Documentación E3

Iaac

Se utilizó **Terraform** de Hashicorp para levantar la infraestructura, debido a la facilidad que proporciona para conectarse a AWS, y la buena documentación disponible.

Las estructuras levantadas utilizando esta herramienta fueron:

- VPC y 2 subnets

```
resource "aws_default_vpc" "default" { … }
resource "aws_default_subnet" "default_az1" { … }
resource "aws_default_subnet" "default_az2" { … }
```

Primero creamos 2 subnets con las regiones donde se replicarán las instancias.

- AMI del backend

```
resource "aws_ami_copy" "copy_ami" { … }
```

Copiamos la AMI que hemos estado utilizando hasta ahora.

- Elastic Load Balancer

```
resource "aws_lb" "test" { … }
resource "aws_alb_target_group" "backend" { … }
resource "aws_lb_listener" "backend" { … }
```

Primero creamos el Balanceador de Carga con *“aws_lb”*, luego le creamos un grupo (el cual se le va a asignar al ASG) con *“aws_alb_target_group”*, y este grupo lo agregamos como listener al Balanceador utilizando *“aws_lb_listener”.*

- Auto Scaling Group

```
resource "aws_autoscaling_group" "bar" { … }
resource "aws_autoscaling_attachment" "asg_attachment_bar" { … }
resource "aws_launch_template" "backend" { … }
```

Creamos un template para que el ASG pueda replicar instancias con *“aws_launch_template”*, utilizando la ami que copiamos. Luego creamos el auto scaling group con *“aws_autoscaling_group”* , finalmente le asignamos el grupo que creamos con el balanceador de carga, utilizando *“aws_autoscaling_attachement”*.

- ElastiCache con cluster de Redis

```
resource "aws_elasticache_cluster" "example" { … }
```

Creamos un cluster con redis.

Todo esto se realizó utilizando la región us-west-2, N. California.

Como espacio de mejora, lo que podría hacerse sería replicar este código, cambiando la variables de entorno, para levantar la infraestructura en otra región de AWS. Además, se puede utilizar la función `import` de Terraform, para así incorporar la RDS actual de la free tier, y no ser redundantes al crear una nueva, o incluso incluir otros recursos como el Elastic Load Balancer, ya que no es algo que requiere muchos ajustes especiales, por lo que replicarlos no requiere una configuración extra más que relacionar al balanceador de carga a un grupo de autoscaling. Otra posible reutilización de infraestructura sería el cluster de Redis, pues el cache no es algo que se debe mantener al largo plazo, por lo que se puede importar la infraestructura y replicarlo.

Monitoreo

Para el monitoreo se utilizó CloudWatch de Amazon en donde se creó un panel de monitoreo central con los siguientes widgets:

- Utilización de CPU
- Tráfico entrante
- Tráfico saliente
- StatusCheckFailed de instancias
- Operaciones de lectura a disco
- Operaciones de escritura a disco
- Tiempo de respuesta API
- Status code de respuestas API
- Uso de la CPU en la BD
- Tiempo de R/W en la BD
- Throughput de R/W en la BD
- Espacio libre en la BD
- Uso de CPU Redis
- Caché hit y misses, memoria liberable en Redis

Todo lo anterior monitorea el grupo de autoscaling utilizado en las entregas anteriores y las bases de datos respectivas según corresponda. La API monitoreada (para el tiempo de respuesta) es la que expone nuestro backend del chat.

Por último, el panel de monitoreo puede ser visualizado por los administradores del chat ingresando al link de “Monitoreo” ubicado en el menú exclusivo de administrador dentro de la aplicación del chat y luego autenticándose con la siguiente cuenta:

Username: r.espinoza1.al
Password: Grupo20!

El sistema, para esta entrega, incluye CloudWatch para el monitoreo y el código para IaaC como archivo separado. Se presenta a continuación y se puede ver a partir del link (https://app.diagrams.net/#G1fETqCjtk_2rLIWYi7-_HsT_yHOJYo2zH):