Actividad 3: Introducción a los servicios de Amazon Web Services (AWS)

Objetivos de la actividad

* Lograr un acercamiento a las potencialidades de Amazon Web Services, en particular, la creación de instancias EC2.
* Manejo de dockers.
* Creación de un entorno de desarrollo con una plataforma como servicio (Elastic Beanstalk).
* Ejecución de funciones serverless.

Descripción de la actividad

En el desarrollo de la actividad se deberá realizar las siguientes actividades:

**Fase 1.** Creación Instancia de EC2 Linux con una página web (opcional, creación de una instancia EC2 Windows limpia).

Procedimiento:

* Selección el servicio EC2.
* Pulsar en Launch Instance.
* Elegir una AMI.
* Elegir tipo de instancia.
* Configurar los detalles de la instancia. El código para User Data es:

**a.** #!/bin/bash

**b.** yum -y install httpd

**c.** systemctl enable httpd

**d.** systemctl start httpd

**e.** echo "Hello World from $(hostname -f)" > /var/www/html/index.html

* Añadir el almacenamiento.
* Anadir una etiqueta.
* Agregar un grupo de seguridad. Asegurarse de agregar el puerto 22 para control SSH y 80 para ver la web.
* Revisar y lanzar la instancia.
* Generar el par de claves público-privado.
* Revisar el estatus de la máquina virtual.
* Conectar por consola.
* Conectar con la aplicación Putty a la instancia por su IP pública.

EXTRA: realizar la instalación de una segunda instancia con Windows. En este caso no usaremos el código de user data y no conectaremos por Putty, sino que conectaremos por escritorio remoto.

**Fase 2.** Dockers. Instalación de un servidor web NGINX (extra, crear un segundo Docker que sea un tablero Kanban).

Procedimiento:

* Instalación de una instancia de EC2 Linux.
* Nos conectamos por consola o putty.
* Actualizamos e instalamos Docker en la instancia EC2.

sudo su

yum update -y

yum install -y docker

* Arrancamos el servicio Docker.

service docker start

* Nos validamos en Docker Registry.

Docker login (con las credenciales de cada uno de nosotros).

* Instalamos Nginx cogiendo una imagen de la web de Docker.com.

docker run -d -p 80:80 –name nginx nginx

* Comprobamos si está creado.

docker ps

* Probamos el servidor web en un navegador (<http://ip-publica-EC2/>).

**EXTRA:** busca en Docker.com un tablero Kanban e intenta instalarlo en otro Docker. Disponible es: <https://docs.kanboard.org/en/1.2.2/admin_guide/docker.html>

**Fase 3.** Servicios utilizando una plataforma como servicio como es Elastic beanstalk.

Con Elastic Beanstalk podemos implementar, monitorizar y escalar una aplicación con rapidez y facilidad. Es una especie de *wizard* que le permite crear aplicaciones con facilidad ocultándole los detalles técnicos de los servicios de Amazon, que son utilizados por la aplicación que usted instala (EC2, Route 33, S3 u otros).

Procedimiento:

* Diríjase al servicio Elastic Beanstalk de su cuenta de AWS.
* Cree una aplicación indicando su nombre, la plataforma sobre la cual se desplegará:
  + **a.** Elija para este primer ejercicio Tomcat.
  + **b.** Use un código de muestra de la aplicación. AWS tomará el código disponible de Tomcat en el hub de AWS.
* Cree la aplicación haciendo clic en el botón indicado por el Elastic Beanstalk.
* En este momento Elastic Beanstalk creará la aplicación, instanciando los servicios de AWS de los cuales haga uso la aplicación en cuestión.
* Una vez finalice la instalación, despliegue la aplicación instalada utilizando el enlace provisto por el Elastic Beanstalk.
* Diríjase al servicio EC2 de su cuenta de AWS y verifique la nueva instancia creada por Elastic Beanstalk en este servicio.
* Verifique cualquier otro servicio utilizado por Elastic Beanstalk.
* Elimine los servicios utilizados.

**Fase 4.** Funciones Lambda (Serverless). Creación de una función lambda desde cero (extra, crear una función lambda para parar una instancia de EC2).

Procedimiento:

* Navegar al servicio Lambda.
* Pulsar en el botón de crear una función.
* Poner nombre a la función.
* Elegir el Runtime Python 3.8.
* Crear la función.
* Busca el texto prueba y cambia Hello From Lambda por Hola Mundo.
* Probar la función, pulsando Deploy y luego Test.
* Damos el nombre Prueba al «Event name» y volvemos a pulsar Test.
* Vemos el resultado de tiempo y CPU consumido.

EXTRA: crea una nueva función StopEC2 con python 3.8 también y utiliza el siguiente código para parar una instancia EC2 creada con anterioridad.

IMPORTANTE: cambia el ID de la instancia de este código por el ID de tu instancia.

import boto3

region = 'eu-west-1'

instances = ['i-0d6a9b386e8e2553a']

ec2 = boto3.client('ec2', region\_name=region)

def lambda\_handler(event, context):

ec2.stop\_instances(InstanceIds=instances)

print('stopped your instances: ' + str(instances))

Extensión y formato

Para todas las fases, el estudiante debe reportar los resultados de cada uno de los puntos de la actividad en un informe que no exceda las 10 páginas.

Rúbrica

Según fichero Excel adjunto,

Enlaces de ayuda

AWS Academy <https://unir.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=1c2361b1-c759-4835-b939-ae000052b6ff>

AWS EC2 <https://unir.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=6e2416ac-bd3b-4471-90ee-afa901528a04>

AWS Contenedores <https://unir.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=a357f735-b1c9-483a-b86f-afa9015289b0>

AWS ElasticBeanstalk <https://unir.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=38c39498-5175-4711-8966-afa9015289de>

AWS Lambda <https://unir.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=37cea0cd-b6bd-44f7-87a8-afa901553920>