



INFRAESTRUCTURAS DE CLOUD PÚBLICO

Despliegue de API REST en Amazon

Universidad Politécnica de Valencia
Mater Universitario en computación paralela y distribuida

INFRAESTRUCTURAS DE CLOUD PÚBLICO
dooros@posgrado.upv.es

Contenido

Introducción	2
Esquema General	3
Objetivos	3
Desarrollo	4
Requisitos previos	4
Otorgar permisos a nuestro usuario Linux.....	4
Instalación de CLI Amazon	4
Instalación de Ansible	4
Scripts de configuración	5
Ejecución	9
Creación de Base de datos	9
Creación de infraestructura de alta disponibilidad.....	11
Conclusiones	15
Referencias.....	16

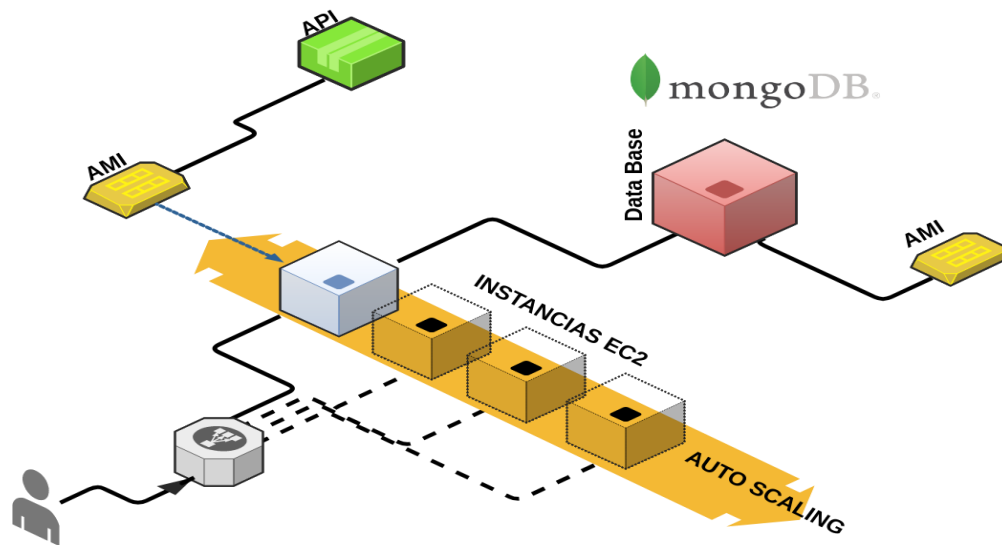
Introducción

El uso de tecnologías como es el Cloud Computing es cada vez más aceptado por las empresas, quienes ahora están más dispuestos a pagar por infraestructuras albergadas en la nube y con ello reducen los costos de implementación y mantenimiento a gran escala por no decir casi cero. Este documento presenta una forma de desplegar una API REST en la plataforma de infraestructura pública como lo es Amazon Web Service con una configuración de auto escalado a una aplicación Stateless que nos va garantizar alta disponibilidad de nuestro servicio web. A lo largo del desarrollo se hará uso de los servicios de Amazon con ser su CLI, EC2 y la herramienta Ansible que nos servirá para crear una AMI (Amazon Machine Image) personalizada con nuestra aplicación.

Esquema General

Objetivos

- Utilizar Amazon EC2 para el despliegue de API REST.
- Incluir grupo de auto-escalado.
- Creación de AMI personalizada para la aplicación.
- Utilizar AWS CLI para automatizar el despliegue.



Nuestra infraestructura consta de una instancia EC2 configurada con un gestor de base de datos de MongoDB. Luego tendremos una AMI configurada con un servidor NodeJS y nuestra aplicación escuchando el puerto 8080. Luego crearemos un balanceador de carga y por último configuraremos un grupo de auto escalado a partir de nuestra AMI.

Desarrollo

Requisitos previos

Para realizar de forma automatizada el despliegue de nuestra API REST haremos tendremos como sistema operativo una distribución de Linux, en este caso se realizó con la distribución Linux Mint y con ella las herramientas CLI de Amazon y Ansible.

