PRÁCTICA 3: CLOUD FORMATION

RAÚL MATEUS SÁNCHEZ

COMPUTACIÓN EN LA NUBE
Escuela de Ingeniería Informática

Índice

Introducción

Uno de los servicios más potentes de AWS es CloudFormation. Esta herramienta nos permite definir arquitecturas en la nube de AWS de forma que ayude a la automatización de procesos y de creación de estas, haciendo al usuario mucho más fácil el despliegue de una arquitectura determinada sin prácticamente tener que tocar la interfaz gráfica o algún servicio de EC2 que hemos visto hasta el momento.

Objetivos

El objetivo de esta práctica es explorar y experimentar con CloudFormation, de forma que se aprenda a definir arquitecturas en la nube de AWS usando esta herramienta en forma de stacks, usando para ello ficheros JSON o YAML.

Desarrollo de la práctica

1. Describe un Stack de Cloud Formation que incluya los siguientes elementos:

    1. Instancia en EC2 que se pueda acceder por SSH desde el exterior

```
"sshGateInstance": {
            "Type": "AWS::EC2::Instance",
            "Properties": {
                "ImageId": "ami-09d3b3274b6c5d4aa",
                "InstanceType": "t2.micro",
                "SecurityGroupIds": [
                    {
                        "Ref": "sshGateSecurityGroup"
                    }
                ],
                "KeyName": {
                    "Ref": "sshGateKey"
                }
            }
        },
```

Para crear la instancia en CloudFormation, dentro del fichero JSON que servirá de template al stack que crearemos, este será el bloque de código que cree la instancia accesible por SSH desde el exterior. Para ello, se declara el tipo (una instancia de EC2), así como el ID de la imagen a usar y el tipo de instancia, los cuales han sido escogidos a raíz de las instancias creadas anteriormente.

Por otro lado, se referencia el grupo de seguridad de la instancia, el cual será declarado en siguientes apartados, así como el par de claves creado para entrar a través de PuTTY en el caso de Windows. Es visible como, al fin y al cabo, son los mismos parámetros que configuramos cuando la creamos desde el panel de control gráfico de EC2.

2. Instancia en EC2 que tenga una servidor web en la que muestre alguna característica de la máquina para poder diferenciarla (e.g. nombre, IP, dominio)

```
"webServerInstance": {
            "Type": "AWS::EC2::Instance",
            "Properties": {
                "ImageId": "ami-09d3b3274b6c5d4aa",
                "InstanceType": "t2.micro",
                "SecurityGroupIds": [
                    {
                        "Ref": "webServerSecurityGroup"
                    }
                ],
                "KeyName": {
                    "Ref": "webServerKey"
                },
                "UserData": {
                    "Fn::Base64": {
                        "Fn::Sub": "#!/bin/bash\nyum update -y\nyum install -y
httpd\nsystemctl enable httpd\nsystemctl start httpd\necho \"<html><h1>El
servidor de Raul: Generado por CloudFormation</h1></html>\" >
/var/www/html/index.html\n"
                    }
                }
            }
        }
```

La creación de la instancia que tiene un servidor web no es muy distinta de la instancia creada anteriormente, pero con una diferencia. Además de que el grupo de seguridad y el par de claves es distinto, la principal diferencia es en el campo UserData. Este campo nos permite automatizar la instalación del servicio necesario para permitir que la instancia funcione como servidor web, así como la configuración del mismo, para que cuando accedamos a la dirección IP pública de la misma, aparezca algo así y sea diferenciable:

Figura 1: Servidor Web creado por CloudFormation

3. Grupos de seguridad y pares de claves (key pairs) que crea necesario

```
"AWSTemplateFormatVersion": "2010-09-09",
    "Description": "First Activity - Practica 3 - Computacion en la Nube",
    "Resources": {
        "sshGateSecurityGroup": {
            "Type" : "AWS::EC2::SecurityGroup",
            "Properties": {
                "GroupDescription" : "Security Group for SSH access",
                "GroupName" : "sshGateSecurityGroup",
                "SecurityGroupIngress" : [{
                    "IpProtocol": "tcp",
                    "FromPort": 22,
                    "ToPort": 22,
                    "CidrIp": "0.0.0.0/0"
                }]
            }
        },

        "webServerSecurityGroup": {
            "Type" : "AWS::EC2::SecurityGroup",
            "Properties": {
                "GroupDescription" : "Security Group for Web Servers",
                "GroupName" : "webServerSecurityGroup",
                "SecurityGroupIngress" : [{
                    "IpProtocol": "tcp",
                    "FromPort": 22,
                    "ToPort": 22,
                    "SourceSecurityGroupName" : "sshGateSecurityGroup"
                },
                {
                    "IpProtocol" : "tcp",
                    "FromPort" : 80,
                    "ToPort" : 80,
                    "CidrIp" : "0.0.0.0/0"
                }]
            }
        },

        "sshGateKey": {
            "Type": "AWS::EC2::KeyPair",
            "Properties": {
                "KeyName": "sshGateKey",
                "KeyType": "rsa"
            }
        },

        "webServerKey": {
```

```
            "Type": "AWS::EC2::KeyPair",
            "Properties": {
                "KeyName": "webServerKey",
                "KeyType": "rsa"
            }
        },
```

En este caso, se ha considerado necesario crear dos grupos de seguridad, uno para el acceso por SSH y otra para gestionar las peticiones HTTP a un servidor web. Para ello, se declara cada uno de ellos, donde solo tienen declarado reglas "inbound", es decir, entrantes. El grupo de seguridad para el acceso SSH solo tendrá una regla que permita el acceso por el puerto 22 al exterior, restrigiendo cualquier otra conexión entrante.

