Mejoramiento de la productividad: Automatización (MISO 4202)

Semestre 202010

Enunciado del taller de parejas

(Inspirado en idea de Miguel González)

Contexto

Con la llegada de la computación en la nube (cloud computing), hoy en día, es más fácil, rápido y barato para los ingenieros aprovisionar infraestructura IT y servicios. Los proveedores Amazon Web Services (AWS) y Google Cloud Platform (GCP) ofrecen una variedad de servidores y servicios, por ejemplo, servidores de base de datos relacionales y no relacionales, servicios de analítica basados en inteligencia artificial. La creación de ambientes de desarrollo, pruebas y producción dentro de las plataformas de cloud es un proceso que requiere de conocimientos específicos sobre el proveedor que se está usando. Por otro lado, si se usan las interfaces web de los proveedores, la configuración de la infraestructura implica una serie de pasos que pueden volverse tediosos y repetitivos. La mayoría de los proveedores ofrecen herramientas de línea de comando para automatizar estos pasos.

En los últimos años han surgido herramientas que permiten especificar infraestructura como código (IaC). La infraestructura como código es un método de aprovisionamiento y gestión de infraestructura IT y servicios a través del uso de código fuente, sustituyendo el procedimiento estándar de operación.

Ejemplos de herramientas de IaC son Terraform, Pulumi, Ansible las cuales permiten configurar la infraestructura, en diferentes proveedores de nube, a través de scripts. Dichos scripts se pueden compartir, reutilizar y versionar usando un repositorio de versiones. El hecho de poder reutilizar las configuraciones (semi) automatiza el aprovisionamiento y la gestión de la infraestructura.

Problema

La empresa Ceffective, que desarrolla aplicaciones empresariales, realiza el aprovisionamiento de infraestructura usando la interfaz web de los proveedores de nube, haciendo que el despliegue de aplicaciones en distintos ambientes demore mucho tiempo.

Por lo anterior, la empresa desea incursionar en las herramientas de IaC. Un inconveniente es que cada herramienta define su propia sintaxis lo que conlleva a que el desarrollador tenga que aprender una sintaxis diferente por cada una y migrar los scripts si desea cambiar de herramienta. La empresa se pregunta si existe una solución que le permita al desarrollador definir una configuración agnóstica a la herramienta y que luego se genere el script correspondiente a la herramienta escogida o deseada.

Reto

Para responder a la pregunta de la empresa, en este curso se desarrollará un robot que genere scripts de configuración de infraestructura a partir de una ***especificación agnóstica***. Los scripts están escritos en la sintaxis definida en por lo menos una de las herramientas IaC (e.g., Terraform, Pulumi, Ansible).

La ***especificación*** debe permitir la definición de:

- El proveedor de nube que se va a utilizar (e.g., AWS, GCP) con sus respectivos parámetros de conexión, que en la mayoría de los casos corresponde a un usuario y contraseña o una llave de acceso con un secreto.
- Ambiente de despliegue con un nombre como por ejemplo desarrollo, pruebas, QA y producción, y para cada ambiente una lista de recursos que se quieren aprovisionar.
- Los recursos que la empresa requiere, inicialmente, son servidores de aplicaciones, servidores de bases de datos y almacenamiento. Entre los recursos se establecen conexiones, por ejemplo, servidores de aplicaciones con bases de datos o servidores de aplicaciones con almacenamientos
- Un servidor de aplicaciones puede ser especificado con un nombre, el tamaño de la máquina que se quiere aprovisionar, puede ser micro, small, medium o large, y el sistema operativo.
- Un servidor de bases de datos debe tener un nombre, un tipo (que puede ser relacional o NoSQL), el sistema manejador de bases de datos a utilizar (i.e. PostgreSQL, MySQL, Oracle, SQLServer) y el tamaño del servidor, micro, small, medium o large.
- Uno o varios servidores de aplicaciones pueden compartir almacenamientos por lo que debe poderse especificar almacenamientos con un nombre y un tamaño inicial de almacenamiento en Giga Bytes y poderse asociar a servidores de aplicaciones.

- Mecanismo de seguridad de la infraestructura, por lo que debe ser posible especificar un Internet Gateway con un nombre, redes virtuales privadas (VPC) con un nombre y un CIDR (Inter-Domain Routing (CIDR) block; por ejemplo, 10.0.0.0/16). Se asigna una VPC a cada Internet Gateway. También se pueden especificar subredes cada una con un nombre, la VPC a utilizar, un CIDR (Inter-Domain Routing (CIDR) block; por ejemplo, 10.0.0.0/16) y una zona de disponibilidad. Además debe poderse especificar grupos de seguridad con un nombre, una descripción , una VPC y una lista de reglas de entrada y otra lista de reglas de salida. Cada regla de entrada o salida se especifica definiendo el tipo, el protocolo, el puerto, el origen y una descripción. Todas los recursos de la VPC deben respetar las reglas definidas dentro del grupo de seguridad. Un ejemplo de estas reglas se muestra en la tabla 1. Finalmente cada recurso especificado deberá permitir seleccionar una VPC configurada.