Otorgar permisos a nuestro usuario Linux

Para una mejor experiencia en el uso del script otorgaremos a nuestro usuario todos los permisos mediante los siguientes comandos:

```
sudo visudo
# User privilege specification
root ALL=(ALL:ALL) ALL
miusuario ALL=(ALL:ALL) ALL
```

Instalación de CLI Amazon

```
sudo apt-get install python-pip
pip install awscli
```

Una vez instalado en nuestro sistema realizaremos la configuración de acceso a la plataforma de Amazon mediante el comando:

```
aws configure
AWS Access Key ID [None]: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
AWS Secret Access Key [None]: XXXXXXXXXXXXXXXXXXXXXXXXXXXX
Default region name [None]: us-west-2
Default output format [None]: json
```

Tendremos que agregar nuestras credenciales como también la región con la que estamos registrados.

Instalación de Ansible

Ansible es una herramienta de software libre para la configuración y administración de computadoras. Para instalar la herramienta ejecutaremos los siguientes comandos:

```
sudo apt-get install software-properties-common
sudo apt-add-repository ppa:ansible/ansible
sudo apt-get update
sudo apt-get install ansible
```

Para tener una mejor experiencia en el uso de ansible desactivaremos la aprobación de host key en el archivo de configuración de ansible.

```
sudo nano /etc/ansible/ansible.cfg
host_key_checking = False
```

Scripts de configuración

Creación de instancia con base de datos:

create_database.sh

```
#!/bin/bash
id=$(aws ec2 run-instances --image-id ami-841f46ff --key-name aluccloud35-
keypair --security-group-ids sg-c171d4b4 --instance-type t2.micro --
subnet-id subnet-2bfb6c4f --query 'Instances[*].[InstanceId]' --output
text)
echo $id
while true; do
    code=$(aws ec2 describe-instance-status --instance-ids $id --
query 'InstanceStatuses[*].InstanceStatus.[Status]' --output text)
    echo $code
    if [ "$code" = "ok" ]; then
        dns=$(aws ec2 describe-instances --instance-ids $id --
query 'Reservations[*].Instances[*].[PublicDnsName]' --output text)
        echo $dns
        break
    fi
    sleep 8
done

sudo echo -e "[webserver]\n$dns">/etc/ansible/hosts
sudo echo -e "module.exports={IP_DATABASE:'$dns'}">config.js
ansible-playbook -e 'host_key_checking=False' --private-key aluccloud35-
keypair.pem mongodb.yml
```

Las letras en rojo especifican recursos previos creados en plataforma de Amazon que no se explican en este documento, pero algunos archivos se incluyen en la carpeta del proyecto. Los pasos que sigue el script anterior son:

1. Creación de una instancia a partir de una AMI Linux en el servicio EC2 de Amazon.
2. Verificación del estado de creación de la instancia.
3. Configuración del archivo host de ansible (se necesitan permisos de root para ejecutar este comando).
4. Copiamos el DNS de nuestra instancia de base de datos al archivo de configuración de nuestra aplicación.
5. Ejecución de la herramienta ansible para la instalación y configuración de nuestro gestor de base de datos.

A continuación, se muestra el archivo de confirmación usado por ansible para instalar MongoDB como gestor de base de datos no relacional.

```
---
- hosts: all
  user: ubuntu
  gather_facts: True
  sudo: True
  tasks:
    - name: Update and upgrade apt packages
      become: true
      apt:
        upgrade: yes
    - name: MongoDB | Import public key
      apt_key:
        keyserver: hkp://keyserver.ubuntu.com:80
        id: EA312927
    - name: MongoDB | Add repository
      apt_repository:
        filename: '/etc/apt/sources.list.d/mongodb-org-3.2.list'
        repo: 'deb http://repo.mongodb.org/apt/ubuntu trusty/mongodb-
org/3.2 multiverse'
        state: present
        update_cache: yes
    - name: MongoDB | Install MongoDB
      sudo: yes
      apt:
        force: yes
        name: mongodb-org
        state: present
        update_cache: yes
    - name: creating directory
      shell: cd / && sudo mkdir data && cd data && sudo mkdir db
    - copy:
      src: mongod.conf
      dest: /etc/mongod.conf
    - copy:
      src: mongodb.service
      dest: /etc/systemd/system/mongodb.service
    - name: install systemd
      become: yes
      apt: name=systemd
    - name: start mongodb
      shell: sudo systemctl start mongodb
    - name: service enable mongodb
      shell: sudo systemctl enable mongodb
    - name: reset service mongod
      shell: sudo service mongod restart
```