Para el grupo de seguridad que define las reglas para los servidores web, se añade otra regla entrante donde se permiten las peticiones HTTP del exterior. Además, se modifica el acceso SSH para que solo desde las instancias que tengan definido el grupo de seguridad *"sshGateSecurityGroup"* puedan acceder con el par de claves correspondiente.

Por otra parte, se crea dos par de claves, uno para cada instancia que pueden ser usados para entrar a las máquinas. Al crear el par de claves desde *CloudFormation*, la clave privada se guarda como parámetro en *AWS Systems Manager*, pudiendo ser descargada desde el CLI del *Learner Lab* con el comando:

```
aws ssm get-parameter --name /ec2/keypair/{key_pair_id} --region {region}
--with-decryption
```

```
ddd_v1_w_9wL_1447466@runweb67483:~$ aws ssm get-parameter --name /ec2/keypair/key-09c42
9ba6824e383f --region us-east-1 --with-decryption
{
    "Parameter": {
        "Name": "/ec2/keypair/key-09c429ba6824e383f",
        "Type": "SecureString",
        "Value": "-----BEGIN RSA PRIVATE KEY-----\nMIIEogIBAAKCAQEAuSyPQJyZNkfqQrYIgcGw
jXHNhXVEISQi+KCmU+19L32JpW5b\n4GFjuC5PiIFzTK4yTD4sJv04Ugv0OIoQ3bCuPdgitJC6zPQ/V3ahuHSfb
zEDIvPi\nhbM88CR8VzmLdFtrJjmty+Mj2uAW+mhRJseqm/62TJahRAGZ7P+FeW4ViwE93WzR\n72JTVcddtif1
YHeKWA2CFvtg8KHdpnVXsls2fiFCoujL7t8Cm1zTJy1TH42AKGot\nSu0UA3SrNqjXUOzDS5FyoDQT3SGGgOZY4
rfjIqhUQKjgE9yFZvHOrDQtwgZkpQie\nD9Iq72HnExGtYiFkdf6nzdqf+Gl601741E0QVwIDAQABAoIBAB610i
hAEHTdJm/b\n6rtCWTqMFi69DQiGCrq7QfqGrblTk6F5FQm4fSZNn+ketvWPqk3UvdPViSKqUt2R\noEx0AeHF1
7OkkSZu15a02QtTZCG3ujJwyi+sUz3sfyakJ4mro7CaPkWjLQ4evhDX\nfiX1VUk7zBYVoTc7IMEae5Nw+2erCw
DqMMrXExqck+kWDkoNQKm428vLRLXRCY16\ninVshNM7UOfGdv5/ja7FuEkFCRw0vb5XXYtX2ETHKb6ADdup17O
Sp/AIy6wmdcWl\n41p32eckl Z0qEMg/xSo9H7Uhxb/20ktbf7z6igb0VtPGvCyzGek4m4o1hI74p/Cs\nYuUb2k
ECgYEA/UILB7nrJ1IhoPzyUpRPfZf5xfzLYYqtKH0S0mY+nhp80hZauxmc\nByiysb61bkTf5mZIJx/N18tjsDy
XO0Xsb3wHcKTGL+MVtB87eKFHKniuST7ZB9Rp\nv/FI6E94nA1vxV+8ylMGM/zm6BoH3gXYWH7tcWz2ngvRDHwY
SBBSjzcCgYEAuy3O\nzRT1AH7KieFGK2t7MwxmmzrsydYTP1wdxTKyw8e3d6jzj+FI0bM0JqqlNRIgPEAF\ntz+
a6cwoqaHFwn0N1EomFTuAFJ2Mru5t/apRaQ3gxq9iFoPhiEqrF3DuF/cKx9M7\nmL+7/tNtvZwhk4xx98W8DLGP
II4wZgTVCGU61+ECgYA1uOK6GtglGSkFC2Mj3U4X\naDuVohBKeq5Mnmt1sjaqNxgvQIqykbY1Vp7BUaEO6RFNw
Hihw/9KqdxVsCSscCK9\na2PwoyNfWR5rfQyShDOicMMXynsjTA26p6Y9kU4WXEtPZQWZLY9tS8J3jb4JfNft\n
vc8EdobB8+O+cDx+XOgGdQKBgA4YQfVReHkUEx5U11rXqhxK98MCqef89dAUmK1L\nxwhVFqytDUItXbVkwC2VN
gGMDt9altOYt4qZdApqAyFFF2uuNbXtmBR1VK2riJ7M\nX3sAr4dml4VIgwgnT0DhoRwu3yHy9w7miTqEI0Re6G
xa7A4TVrjB+hwieOSFwqDy\nNn8hAoGAN+Q2VNvw9N0lT6kS+pMT/fSpeI5BrO+XNbnPU3mhiSxyf3sOVXcss1t
E\nyKjRwu6g7dsVSBUwjucoOXBWrJ/3BbuOdw7bxFhaK2+7KqSE5henmtoVN8S7jRiK\nBo3u7UdOmNnzXVqHwc
4tnm2uaI1OGQ0Zlf1MJ3d/p3Aiqm8OSkw=\n-----END RSA PRIVATE KEY-----",
        "Version": 1,
        "LastModifiedDate": "2022-11-18T09:24:36.630000-08:00",
        "ARN": "arn:aws:ssm:us-east-1:340954026236:parameter/ec2/keypair/key-09c429ba68
24e383f",
        "DataType": "text"
    }
}
ddd_v1_w_9wL_1447466@runweb67483:~$
```