Type	Protocol	Port Range	Source	Description
HTTP	TCP	80	0.0.0.0/0	Load Balancer
HTTPS	TCP	443	0.0.0.0/0	HTTPS Certificate
SSH	TCP	22	0.0.0.0/0	Conexión SSH
NFS	TCP	2049	0.0.0.0/0	EFS
PostgreSQL	TCP	5432	0.0.0.0/0	RDS

Tabla 1

Invariantes

1. Validar que las CIDR se escriban de forma correcta. Las CIDR se definen de forma específica en el RFC 4632, en nuestro caso debemos validar que el texto se escriba n.n.n.n/m, donde **n** corresponde a un número entre 0-255 y **m** un número entre 0-32.
2. Validar que un recurso de tipo base de datos no puede tener una conexión con un recurso de tipo almacenamiento.
3. Validar que de acuerdo al tipo de regla especificado, los valores de protocolo y puerto deben ser los valores fijos dados en la siguiente tabla:

Tipo de regla	Protocolo	Puerto
Custom TCP	TCP	1024 to 65535

Custom UDP	UDP	1024 to 65535
DNS UDP	UDP	53
DNS TCP	UDP	53
HTTP	TCP	80
HTTPS	TCP	443
SSH	TCP	22
NFS	TCP	2049
PostgresSQL	TCP	5432
MySQL/Aurora/MariaDB	TCP	3306
POP3	TCP	110
IMAP	TCP	143
LDAP	TCP	389
SMTP	TCP	25
Oracle	TCP	1521

4. Validar que de acuerdo al tipo de proveedor seleccionado (e.g., AWS, GCP) se estén aprovisionando solo los tipos de servidores de base de datos que ofrecen cada proveedor, como lo indica la siguiente tabla:

Proveedor	Tipo	Bases de datos soportadas
AWS	Relacional	PostgreSQL, Oracle, Aurora, MariaDB,MySQL,SQL,Server
AWS	NoSQL	DocumentDB,Dynamo,Cassandra,Redis
GCP	Relacional	-MySQL, PostgreSQL y SQL Server (CloudSQL soporta estos tres tipos). -CLOUD SPANNER
GCP	NoSQL	Cloud Big table, Cloud memory store (Redis), Cloud firestore, Firebase realtime database

5. Validar que de acuerdo al tipo de proveedor seleccionado (e.g., AWS, Azure) se estén aprovisionando recursos en zonas de disponibilidad apropiadas. A continuación, encontrarán dos tablas, una con las zonas de AWS y otra con las zonas de GCP:
   Para AWS:

Code	**Nombre**
us-east-2	EE.UU. Este (Ohio)
us-east-1	US East (N. Virginia)
us-west-1	EE.UU. Oeste (Norte de California)
us-west-2	EE.UU. Oeste (Oregón)
ap-east-1	Asia Pacífico (Hong Kong)
ap-south-1	Asia Pacífico (Mumbai)
ap-northeast-3	Asia Pacífico (Osaka-local)
ap-northeast-2	Asia Pacífico (Seúl)
ap-southeast-1	Asia Pacífico (Singapur)
ap-southeast-2	Asia Pacífico (Sídney)
ap-northeast-1	Asia Pacífico (Tokio)
ca-central-1	Canadá (Central)
eu-central-1	Europa (Fráncfort)

eu-west-1	Europa (Irlanda)
eu-west-2	Europa (Londres)
eu-west-3	Europa (París)
eu-north-1	Europa (Estocolmo)
me-south-1	Medio Oriente (Baréin)
sa-east-1	América del Sur (São Paulo)

Para GCP:

Región	**Ubicación**
asia-east1	Condado de Changhua, Taiwán
asia-east2	Hong Kong
asia-northeast1	Tokio, Japón
asia-northeast2	Osaka, Japón
asia-south1	Bombay, India
asia-southeast1	Jurong West, Singapur
australia-southeast1	Sídney, Australia
europe-north1	Hamina, Finlandia
europe-west1	Saint-Ghislain, Bélgica
europe-west2	Londres, Inglaterra, Reino Unido
europe-west3	Fráncfort, Alemania
europe-west4	Puerto de Ems, Países Bajos

europe-west6	Zúrich, Suiza
northamerica-northeast1	Montreal, Quebec, Canadá
southamerica-east1	Osasco (São Paulo), Brasil
us-central1	Council Bluffs, Iowa, EE.UU.
us-east1	Moncks Corner, Carolina del Sur, EE.UU.
us-east4	Ashburn, Virginia, EE.UU.
us-west1	The Dalles, Oregón, EE.UU.
us-west2	Los Ángeles, California, EE.UU.

