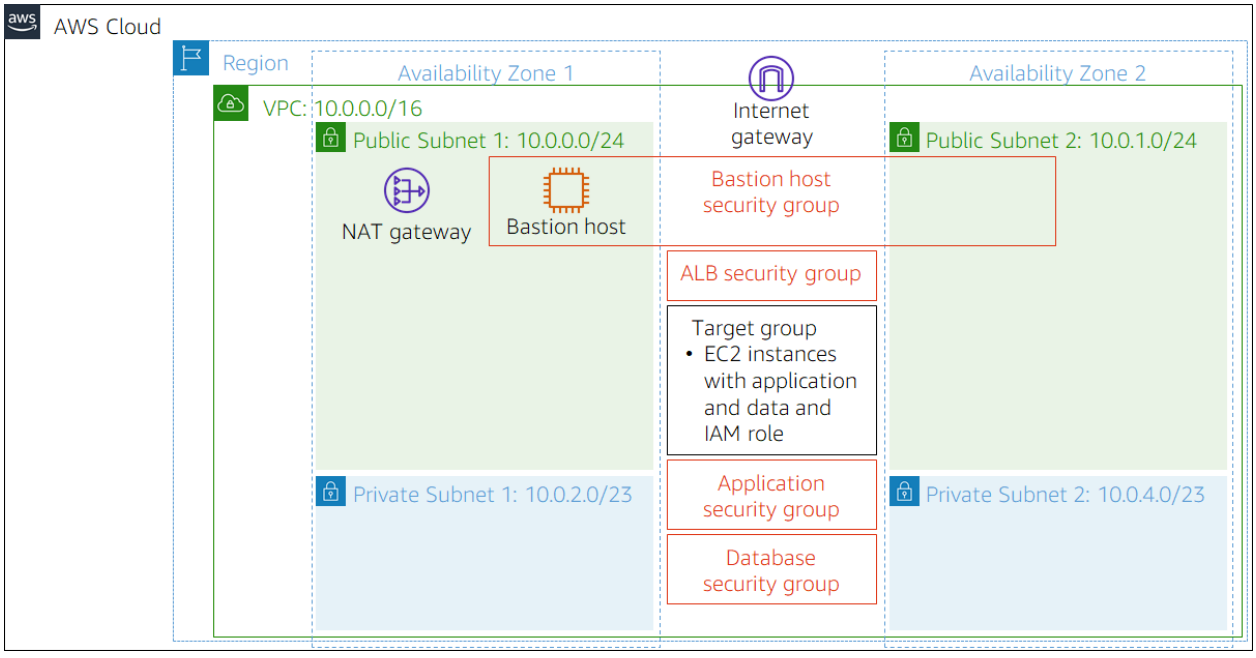
5. Web Services

Web services used in the app have been created by launch templates provided by AWS. Which contains PHP codes and SQL assets to run the application.

6. Application Architecture

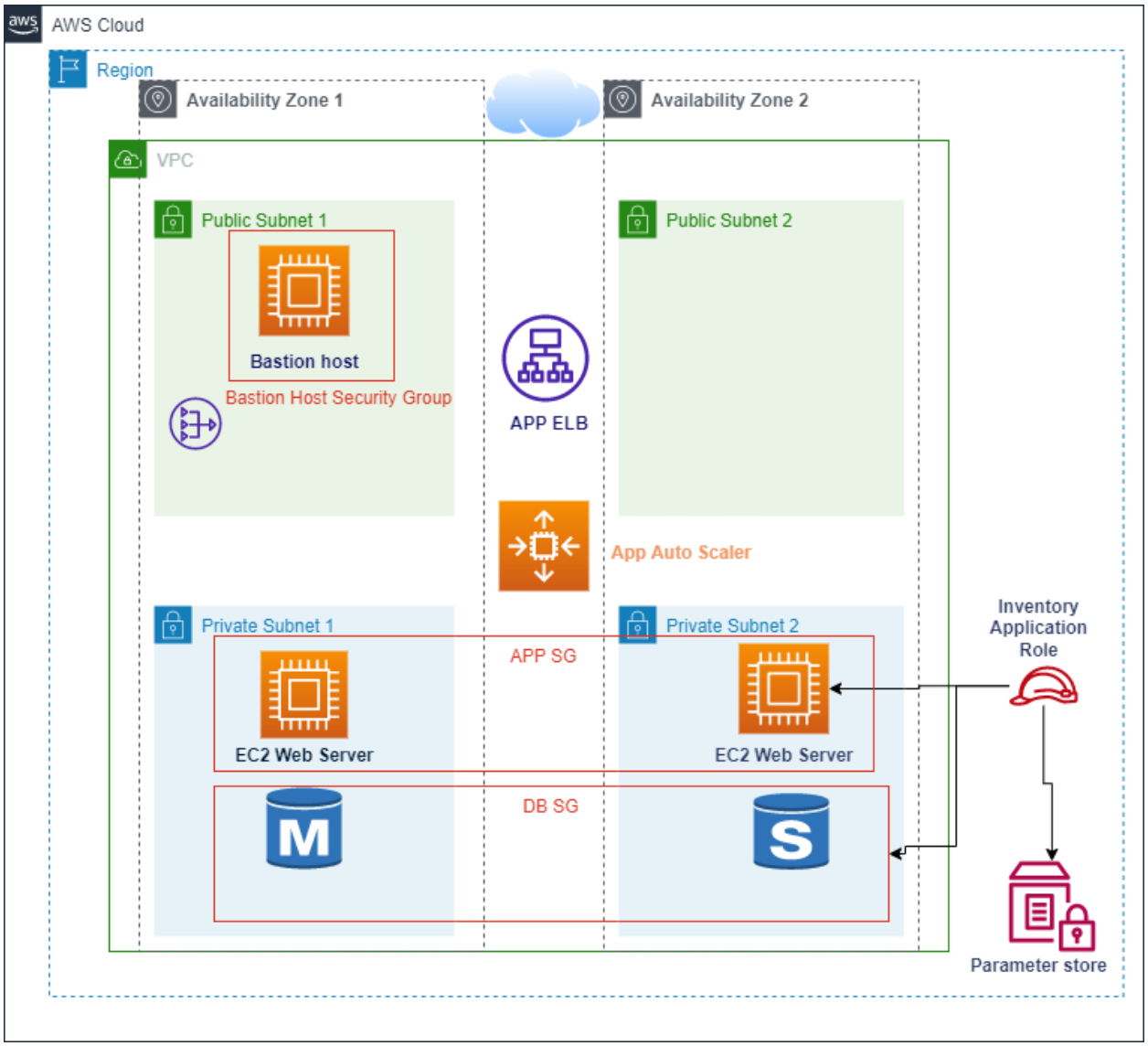
6.1 Before

we have a vpc with four subnets two public and two private subnets and we have a bastion host which basically allows us to create an ssh session to the ec2 instances that we will have here.



6.2 After

And this is what we create. Via launch templates, we get the project assets here in the template, and then we use that to deploy the PHP application. After that, we access the templates to import the data to the rds here in our private subnet and then we create an application load balancer, auto scaler and then we launch the private ec2 in a sense using the template that was given to the lab.



Reference:

[1]https://www.citrix.com/pl-pl/solutions/app-delivery-and-security/load-balancing/what-is-load-balancing.html
[2] https://docs.aws.amazon.com/autoscaling/ec2/userguide/AutoScalingGroup.html
[3] https://aws.amazon.com/rds/
[4] https://aws.amazon.com/rds/sqlserver/