Figura 2: Ejemplo de obtención de clave privada

2. Describe un Stack de Cloud Formation que incluya los siguientes elementos:
    1. Dos instancias en EC2 con un servidor web que muestre una página similar pero que se pueda reconocer que es un servidor distinto. La página tiene que contener alguna característica de la máquina para poder diferenciarla (e.g. nombre, IP, dominio) y ser accesible desde fuera

```
"webServersSecurityGroup": {
            "Type" : "AWS::EC2::SecurityGroup",
            "Properties": {
                "GroupDescription" : "Security Group for Web Servers",
                "GroupName" : "webServersSecurityGroup",
                "SecurityGroupIngress" : [{
                    "IpProtocol" : "tcp",
                    "FromPort" : 80,
                    "ToPort" : 80,
                    "SourceSecurityGroupId": {
                        "Fn::GetAtt": [
                            "ELBSecurityGroup",
                            "GroupId"
                        ]
                    }
                }]
            }
        },

        "webServersKey": {
            "Type": "AWS::EC2::KeyPair",
            "Properties": {
                "KeyName": "webServersKey",
                "KeyType": "rsa"
            }
        },

        "webServer1Instance": {
            "Type": "AWS::EC2::Instance",
            "Properties": {
                "ImageId": "ami-09d3b3274b6c5d4aa",
                "InstanceType": "t2.micro",
                "SecurityGroupIds": [
                    {
                        "Ref": "webServersSecurityGroup"
                    }
                ],
                "KeyName": {
                    "Ref": "webServersKey"
```

```
                },
                "UserData": {
                    "Fn::Base64": {
                        "Fn::Sub": "#!/bin/bash\nyum update -y\nyum install -y
httpd\nsystemctl enable httpd\nsystemctl start httpd\necho Servidor:
$(hostname -f) - Generado por CloudFormation> /var/www/html/index.html\n"
                    }
                }
            }
        },

        "webServer2Instance": {
            "Type": "AWS::EC2::Instance",
            "Properties": {
                "ImageId": "ami-09d3b3274b6c5d4aa",
                "InstanceType": "t2.micro",
                "SecurityGroupIds": [
                    {
                        "Ref": "webServersSecurityGroup"
                    }
                ],
                "KeyName": {
                    "Ref": "webServersKey"
                },
                "UserData": {
                    "Fn::Base64": {
                        "Fn::Sub": "#!/bin/bash\nyum update -y\nyum install -y
httpd\nsystemctl enable httpd\nsystemctl start httpd\necho Servidor:
$(hostname -f) - Generado por CloudFormation> /var/www/html/index.html\n"
                    }
                }
            }
        }
```

Para crear dos web servers, se sigue el mismo procedimiento que en apartados anteriores. Se genera un grupo de seguridad para los web servers con una gran diferencia. En este caso, solo aceptará peticiones HTTP si provienen del grupo de seguridad asociado al Load Balancer que se creará más adelante, para que estas no puedan ser accedidas individualmente desde el exterior. Por otra parte, se genera un par de claves para estos servidores y dos instancias donde se configure un pequeño script para que se configure automáticamente un servidor web. Para diferenciarlas, se muestra el hostname de cada una.

