Herramientas Devops

Introducción

En esta actividad, aprenderás a utilizar Packer mediante un ejemplo sencillo, pero con una stack (pila) tecnológica muy utilizada. Los objetivos que se pretenden conseguir son los siguientes:

- Aplicar la herramienta a la adaptación de unas instrucciones a automatismos para crear imágenes reutilizables.
- Crear una template (plantilla) de Packer que te permita generar una imagen con una aplicación con Node.js ya instalada y configurada con Nginx como servidor web.
- Realizar el despliegue en una nube pública (se recomienda Amazon AWS, pero es opcional).
- Ejecutar un despliegue automático, sin intervención manual.

Requisitos

Para realizar esta actividad necesitarás los siguientes recursos:

- Una cuenta en GitHub.
- Una cuenta en Amazon AWS.
- Una cuenta en AZURE.
- Un entorno de desarrollo con AWS CLI instalado.
- Un entorno de desarrollo con Packer instalado.
- Un entorno de desarrollo con Node.js instalado.
- Un entorno de desarrollo con Nginx instalado.
- Un entorno de desarrollo con Git instalado.

Desarrollo

Ejercicio 1

Creación de una template de Packer. Con el proyecto seleccionado, debes crear una template (plantilla) de Packer que te permita generar una imagen mediante una aplicación con Node.js ya instalada y configurada con Nginx como servidor web. Para desarrollar la plantilla, aplica la información que se recoge en el enlace de apoyo proporcionado con anterioridad. En esta tarea, se tendrá en cuenta el uso de IP pública.

Crear un repositorio en GitHub

1. Crea un repositorio en GitHub con el nombre que desees.
2. Clona el repositorio en tu entorno de desarrollo.
3. Crea un archivo README.md con la descripción de la actividad.
4. Realiza un commit con el mensaje "Creación del repositorio".
5. Realiza un push al repositorio remoto.

Crear un proyecto básico de Node.js

1. Crea un proyecto básico de Node.js en tu entorno de desarrollo.
2. Crea un archivo app.js con el siguiente contenido:

```
const express = require('express');
const app = express();

app.get('/', (req, res) => {
  res.send('¡Hola Mundo!');
});

const PORT = process.env.PORT || 3000;

app.listen(PORT, () => {
  console.log(`Servidor corriendo en puerto ${PORT}`);
});
```


Node app running

Configurar Nginx como Proxy Inverso

1. Navega a la carpeta sites-available de Nginx.

```
cd /etc/nginx/sites-available
```

2. Crea un archivo de configuración para tu aplicación.

```
sudo nano express_app
```

3. Agrega la siguiente configuración:

```
server {
    listen 80;
    server_name localhost;

    location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

4. Crea un enlace simbólico en la carpeta sites-enabled.

```
sudo ln -s /etc/nginx/sites-available/express_app /etc/nginx/sites-enabled/express_app
```

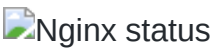
5. Eliminar el archivo default de sites-enabled.

```
sudo rm /etc/nginx/sites-enabled/default
```

6. Reinicia el servicio de Nginx.

```
sudo systemctl restart nginx
```

7. Prueba que tu aplicación de Node.js está funcionando correctamente.



Crear una template de Packer

1. Copia el archivo nginx.conf en la carpeta de tu proyecto.

2. Crea un archivo .pkr.hcl con la siguiente configuración:

```
packer {
  required_plugins {
    amazon = {
      version = ">= 1.2.8"
      source  = "github.com/hashicorp/amazon"
    }
  }
}

variable "region" {
  type    = string
  default = "us-east-1"
}

source "amazon-ebs" "ubuntu" {
  region        = var.region
  ami_name      = "devops-tools-nginx-nodejs"
  instance_type = "t2.micro"
  source_ami_filter {
    filters = {
      name                = "ubuntu/images/hvm-ssd/ubuntu-focal-20.04-amd64-server-*"
      root-device-type    = "ebs"
      virtualization-type = "hvm"
    }
    most_recent = true
    owners      = ["099720109477"]
  }
  ssh_username = "ubuntu"
}

build {
  name = "packer-devops-template"
  sources = [
    "source.amazon-ebs.ubuntu"
  ]

  provisioner "shell" {
    inline = [
      "curl -fsSL https://deb.nodesource.com/setup_14.x | sudo -E bash -",
      "sudo apt-get update",
      "sudo apt-get install -y nodejs nginx",
      "sudo npm install -g n",
      "sudo n stable",
      "sudo mkdir -p /var/www/node_app",
```

```
      "sudo chown -R ubuntu:ubuntu /var/www",
      "git clone https://github.com/LuisArana631/packer-devops-tools-hw.git /var/www/node
      "cd /var/www/node_app/nodejs-project/src && sudo npm i",
      "sudo chown -R ubuntu:ubuntu /var/www/node_app",
      "echo '[Unit]\nDescription=Node.js App\n[Service]\nExecStart=/usr/bin/node /var/www
      "sudo systemctl enable nodeapp",
      "sudo systemctl start nodeapp",
      "sudo cp /var/www/node_app/nodejs-project/nginx.conf /etc/nginx/sites-available/nod
      "sudo ln -s /etc/nginx/sites-available/node_app /etc/nginx/sites-enabled/",
      "sudo rm /etc/nginx/sites-enabled/default",
      "sudo nginx -t",
      "sudo systemctl restart nginx"
    ]
  }
}
```

3. Ejecuta el siguiente comando para instalar el plugin de Packer para AWS.