Creación de AMI y Auto escalado de aplicación:

create_app.sh

```
#!/bin/bash
id=$(aws ec2 run-instances --image-id ami-841f46ff --key-name aluccloud35-
keypair --security-group-ids sg-c171d4b4 --instance-type t2.micro --
subnet-id subnet-2bfb6c4f --query 'Instances[*].[InstanceId]' --output
text)
echo $id
while true; do
    code=$(aws ec2 describe-instance-status --instance-ids $id --
query 'InstanceStatuses[*].InstanceStatus.[Status]' --output text)
    echo $code
    if [ "$code" = "ok" ]; then
        dns=$(aws ec2 describe-instances --instance-ids $id --
query 'Reservations[*].Instances[*].[PublicDnsName]' --output text)
        echo $dns
        break
    fi
    sleep 8
done
sudo echo -e "[webserver]\n$dns">/etc/ansible/hosts
echo "Configurando AMI"
ansible-playbook -e 'host_key_checking=False' --private-key aluccloud35-
keypair.pem node.yml
ami=$(aws ec2 create-image --instance-id $id --name "ICP_AMI_API_$id" --
description "API nodejs" --output text)
echo $ami
while true; do
    code=$(aws ec2 describe-images --image-ids $ami --query
'Images[*].[State]' --output text)
    echo "estado de AMI: "$code
    if [ "$code" = "available" ]; then
        break
    fi
    sleep 5
done
echo "ami creada!!"
echo "creando configuracion de autoscaling.."
aws autoscaling create-launch-configuration --launch-configuration-name
launch-config-$id --image-id $ami --instance-type t2.micro --security-
groups sg-c171d4b4 --key-name aluccloud35-keypair --associate-public-ip-
address
echo "creando balanceador de carga.."
aws elbv2 create-load-balancer --name lb-$id --subnets subnet-2bfb6c4f
subnet-c2f25afd --scheme internet-facing --security-groups sg-c171d4b4
lb=$(aws elbv2 describe-load-balancers --names lb-$id --query
'LoadBalancers[*].[LoadBalancerArn]' --output text)
echo "creando target group..."
```



```

tg=$(aws elbv2 create-target-group --name lb-$id-tg --protocol HTTP --
port 8080 --vpc-id vpc-83a213fb --query
'TargetGroups[*].[TargetGroupArn]' --output text)
#registramos las instancias
echo "Register target group..."
aws elbv2 register-targets --target-group-arn $tg --targets Id=$id
#asignamos tg a lb
echo "creando listener de balanceador de carga..."
aws elbv2 create-listener --load-balancer-arn "$lb" --protocol HTTP --
port 8080 --default-actions Type=forward,TargetGroupArn="$tg"
echo "creando grupo de autoescalado..."
aws autoscaling create-auto-scaling-group --auto-scaling-group-name as-
group-$id --launch-configuration-name launch-config-$id --min-size 1 --
max-size 2 --default-cooldown 120 --target-group-arns $tg --vpc-zone-
identifier "subnet-2bfb6c4f,subnet-c2f25afd"
link=$(aws elbv2 describe-load-balancers --names lb-$id --query
'LoadBalancers[*].[DNSName]' --output text)
echo "conectar a: www.$link:8080"

```

El script anterior sigue los siguientes pasos:

1. Creación de una instancia a partir de una AMI Linux en el servicio EC2 de Amazon.
2. Verificación del estado de creación de la instancia.
3. Configuración del archivo host de ansible (se necesitan permisos de root para ejecutar este comando).
4. Ejecución de la herramienta ansible para la instalación y configuración de nuestra aplicación API REST.
5. Creación de nuestra AMI personalizada que incluye un servidor NodeJS y nuestra aplicación en el directorio **/home/Ubuntu/proyecto**
6. Verificación del estado de la creación de la AMI.
7. Creación de la configuración del grupo de auto escalado que hace uso de nuestra AMI personalizada.
8. Creación del balanceador de carga.
9. Creación de Target Group con el puerto 8080.
10. Registramos en nuestro Target Group nuestra instancia creada.
11. Creamos un listener para nuestro balanceador de carga que escuchara el puerto 8080.
12. Creación del grupo de auto escalado con un mínimo de una instancia y máximo de dos.