2. Un load balancer que distribuya las peticiones entre los dos servidores a partes iguales

```
"ELBSecurityGroup": {
            "Type" : "AWS::EC2::SecurityGroup",
            "Properties": {
                "GroupDescription" : "Security Group for Load Balancers",
                "GroupName" : "ELBSecurityGroup",
                "SecurityGroupIngress" : [{
                    "IpProtocol" : "tcp",
                    "FromPort" : 80,
                    "ToPort" : 80,
                    "CidrIp" : "0.0.0.0/0"
                }]
            }
        },

"AppLoadBalancer": {
            "Type" : "AWS::ElasticLoadBalancingV2::LoadBalancer",
            "Properties" : {
                "IpAddressType" : "ipv4",
                "Name" : "AppLoadBalancer",
                "Scheme" : "internet-facing",
                "Subnets" : [ "subnet-00db343d4ea75c3f8", "subnet-
067218daac8437c2d"],
                "SecurityGroups": [
                    {
                        "Fn::GetAtt": [
                            "ELBSecurityGroup",
                            "GroupId"
                        ]
                    }
                ]
            }
        },

        "ELBListener": {
            "Type" : "AWS::ElasticLoadBalancingV2::Listener",
            "Properties" : {
                "DefaultActions" : [{
                    "Type": "forward",
                    "TargetGroupArn": { "Ref": "ELBTargetGroup"}
                }],
                "LoadBalancerArn" : { "Ref": "AppLoadBalancer"},
                "Port" : 80,
                "Protocol" : "HTTP"
            }
        },
```

```
    "ELBTargetGroup": {
        "Type" : "AWS::ElasticLoadBalancingV2::TargetGroup",
        "Properties" : {
            "HealthCheckIntervalSeconds" : 30,
            "HealthCheckPath" : "/",
            "HealthCheckTimeoutSeconds" : 5,
            "HealthyThresholdCount" : 2,
            "Matcher" : {"HttpCode": "200"},
            "Name" : "ELBTargetGroup",
            "Port" : 80,
            "Protocol" : "HTTP",
            "Targets": [
                {
                    "Id": { "Ref": "webServer1Instance"},
                    "Port": 80
                },

                {
                    "Id": { "Ref": "webServer2Instance"},
                    "Port": 80
                }
            ],
            "UnhealthyThresholdCount" : 5,
            "VpcId" : "vpc-088febf7f494a9d6b"
        }
    }
}
```

Para crear el *Load Balancer*, escogemos la versión 2 del mismo para generarlo. En primer lugar, se crea un grupo de seguridad asociado al *Load Balancer* que no es muy distinto de los que se han creado anteriormente. Este si aceptará peticiones HTTP del exterior, sea de quien sea, ya que será el *Load Balancer* el encargado de distribuir las peticiones.

Para declarar el *Load Balancer*, necesitamos tres cosas: la declaración del tipo de *Load Balancer*, un *Listener* y un *Target Group*. En la declaración, se especifica que aceptará IPs de tipo IPv4 y su esquema será de internet-facing para enrutar las peticiones a los servidores de manera que balancee la carga. Se especifican dos subnets asociadas a dos zonas de disponibilidad (us-east-1b y us-east-1c), y se asocia el grupo de seguridad a través de su ID.

3. Un “Auto-Scaling Group”(ASG) que tenga como mínimo una instancia y como máximo 2. El ASG debe añadirse al “load balancer” previamente descrito

```
"webServerLaunchTemplate": {
            "Type": "AWS::EC2::LaunchTemplate",
            "Properties": {
                "LaunchTemplateName": "Launch-Template",
                "LaunchTemplateData": {
                    "SecurityGroupIds":[{
                        "Fn::GetAtt": [
                            "webServersSecurityGroup",
                            "GroupId"
                        ]
                    }],
                    "ImageId": "ami-09d3b3274b6c5d4aa",
                    "InstanceType": "t2.micro",
                    "KeyName": {
                        "Ref": "webServersKey"
                    },
                    "UserData": {
                        "Fn::Base64": {
                            "Fn::Sub": "#!/bin/bash\nyum update -y\nyum
install -y httpd\nsystemctl enable httpd\nsystemctl start httpd\necho
Servidor: $(hostname -f) - Generado por CloudFormation y Launch Template>
/var/www/html/index.html\n"
                        }
                    }
                }
            }
        },

        "autoScalingGroup": {
            "Type" : "AWS::AutoScaling::AutoScalingGroup",
            "Properties" : {
                "AutoScalingGroupName" : "Auto-Scaling-Group",
                "AvailabilityZones" : [
                    "us-east-1b",
                    "us-east-1c"
                ],
                "DesiredCapacity" : "1",
                "HealthCheckType" : "ELB",
                "LaunchTemplate": {
                    "LaunchTemplateId": {
                        "Ref": "webServerLaunchTemplate"
                    },
                    "Version": {
                        "Fn::GetAtt": [
                            "webServerLaunchTemplate",
```

```
                        "LatestVersionNumber"
                    ]
                }
            },
            "MaxSize" : "2",
            "MinSize" : "1",
            "TargetGroupARNs" : [ { "Ref": "ELBTargetGroup"}]
        }
    }
```

En último lugar, se describe la creación de un *"Auto-Scaling Group"* el cual tiene asociado un template a raíz del cual genera las instancias. Para la declaración del template, se usa de base la misma declaración de una instancia, pues, al fin y al cabo, emplean la misma imagen, tipo de instancia, par de claves, grupo de seguridad y script de creación de servidor web que las anteriores.