```
packer init .
```


Packer Init

4. Configura tus credenciales para poder acceder a tu cuenta de AWS. Para obtener tus credenciales de AWS puedes ir a tu usuario > Security Credentials > Access keys.

```
export AWS_ACCESS_KEY_ID="<YOUR_AWS_ACCESS_KEY_ID>"
export AWS_SECRET_ACCESS_KEY="<YOUR_AWS_SECRET_ACCESS_KEY>"
```

5. Formatea el archivo .pkr.hcl.

```
packer fmt .
```

6. Valida el archivo .pkr.hcl.

```
packer validate .
```


Packer Validate

7. Ejecuta el comando packer build.

```
packer build .
```

 Packer Build Log de ejecucion

8. Comprueba que la imagen se ha creado correctamente en tu cuenta de AWS.

 AMI Creada

Prueba del AMI en AWS

1. Crea una instancia en AWS con la imagen que has creado.

   Debes agregar los permisos para poder acceder a las peticiones HTTP

2. Comprueba que la instancia está funcionando correctamente.
3. Accede a la IP pública de la instancia.
4. Comprueba que la aplicación de Node.js está funcionando correctamente.

Ejercicio 2

Despliegue sin intervención manual. Ahora, debes modificar la imagen y la template de la tarea anterior de tal forma que la imagen se cree y el despliegue completo se realice de forma automática, sin intervención manual. Te aconsejamos revisar documentación del proveedor para utilizar el CLI.

Despliege usando AWS CLI

1. Si no has configurado tus credenciales de AWS, ejecuta el siguiente comando.

```
export AWS_ACCESS_KEY_ID="<YOUR_AWS_ACCESS_KEY_ID>"
export AWS_SECRET_ACCESS_KEY="<YOUR_AWS_SECRET_ACCESS_KEY>"
```

2. Ejecuta el siguiente comando para crear una instancia en AWS.

```
aws ec2 run-instances --image-id <AMI_ID> --count 1 --instance-type t2.micro --key-name <
```

EC2 Console Node App AWS CLI

Log de ejecución

Ejercicio 3

Añade un builder de otro tipo para una nube pública (secundario AZURE) en caso de usar otro proveedor y justifica su uso.

Agregar un builder para AZURE

1. Crea un archivo .pkr.hcl con la siguiente configuración:

```
packer {
  required_plugins {
    azure = {
      version = ">= 1.9.0"
      source  = "github.com/hashicorp/azure"
    }
  }
}

variable "location" {
  type    = string
  default = "East US"
}

source "azure-arm" "autogenerated_1" {
  use_azure_cli_auth                = true
  image_offer                       = "0001-com-ubuntu-server-jammy"
  image_publisher                   = "canonical"
  image_sku                         = "22_04-lts"
  location                          = var.location
  managed_image_name                = "myPackerImage"
  managed_image_resource_group_name = "devops-group-tools"
  os_type                           = "Linux"
  vm_size                           = "Standard_DS2_v2"
}

build {
  name = "packer-azure-devops-template"
  sources = [
    "source.azure-arm.autogenerated_1"
  ]

  provisioner "shell" {
    inline = [
      "curl -fsSL https://deb.nodesource.com/setup_14.x | sudo -E bash -",
      "sudo apt-get update",
      "sudo apt-get install -y nodejs nginx",
```

```
        "sudo npm install -g n",
        "sudo n stable",
        "sudo mkdir -p /var/www/node_app",
        "sudo chown -R ubuntu:ubuntu /var/www",
        "git clone https://github.com/LuisArana631/packer-devops-tools-hw.git /var/www/node
        "cd /var/www/node_app/nodejs-project/src && sudo npm i",
        "sudo chown -R ubuntu:ubuntu /var/www/node_app",
        "echo '[Unit]\nDescription=Node.js App\n[Service]\nExecStart=/usr/bin/node /var/www
        "sudo systemctl enable nodeapp",
        "sudo systemctl start nodeapp",
        "sudo cp /var/www/node_app/nodejs-project/nginx.conf /etc/nginx/sites-available/nod
        "sudo ln -s /etc/nginx/sites-available/node_app /etc/nginx/sites-enabled/",
        "sudo rm /etc/nginx/sites-enabled/default",
        "sudo nginx -t",
        "sudo systemctl restart nginx"
      ]
    }
}
```

2. Ejecuta el siguiente comando para instalar el plugin de Packer para AZURE.

```
packer init .
```


Packer Init

3. Haz login en AZURE.

```
az login
```

4. Crea el grupo de recursos en AZURE.

```
az group create --name devops-group-tools --location "East US"
```

5. Ejecuta el comando packer build.

```
packer build .
```