Ejecución

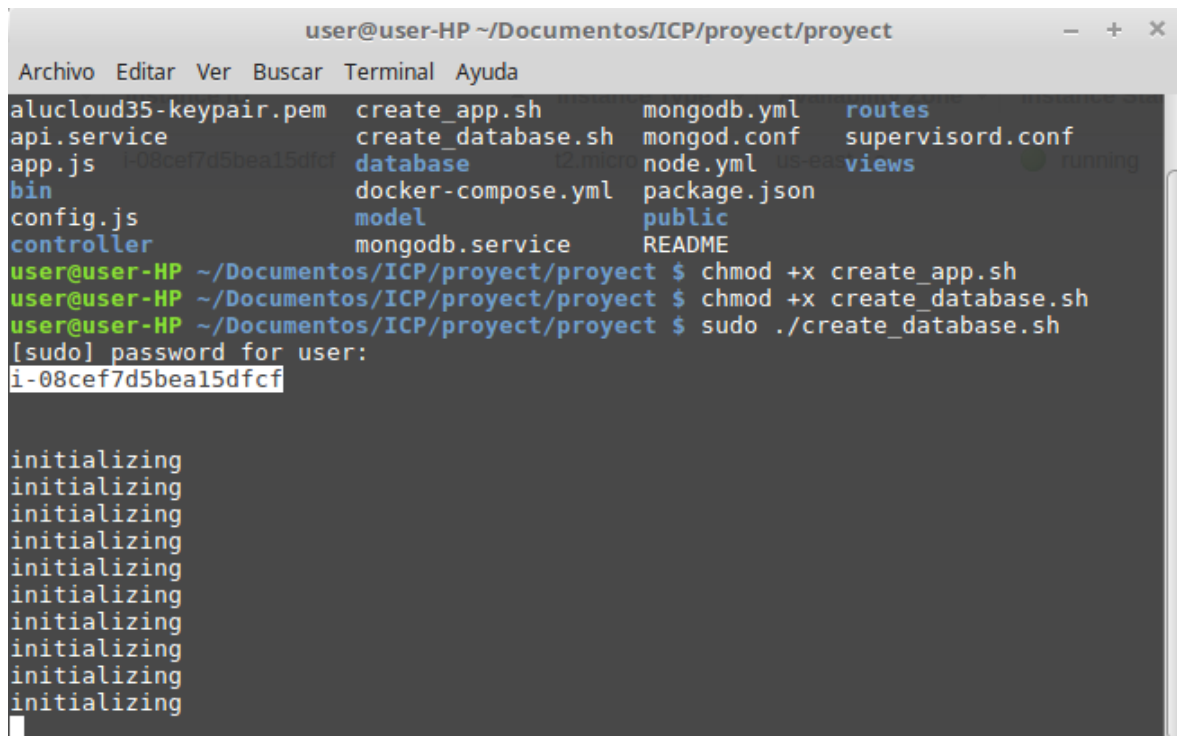
Para crear nuestras instancias como también nuestro grupo de auto escalado tendremos que estar en carpeta `project` que puedes descargar del siguiente repositorio de GitHub: https://github.com/adolfo24/autoscaling_aws

```
cd autoscaling_aws/project
```

Creación de Base de datos

```
sudo chmod +x create_database.sh
sudo ./create_database.sh
```