Para declarar un ASG, se necesita especificar las zonas de disponibilidad, las cuales han sido seleccionadas para que concuerde con lo detallado en la creación del *Load Balancer*. Por otra parte, se especifica que como máximo tenga dos instancias y como mínimo una, siendo una instancia el tamaño deseado. Además, se debe asociar el template para generar las instancias, indicando en este caso la referencia del template que se creará junto con él, así como su versión, la cual se obtiene empleando una función *Get Attribute*.

Por último, se asocia el ASG con el Load Balancer mediante el *Target Group*, de manera que las instancias creadas por el ASG sean añadidas al *Target Group* al cual el *Load Balancer* redirigirá las peticiones HTTP.

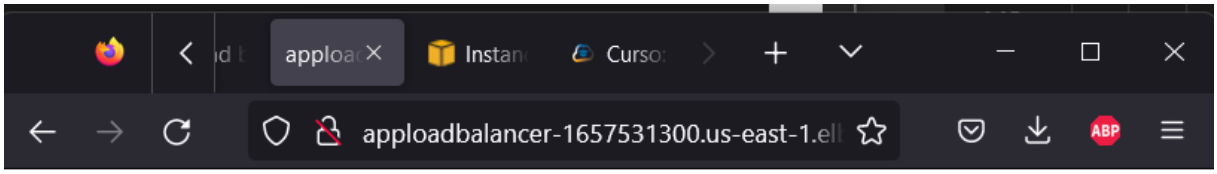

Figura 3: Load Balancer redirigiendo a Web Server 1

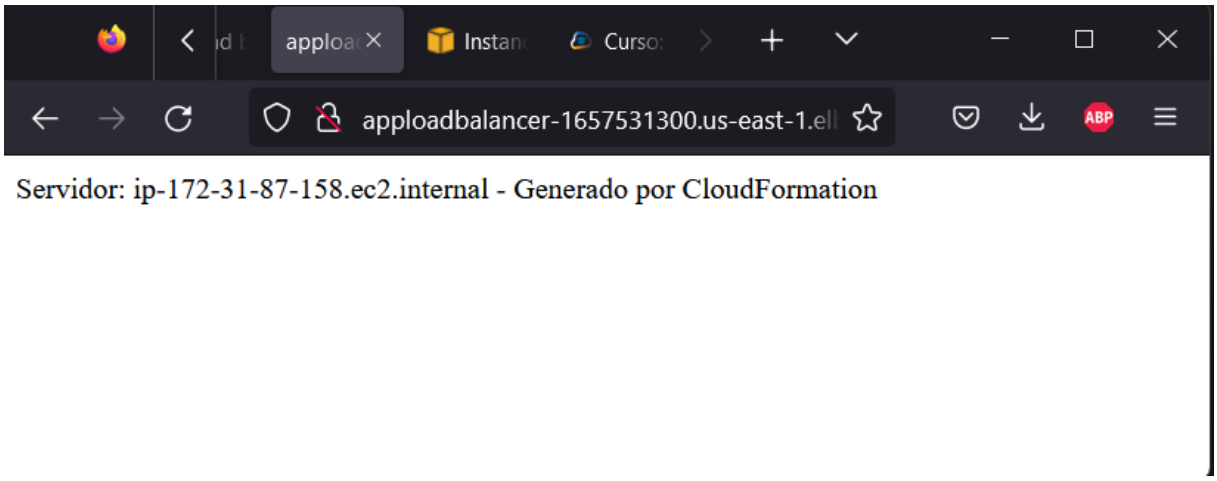

Figura 4: Load Balancer redirigiendo a Web Server 2

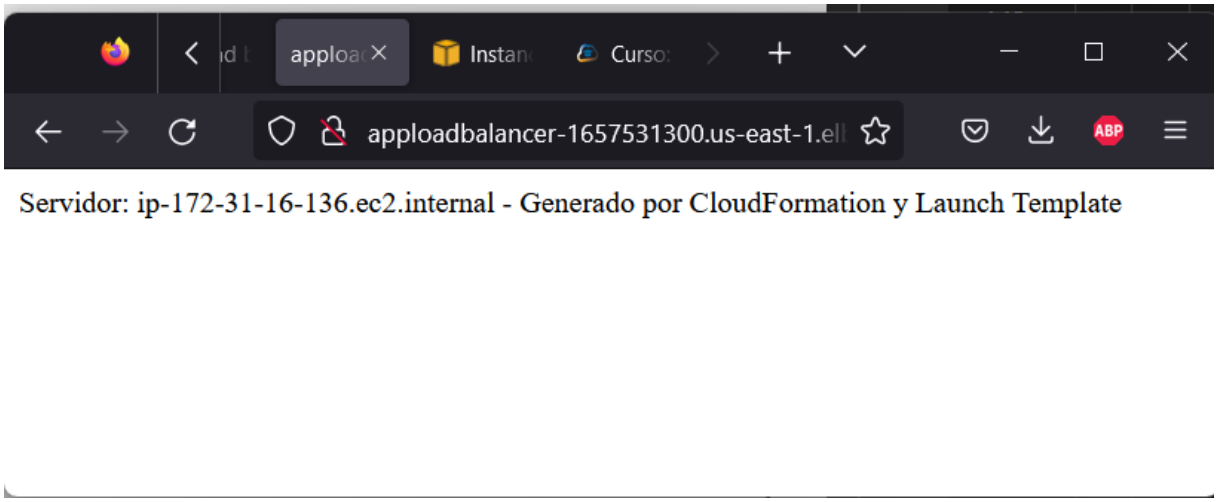

Figura 5: Load Balancer redirigiendo a Web Server creado por ASG

Diagrama de arquitectura desplegada

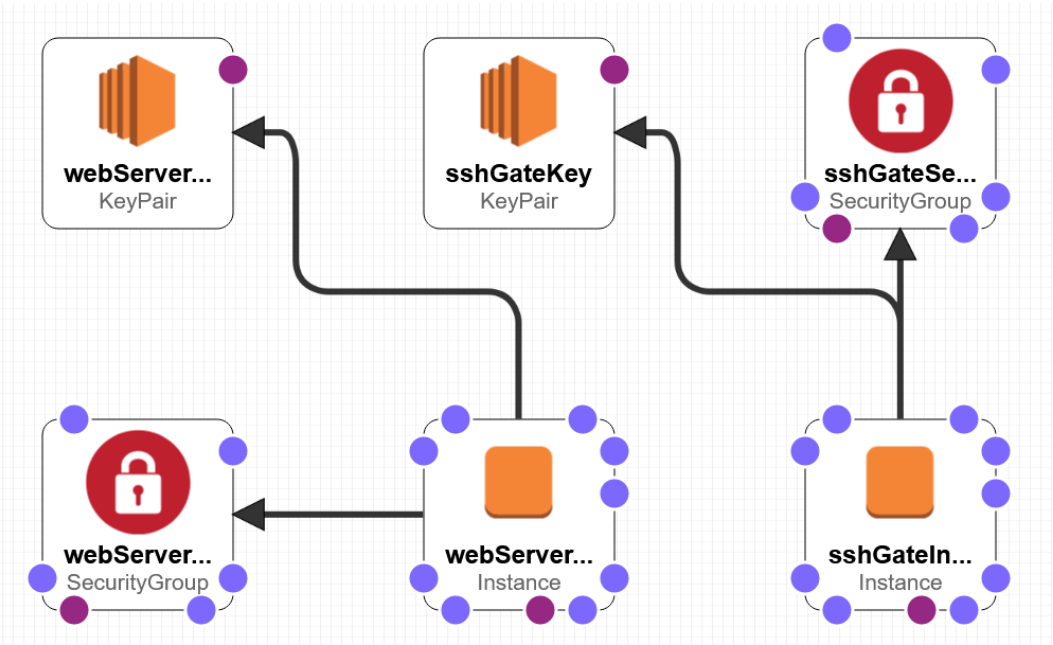

