



MEMORIA DE PRÁCTICAS

DESARROLLO DE PORTAL AUTOSERVICIO DE PROVISION AUTOMATIZADA DE IAAS Y PAAS EN AZURE UTILIZANDO ANSIBLE COMO HERRAMIENTA DE CONFIGURACIÓN

Alberto Ruiz Andrés

Índice

1. Introducción:	3
2. Diagrama de la arquitectura:	3
3. Portal Web	4
3.1. Recogida de datos	4
3.1.1. Iaas	4
3.1.2. Paas	4
3.2. Funcionamiento del portal web:	5
3.3. Creación del archivo generador:	7
3.3.1. Prerrequisitos:	7
3.3.2. Creación del archivo Linux:	9
3.3.3. Creación del archivo Windows:	11
3.3.4. Creación del archivo PostgreSQL:	13
3.3.5. Creación del archivo MySQL:	14
4. Ejecuciones:	15
4.1. Prerrequisitos de las máquinas virtuales	15
4.2. Creación de máquinas Linux	16
4.3. Creación de máquinas Windows	17
4.4. Creación de servidores PostgreSQL	18
4.5. Creación de servidores MySQL	19
5. Entrega:	20

1. Introducción:

El propósito de las prácticas era comprender como funciona Ansible y Azure, la nube de Microsoft, además de la creación de un portal Web, mediante el cuál fuese posible comunicar las preferencias de los usuarios con Azure, y así poder desplegar servicios tanto Iaas (Infraestructure as a Service) como Paas (Platform as a Service).

Para ello se ha usado una suscripción gratuita dentro de Azure, con la cual algunas acciones estaban restringidas pero ha sido posible desarrollar todo lo esperado. Para ello se han usado los lenguajes: *HTML*, *JavaScript*, *PHP* y *Ansible*. Además se ha trabajado con la terminal de Azure, la cual usa comandos *Unix* además de sus comandos propios.

El desarrollo del portal Web, se ha realizado mediante *WAMP*, un servidor local cuyo funcionamiento permite lanzar el desarrollo del portal además de poder hacer los envíos de datos y la creación de los archivo generadores. Mediante este servidor se hacía la conexión *SCP* y *SSH* necesarias para trabajar con los archivos en la cuenta de Azure.

2. Diagrama de la arquitectura:

El siguiente diagrama muestra la arquitectura en la que se basa el diseño del portal web con respecto a la nube de Azure y el uso intermedio de Ansible, como generador de los archivos ejecutables para la creación de tanto productos Iaas como Paas.