Con esto se lanzará automáticamente una instancia en el servicio EC2 de Amazon con MongoDB como gestor de base de datos para nuestra aplicación.



```
user@user-HP ~/Documentos/ICP/project/project
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
aluccloud35-keypair.pem  create_app.sh  mongodb.yml  routes
api.service              create_database.sh  mongod.conf  supervisord.conf
app.js                   database       node.yml     us-east-1      views
bin                      docker-compose.yml  package.json
config.js                model          public
controller               mongodb.service  README

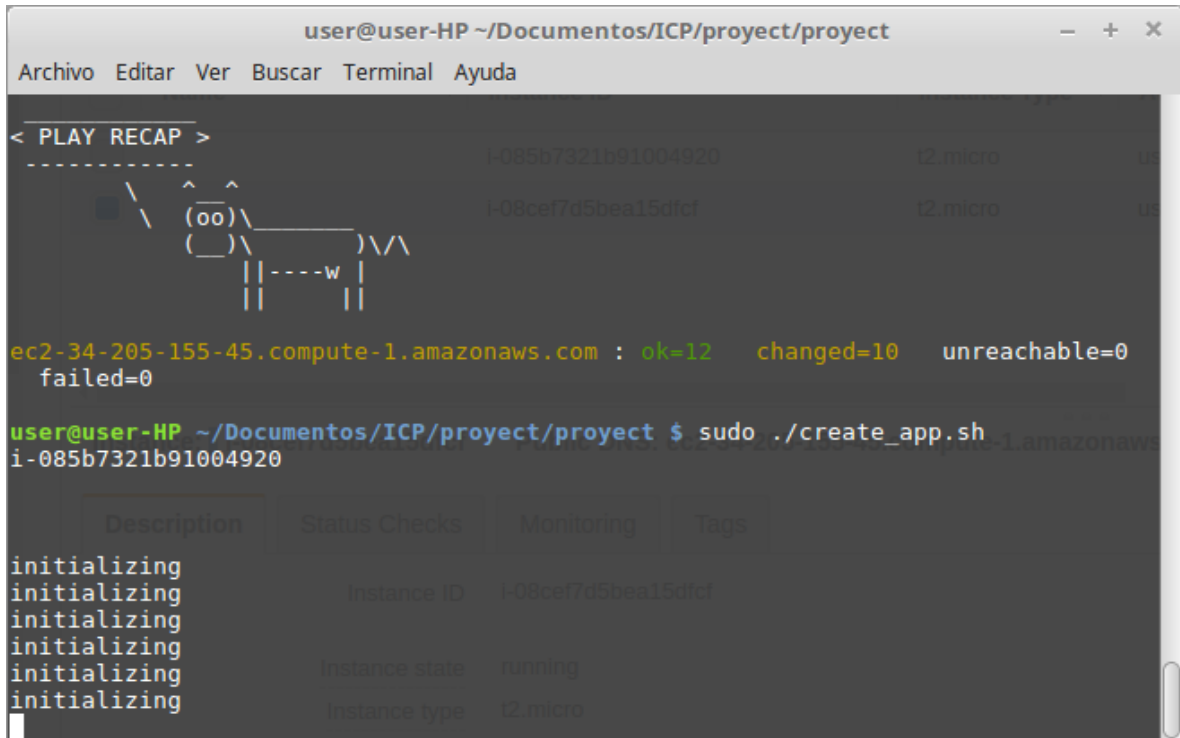
user@user-HP ~/Documentos/ICP/project/project $ chmod +x create_app.sh
user@user-HP ~/Documentos/ICP/project/project $ chmod +x create_database.sh
user@user-HP ~/Documentos/ICP/project/project $ sudo ./create_database.sh
[sudo] password for user:
i-08cef7d5bea15dfcf

initializing
initializing
initializing
initializing
initializing
initializing
initializing
initializing
initializing
initializing
```


Creación de infraestructura de alta disponibilidad

Ubicados en el mismo directorio ejecutamos los siguientes comandos para lanzar en los servicios de Amazon nuestra infraestructura de alta disponibilidad.

```
sudo chmod +x create_app.sh  
sudo ./create_app.sh
```



The screenshot shows a terminal window titled "user@user-HP ~/Documentos/ICP/proyect/proyect". The terminal displays the output of the command `sudo ./create_app.sh`. The output includes a "PLAY RECAP" section with a diagram of a network topology, followed by a summary of the execution results for the `ec2-34-205-155-45.compute-1.amazonaws.com` host. The results show `ok=12`, `changed=10`, `unreachable=0`, and `failed=0`. Below this, the terminal shows the command `sudo ./create_app.sh` being executed again, followed by the output of the `cat` command showing the contents of the `create_app.sh` script. The script is a shell script that creates an AWS EC2 instance with the following parameters:

Description	Status Checks	Monitoring	Tags
initializing			
initializing	Instance ID	i-08cef7d5bea15dfcf	
initializing	Instance state	running	
initializing	Instance type	t2.micro	

```
user@user-HP ~/Documentos/ICP/proyect/proyect
Archivo  Editar  Ver  Buscar  Terminal  Ayuda

changed: [ec2-34-236-238-188.compute-1.amazonaws.com]
i-085b7321b91004920 t2.micro us-east-1
< TASK [get nodejs 8] >
-----
      \  ^__^
       (oo)\_______
          (__)\       )\/\
              ||----w |
              ||     ||

[WARNING]: Consider using get_url or uri module rather than running curl
Instance: i-08cef7d5bea15dfcf Public DNS: ec2-34-205-155-45.compute-1.amazonaws.com
changed: [ec2-34-236-238-188.compute-1.amazonaws.com]
i-085b7321b91004920 t2.micro us-east-1
< TASK [install nodejs] >
-----
      \  ^__^
       (oo)\_______
          (__)\       )\/\
              ||----w |
              ||     ||

Instance ID      i-08cef7d5bea15dfcf
Instance state   running
Instance type    t2.micro
```

```
user@user-HP ~/Documentos/ICP/proyect/proyect
Archivo  Editar  Ver  Buscar  Terminal  Ayuda

changed: [ec2-34-236-238-188.compute-1.amazonaws.com]
i-085b7321b91004920 t2.micro us-east-1
< PLAY RECAP >
-----
      \  ^__^
       (oo)\_______
          (__)\       )\/\
              ||----w |
              ||     ||

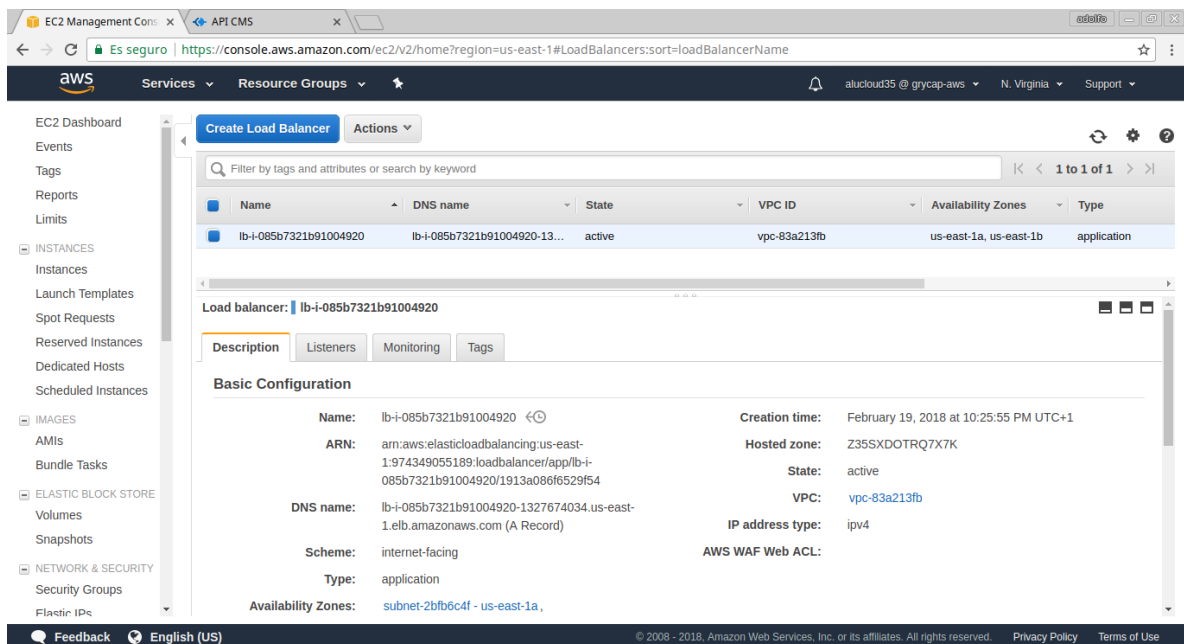
ec2-34-236-238-188.compute-1.amazonaws.com : ok=11  changed=10  unreachable=0  failed=0
ami-99c024e4
estado de AMI: pending
estado de AMI: pending
estado de AMI: pending
estado de AMI: pending
estado de AMI: pending
estado de AMI: pending
estado de AMI: pending
estado de AMI: pending
Instance ID      i-08cef7d5bea15dfcf
Instance state   running
Instance type    t2.micro
```

```
user@user-HP ~/Documentos/ICP/proyect/proyect
Archivo Editar Ver Buscar Terminal Ayuda

ami creada!!
creando configuracion de autoscaling..
creando balanceador de carga..
LOADBALANCERS  Z35SXD0TRQ7X7K  2018-02-19T21:25:55.430Z  lb-i-085b7321b91004920-1327674034.us-east-1.elb.amazonaws.com  ipv4  arn:aws:elasticloadbalancing:us-east-1:974349055189:loadbalancer/app/lb-i-085b7321b91004920/1913a086f6529f54  lb-i-085b7321b91004920  internet-facing application  vpc-83a213fb
AVAILABILITYZONES  subnet-2bfb6c4f us-east-1a
AVAILABILITYZONES  subnet-c2f25afd us-east-1b
SECURITYGROUPS  sg-c171d4b4
STATE  provisioning
creando target group...
Register target group...
creando listener de balanceador de carga...
LISTENERS  arn:aws:elasticloadbalancing:us-east-1:974349055189:listener/app/lb-i-085b7321b91004920/1913a086f6529f54/72603ca976372ba1  arn:aws:elasticloadbalancing:us-east-1:974349055189:loadbalancer/app/lb-i-085b7321b91004920/1913a086f6529f54  8080  HTTP
DEFAULTACTIONS  arn:aws:elasticloadbalancing:us-east-1:974349055189:targetgroup/lb-i-085b7321b91004920-tg/6af6c869dae5227a  forward
creando grupo de autoescalado...
conectar a: www.lb-i-085b7321b91004920-1327674034.us-east-1.elb.amazonaws.com:8080
user@user-HP ~/Documentos/ICP/proyect/proyect $
```