Figura 6: Arquitectura desplegada en el apartado 1 según CloudFormation

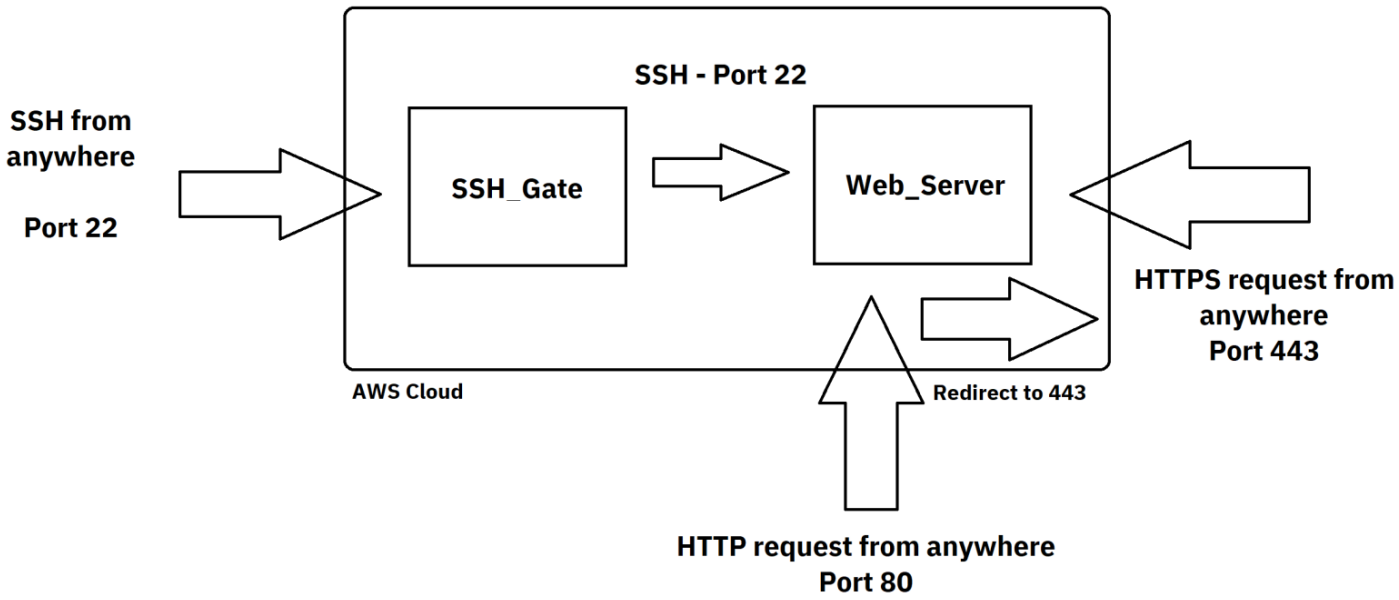

Figura 7: Representación gráfica de la arquitectura desplegada en el apartado 1

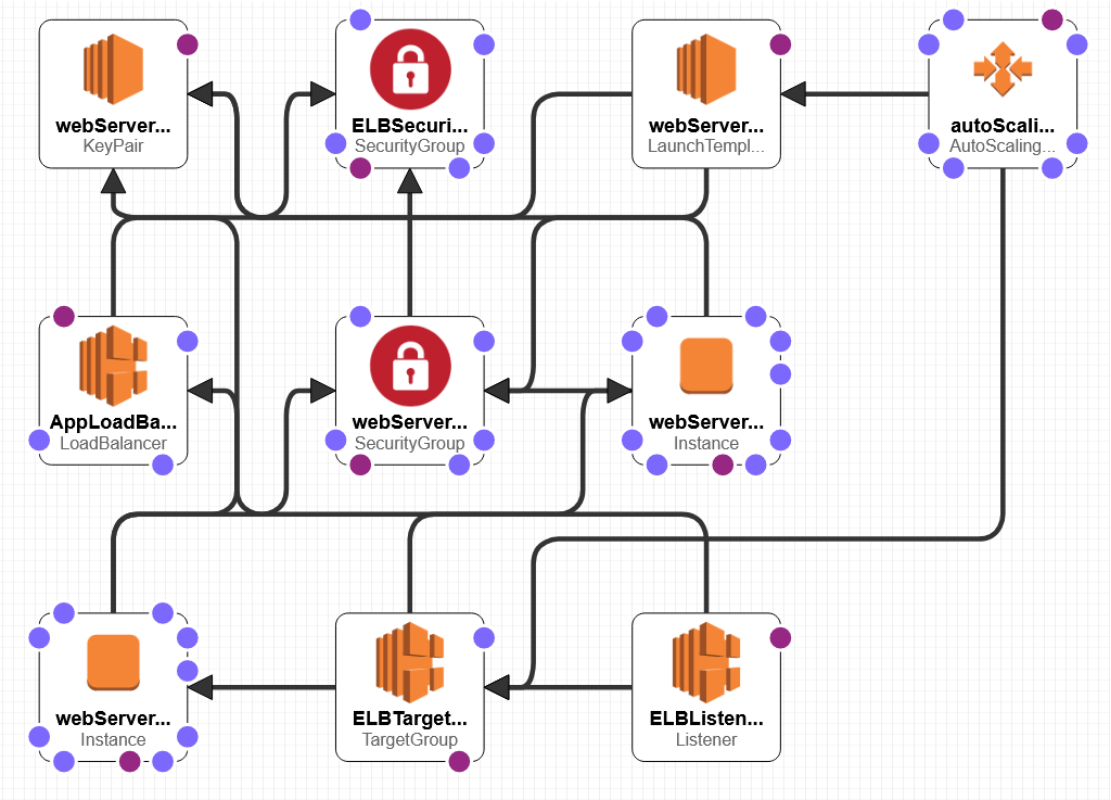

Figura 8: Arquitectura desplegada en el apartado 2 según CloudFormation

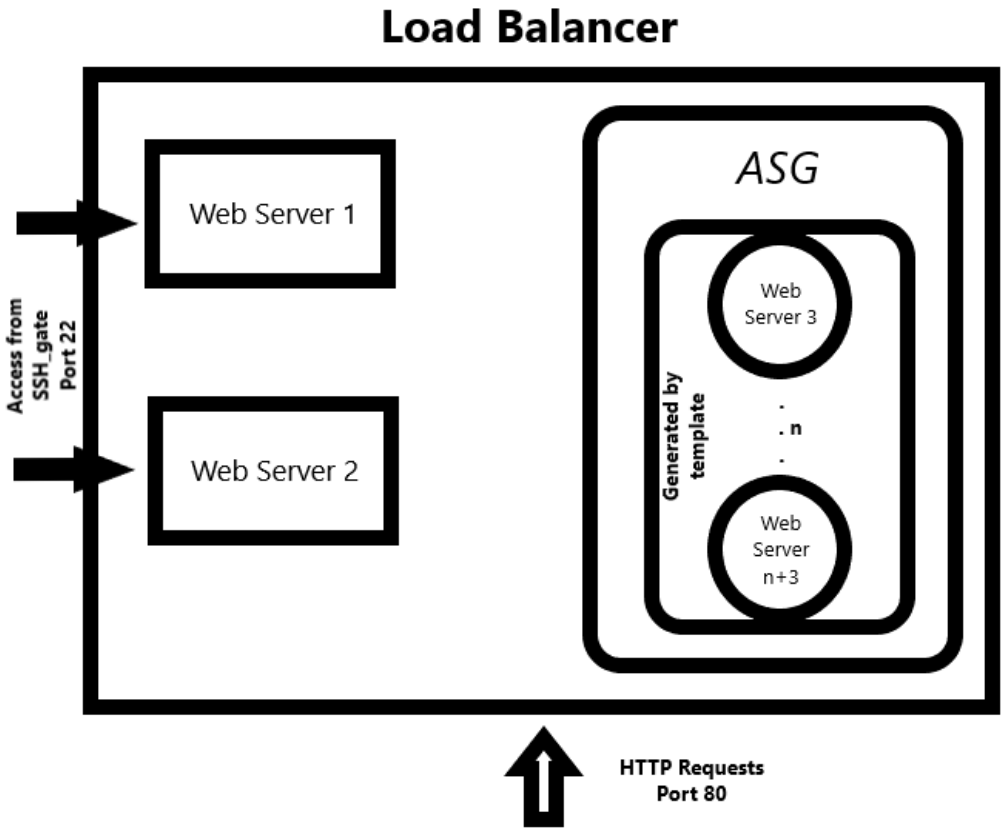

Figura 9: Representación gráfica de arquitectura desplegada en apartado 2

Presupuesto y estimación de gasto de los recursos desplegados

Para esta práctica, el presupuesto total restante de los 100$ iniciales es de 95,8$, quitando así el gasto de las prácticas anteriores. En este caso, se debe dividir la estimación de gasto según cada apartado, pues funcionarán como arquitecturas independientes.

La primera arquitectura desplegada consta de dos instancias, una destinada a actuar como "pasarela" mediante SSH a la segunda instancia, la cual funcionará como servidor web. En este punto, empleando una AMI de Amazon Linux y un tipo de instancia t2.micro, el coste de cada instancia por hora es de 0.0116$. Además, el uso de volúmenes SSD de uso general (gp2) cuesta 0,10$ por GB-mes.