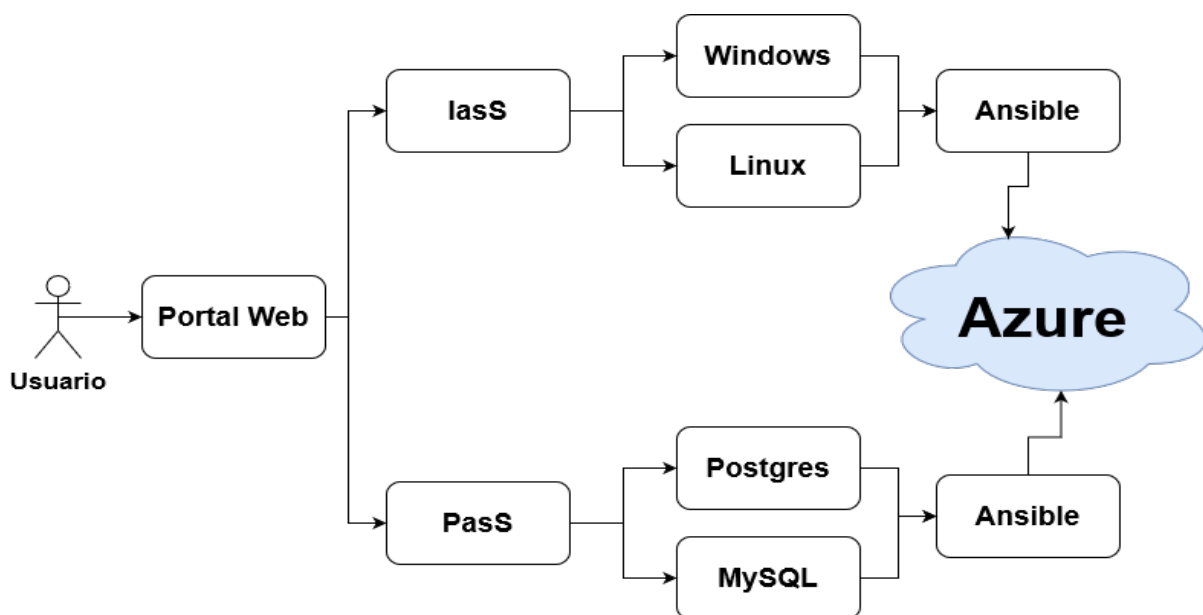


Figura 1: Diagrama de la arquitectura del proyecto

3. Portal Web

Para la realización del portal Web, se ha usado *PHP*, *HTML* y *JavaScript*, cuya funcionalidad era la de conectar al usuario con los servicios de Azure, pudiendo desplegar tanto máquinas virtuales como servicios de base de datos. La idea principal no era diseñar un simple “esqueleto”, sino que también era darle un acabado bonito; para ello se ha usado **Bootstrap**, una biblioteca multiplataforma de código abierto cuya funcionalidad es la de diseño web [3], además se ha trabajado con la gestión de eventos con **AJAX**, una técnica de desarrollo web para crear aplicaciones interactivas, estas se ejecutan en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. [1]

3.1. Recogida de datos

Se ha creado un formulario para la elección de los productos que se ofrecen, tanto en Iaas como en Paas es necesario rellenar los campos para que estos sean los que modifiquen los archivos “maqueta”, que serán enviados y ejecutados en Azure.

3.1.1. Iaas

Su funcionalidad es la creación de máquinas virtuales de diferentes sistemas operativos, los datos que se deben recoger para esta opción son los siguientes:

1. Nombre de la máquina virtual
2. Sistema operativo
 - a) Linux
 - b) Windows
3. Número de máquinas virtuales (limitado a dos por suscripción)

3.1.2. Paas

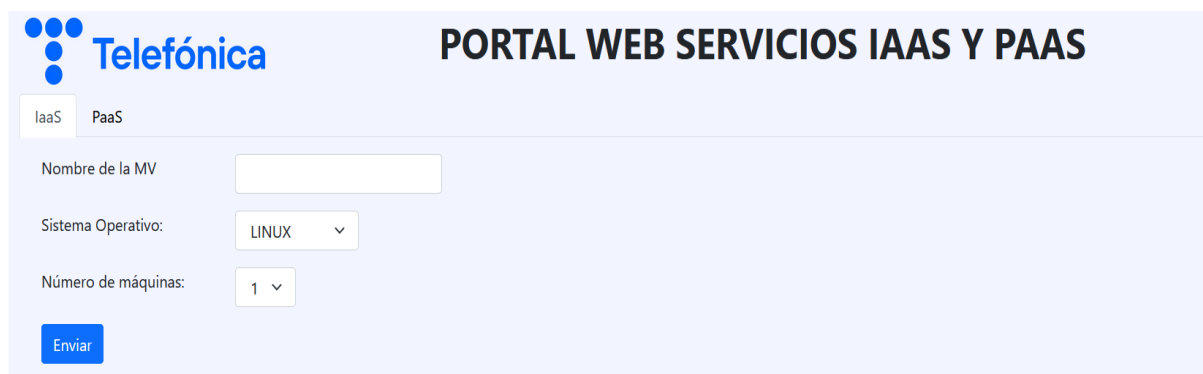
Su funcionalidad es la creación de servidores de bases de datos con una instancia de base de datos creada, para ello los datos que se deben recoger son los siguientes:

1. Nombre del servidor
2. Nombre de la base de datos
3. Tipo de servidor
 - a) PostgreSQL
 - b) MySQL

3.2. Funcionamiento del portal web:

En la siguiente secuencia de imágenes se observa cual es el aspecto final del portal y como se usa. Las capturas de pantalla de los posibles resultados están en la sección 4.

La primera captura de pantalla muestra como es el portal web sin introducir ningún dato, lo que se observa es el formulario IaaS.