Gramática

La empresa Ceffective propone hacer una gramática textual con el fin de que los desarrolladores de la empresa puedan especificar el despliegue que quieren realizar, independientemente de la herramienta de IaC. A continuación se muestra un posible ejemplo de especificación que se quiere poder parsear:

```
Infraestructura ProyectoPrueba{
        tipo = AWS ;
        accessId = "AwsY67HjkXcvbBoP";
        secret = "AOP67nawyXz/8sjj)8s";
        ambientes = [
                {
                        nombre = "dev";
                        vpcs =[
                                {
                                        tipo = VPC;
                                        id = vpcdesarrollo;
                                        CIDR = "172.34.0.0/16";
                                }
                        ]
                        securityresource = [
```

```
{
        tipo = InternetGateway;
        id = gatewaydesarrollo;
        vpc = vpcdesarrollo;
},
{
        tipo = GrupoSeguridad;
        id = gseguridaddesarrollo;
        vpc = vpcdesarrollo;
        reglas = [
                {
                        tipo = HTTP;
                        protocolo = TCP;
                        puerto = 80;
                        descripcion = "HTTP entrada ";
                        origen = "0.0.0.0/0";
                        direccion = ENTRADA;

                },
                {
                        tipo = HTTP;
                        protocolo = TCP;
                        puerto = 80;
                        descripcion = "HTTP salida ";
                        origen = "0.0.0.0/0";
                        direccion = SALIDA;

                },
                {
                        tipo = DATABASE;
                        protocolo = TCP;
                        puerto = 5436;
                        descripcion = "PostgreSQL ";
                        origen = "0.0.0.0/0";
                        direccion = ENTRADA;

                }
                ,
                {
                        tipo = DATABASE;
                        protocolo = TCP;
                        puerto = 5436;
                        descripcion = "PostgreSQL ";
```

```
                                origen = "0.0.0.0/0";
                                direccion = SALIDA;

                        }
                ]
        },
        {
                tipo = Subred;
                id = subred1;
                CIDR = "172.34.0.0/24";
                zonaDisponibilidad=useast1;
                vpc=vpcdesarrollo;
        },
        {
                tipo = Subred;
                id = subred1;
                CIDR = "172.34.1.0/24";
                zonaDisponibilidad=useast1;
                vpc=vpcdesarrollo;
        },
        {
                tipo = Subred;
                id = subred1;
                CIDR = "172.34.2.0/24";
                zonaDisponibilidad=useast1;
                vpc=vpcdesarrollo;
        }
]
recursos =[
        {
                tipo = Server;
                id = servidoraplicacion;
                dimension = MICRO ;
                os = LINUX;
                vpc = vpcdesarrollo;

        },
        {
                tipo = Database;
                id = dbdesarrollo;
                dimension = MICRO ;
                tipo = POSTGRESQL;
                vpc = vpcdesarrollo;
```

```
                        },
                        {
                                tipo = Almacenamiento;
                                id = almacenamientodesarrollo;
                                dimension = MICRO;
                                gigas = 20.0;
                                vpc = vpcdesarrollo;

                        }
                ]
                conexiones = [
                        {
                                recurso1 = servidoraplicacion;
                                recurso2 = dbdesarrollo;

                        },
                        {
                                recurso1 = servidoraplicacion;
                                recurso2 = almacenamientodesarrollo;
                        }
                ]
        }
    ]
}
```

En general, los principales bloques constructores de la especificación son:

```
Infraestructura nombreProyecto {
        proveedor = TipoProveedo;
        accessId =”llave de acceso” ;
        secret=”secreto de acceso”;
        Ambientes = [
                nombre=nombreAmbiente/idambiente
                vpc =[
                        // La lista de VPC
                ]
                securityresource = [
                        //Lista de recursos de seguridad
                ]
                recursos=[
```

```
            // Lista de recursos
        ]
        conexiones = [
            //Lista de conexiones entre servidores
        ]


    ]
}
```

Para tener en cuenta:

- La definición de […] indica que hay una lista y que en las listas no se debe poner el carácter de finalización **(;)**
- La definición de { … } indica que es un elemento
- Los atributos se definen con nombre **=** valor y el carácter de finalización (**;)**
- Las conexiones referencian elementos de tipo Recurso (Servidores, Bases de datos y Almacenamientos).
- Los atributos **vpc** hacen referencia a una VPC previamente creada
- Los atributos **tipo** pueden tomar valores soportados por el proveedor
- Los atributos **dimension** pueden tomar valores **micro, small, medium o large**