al finalizar la ejecución de este script contaremos con:

1. Una instancia EC2 sobre la que corre nuestra aplicación escuchando en el puerto 8080.
2. Un balanceador de carga que sirve nuestra aplicación en el puerto 8080.



- Un grupo de auto escalado que tendrá siempre como mínimo una instancia y máximo dos para mantener nuestro servicio siempre disponible.

The screenshot shows the AWS Management Console interface. On the left is a navigation menu with categories like AMIs, ELASTIC BLOCK STORE, NETWORK & SECURITY, and AUTO SCALING. The 'Auto Scaling Groups' link is selected. The main content area displays a table of Auto Scaling Groups. One group, 'as-group-i-085b7321b91004920', is selected. Below the table, the 'Details' tab is active, showing configuration parameters for the selected group. The parameters include Launch Configuration, Launch Template, Service-Linked Role, Load Balancers, Target Groups, Availability Zone(s), Subnet(s), and Default Cooldown.

Name	Launch Configuration /	Instances	Desired	Min	Max	Availability Zones	Default Cooldown	Health Check Grac
as-group-i-085b...	launch-config-i-085b73...	1	1	1	2	us-east-1a, us-east-1b	120	0

Auto Scaling Group: as-group-i-085b7321b91004920

Details | Activity History | Scaling Policies | Instances | Monitoring | Notifications | Tags | Scheduled Actions | Lifecycle Hooks

Launch Configuration: launch-config-i-085b7321b91004920

Launch Template

Launch Template Version

Service-Linked Role

Load Balancers

Target Groups: lb-i-085b7321b91004920-ig

Desired: 1

Min: 1

Max: 2

Availability Zone(s): us-east-1a, us-east-1b

Subnet(s): subnet-2bfb6c4f, subnet-c2f25afd

Default Cooldown: 120

- Nuestra aplicación en la dirección DNS de nuestro balanceador de carga escuchando el puerto 8080.

The screenshot shows a web browser window with the URL 'lb-i-085b7321b91004920-1327674034.us-east-1.elb.amazonaws.com:8080'. The browser displays a web application titled 'API CMS'. At the top, there are navigation links: 'Inicio', 'Users', and 'new User'. Below the navigation links, the text 'API CMS' is displayed in a large, bold font, followed by 'Welcome to API CMS' in a smaller font.

Conclusiones

1. El uso de plataformas que brindan servicios de infraestructura en la nube actualmente está siendo cada vez más aceptado y usado tanto por la empresa privada como por entidades gubernamentales gracias a su facilidad de implementación y bajos costes.
2. Si somos emprendedores y queremos tener nuestros servicios en internet, las plataformas cloud hasta el momento es la mejor opción para echar a andar nuestros servicios a un bajo coste y poder escalar a medida que nuestro servicio sea más exigido.
3. Amazon Web Service por hoy es una de las plataformas de infraestructuras más populares que cuenta con una amplia gama de servicios con los que podemos hacer infraestructuras de para la mayoría de aplicaciones.

Referencias

1. <https://aws.amazon.com/es/documentation/>
2. <http://docs.ansible.com/>