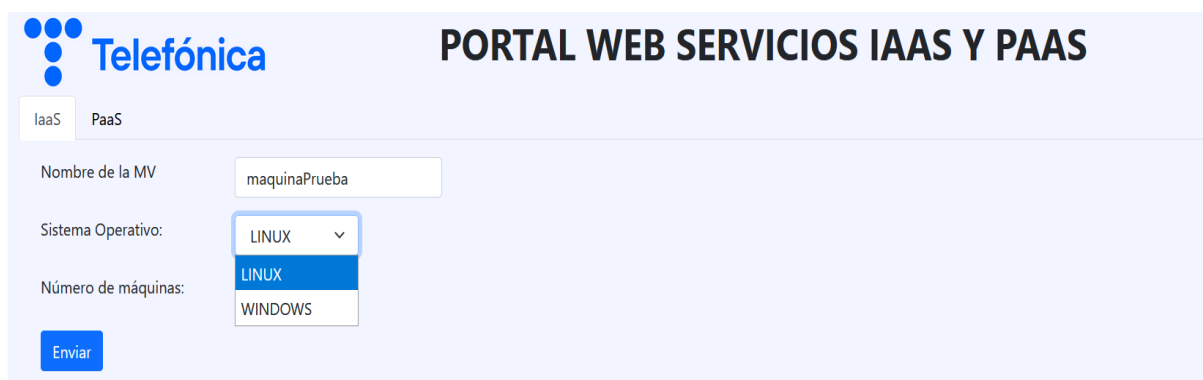


The screenshot shows the 'PORTAL WEB SERVICIOS IAAS Y PAAS' interface. The 'IaaS' tab is selected. The form contains the following fields:

- Nombre de la MV:
- Sistema Operativo:
- Número de máquinas:
- Enviar:

Figura 2: Visión general del portal

La siguiente captura de pantalla muestra la posible de elección de sistema operativo de la máquina virtual que se quiere desplegar.

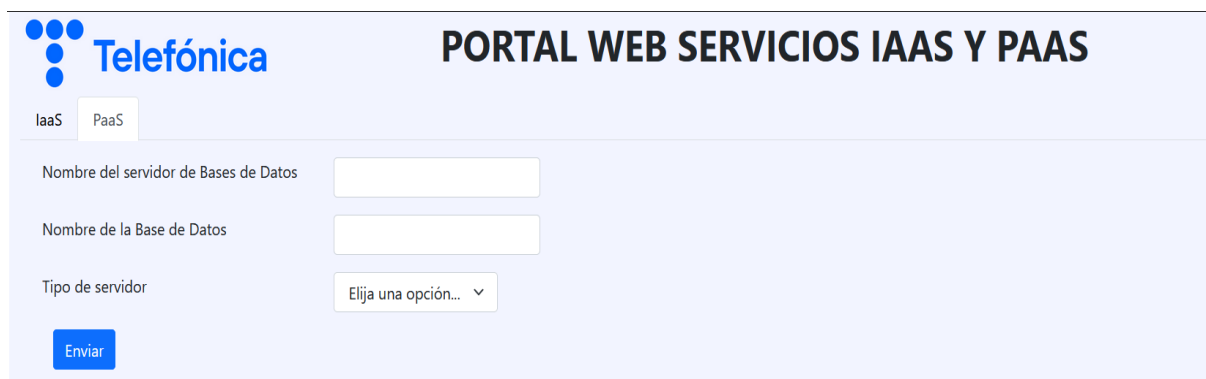


The screenshot shows the same 'PORTAL WEB SERVICIOS IAAS Y PAAS' interface. The 'IaaS' tab is selected. The form contains the following fields:

- Nombre de la MV:
- Sistema Operativo: (dropdown menu is open showing options: LINUX, WINDOWS)
- Número de máquinas:
- Enviar:

Figura 3: Elección del sistema operativo de una máquina virtual

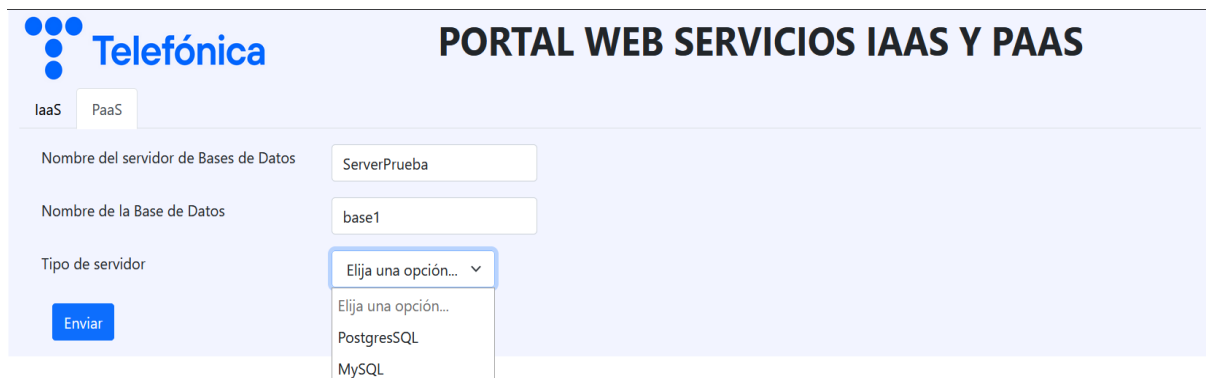
La siguiente imagen muestra el formulario de los servicios Paas, sin introducir ningún dato



The screenshot shows the 'PORTAL WEB SERVICIOS IAAS Y PAAS' interface. At the top left is the Telefónica logo. Below it are two tabs: 'IaaS' and 'PaaS', with 'PaaS' being the active tab. The form contains three input fields: 'Nombre del servidor de Bases de Datos', 'Nombre de la Base de Datos', and 'Tipo de servidor'. The 'Tipo de servidor' field is a dropdown menu with the text 'Elija una opción...' and a downward arrow. A blue 'Enviar' button is located at the bottom left of the form area.

Figura 4: Visión general del formulario Paas

La última imagen muestra la posible elección en el formulario Paas del tipo de servidor de base de datos que se puede elegir.



This screenshot is similar to the previous one, but the 'Tipo de servidor' dropdown menu is open, showing three options: 'Elija una opción...' (highlighted), 'PostgreSQL', and 'MySQL'. The other fields and the 'Enviar' button remain the same.

Figura 5: Elección del tipo de servidor de base de datos

3.3. Creación del archivo generador:

Una vez se han recogido los datos, se va a crear el documento que servirá de *playbook*, para ejecutar con Ansible en el servidor de Azure. Para la creación de estos documentos me he basado en *playbooks* de la comunidad y una vez se han adaptado a las necesidades especificadas estaban listos para ser ejecutados.[5]

Tras generar estos archivo se envían mediante **SCP** al servidor de Azure, donde serán ejecutados como archivos Ansible. Los datos de la conexión **SCP** y posterior **SSH**, se encuentran en archivos diferentes a los archivos PHP que generan el portal, con esto se consigue que si su modificación es necesaria se haga de manera rápida y no se tenga que buscar dentro del código de la página.

Hay que tener en cuenta que estos archivos son *YAML*, este tipo de archivos son sensibles a los espacios, y retornaran un fallo si estos son incorrectos.

3.3.1. Prerrequisitos:

Para poder crear una máquina virtual previamente se tiene que haber creado tanto un grupo de recursos como un vnet y una subnet.

Si se quiere crear un grupo de recursos hay que tener en cuenta las localizaciones donde se pueden obtener servicios de Azure[7].

Se puede crear con un comando de Azure sencillo: [6]

```
az group create --name grupoRecursos --location localizacion
```

Otra opción es hacerlo mediante un playbook de Ansible:

```
---
```

```
- hosts: localhost
```

```
  tasks:
```

```
    - name: Create resource group
```

```
      azure_rm_resourcegroup:
```

```
        name: grupoRecursos
```

```
        location: localizacion
```

Aunque todos los comandos de Ansible se pueden crear con comandos de Azure, solo se va a explicar los playbooks de Ansible.

La primera parte del archivo es una manera de dar un nombre al playbook que se crea, además de indicar a que host va dirigido ya que una de las funcionalidades de Ansible que más llama la atención es que puede ejecutar sus scripts en varias máquinas a la vez, esto queda especificado en el apartado *hosts*.