Amazon calcula el coste de Amazon EBS como el número de instancias, por su capacidad y por su precio GB-mes. Es por ello por lo que, mensualmente, el coste del almacenamiento de EBS será de:

$$8\ GB * 2\ instancias * 0.10\$ = 1{,}60\$$$

Añadiendo el coste de las instancias, el precio mensual será:

$$1{,}60\$ + (0.0116\$ * 2) * (30\ días * 24\ horas) = 18{,}30\$$$

Ahora bien, siendo realistas, una vez se consiga desplegar la arquitectura en CloudFormation y se pruebe su correcto funcionamiento, el stack será eliminado y no volverá a ser creado hasta su defensa, por lo que realmente el uso de estas instancias se reducirá a dos horas en dos días diferentes, siendo por tanto el coste real:

$$\left(\frac{1{,}60\$}{30} * 2\ dias\right) + \left((0.0116\$ * 2) * 2\ horas\right) = 0{,}15\$$$

La segunda arquitectura desplegada consiste en un Elastic Load Balancer, en su versión Application Load Balancer, como dos instancias como servidor web, y una tercera creada por un Auto-Scaling Group. Por hora, nos encontramos con un precio de 0.0225$ para el ELB, mientras que por cada instancia en ejecución el precio será de 0.0116$ cada una por hora como hemos visto anteriormente. Además, el uso de volúmenes SSD de uso general (gp2) cuesta 0,10$ por GB-mes al igual que en la primera arquitectura.

Por todo ello, mensualmente el coste de esta segunda arquitectura es:

$$EBS = 8GB * 3\ instancias * 0{,}10\$\frac{GB}{mes} = 2{,}4\$/mes$$

$$2{,}4\$ + \left(0{,}0225\$ + (0.0116\$ * 3)\right) * 720\ horas = 43{,}66\$$$

Ahora bien, al igual que antes, en un caso realista, esta arquitectura estará desplegada a lo sumo 2 horas, por lo que una estimación realista será:

$$\frac{2{,}4}{30} * 2 + \left(0{,}0225\$ + (0.0116\$ * 3)\right) * 2\ horas = 0{,}27\$$$

Fuentes

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-instance.html

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-whatis-concepts.html

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-security-group.html

https://docs.aws.amazon.com/es_es/AWSCloudFormation/latest/UserGuide/quickref-general.html

https://cloudkatha.com/how-to-execute-ec2-user-data-script-using-cloudformation/