En este caso se ejecuta sobre la propia máquina de Azure, pero si se quisiera configurar el mismo software sobre distintas máquinas sería recomendable crear un **inventario**. [2]

```
- name: CREATE VNET AND SUBNET
```

```
  hosts: localhost
```

```
  connection: local
```

```
  gather_facts: False
```

En la siguiente parte del archivo es donde se crean las tareas o *tasks*, cada una de ellas tiene asociada una funcionalidad, se suele indicar con una frase descriptiva en *name* aunque esto es opcional, se ejecutan de manera secuencial, en el caso de que una de ellas falle la ejecución para mostrando el fallo referente a la tarea fallida.

Justo después se indica el comando que se va a usar, cada uno tiene distintos campos a los cuales se les dan valor según la tarea que se requiere.

En este caso lo que se hace en la primera tarea es crear una vnet, para ello se le da un nombre, se indica en que grupo de recursos se va a crear y es necesario indicar los rangos de direcciones IPv4 usando notación CIDR.

La segunda tarea es la creación de una subnet, para ello es necesario tener una vnet ya creada, en este se indica el grupo de recursos, el nombre de la vnet sobre la que se está creando y el rango de direcciones IPv4 usando notación CIDR.

```
tasks:
- name: Create a virtual network
  azure_rm_virtualnetwork:
    resource_group: grupoRecursos
    name: nombrevnet
    address_prefixes_cidr:
      - "10.1.0.0/16"
      - "172.100.0.0/16"

- name: Create a subnet
  azure_rm_subnet:
    resource_group: grupoRecursos
    virtual_network_name: nombrevnet
    name: nombresubnet
    address_prefix_cidr: "10.1.0.0/24"
```


3.3.2. Creación del archivo Linux:

En este caso la ejecución del archivo que he usado crea una maquina CentOS, una distribución Linux basada en la distribución Red Hat Enterprise Linux, operando de manera similar cuyo objetivo es ofrecer al usuario un software de “clase empresarial” gratuito. [4]

```
- name: CREATE VM PLAYBOOK
  hosts: localhost
  connection: local
  gather_facts: False
```

La siguiente tarea crea una cuenta de almacenamiento en esta se almacenarán todos los objetos de datos, blobs, archivos, colas, tablas y discos. Hay diferentes tipos y la elección depende de las características que se busque.

```
  tasks:
  - name: Create storage account
    azure_rm_storageaccount:
      resource_group: grupoRecursos
      name: nombrevm
      account_type: Standard_LRS
```

A continuación se crea un grupo de seguridad que permitirá conexiones tanto SSH como HTTP, en este caso para cada una de las conexiones se tienen que especificar que protocolo que se va a usar, que puerto es habilitado para la conexión, si se permite el acceso o se bloquea, que prioridad tiene respecto otras posibles conexiones y la dirección del trafico de conexiones.

```
- name: Create security group that allows SSH and HTTP
  azure_rm_securitygroup:
    resource_group: grupoRecursos
    name: nombrevm
    rules:
      - name: SSH
        protocol: Tcp
        destination_port_range: 22
        access: Allow
        priority: 101
        direction: Inbound
      - name: WEB
        protocol: Tcp
        destination_port_range: 80
        access: Allow
        priority: 102
        direction: Inbound
```

La siguiente tarea tiene como objetivo darle a la maquina virtual una ip pública sobre la que el usuario se pueda conectar.

```
- name: Create public IP address
  azure_rm_publicipaddress:
    resource_group: grupoRecursos
```

```
allocation_method: Static
name: nombrevm
domain_name_label: nombrevm
```

La penúltima tarea de este playbook tiene como funcionalidad la creación de una interfaz de red, los requisitos para poder realizar esta tarea es que se haya creado un grupo de recursos, una vnet y una subnet.

```
- name: Create NIC
  azure_rm_networkinterface:
    resource_group: grupoRecursos
    name: nombrevm
    virtual_network_name: nombrevnet
    subnet_name: nombresubnet
    public_ip_name: nombrevm
    security_group: nombrevm
```

Por último creamos la máquina virtual dando a los campos los valores necesarios. Muchos de los campos que se rellenan son nombres de recursos que se han creado anteriormente en este playbook.

En esta tarea, la etiqueta *vm_size* indica el tipo de máquina virtual que se va a crear, en este caso se indica una genérica como es la *Standard_D2s_v3*, es en esta **web** se explican las diferentes variantes que Azure tiene en catálogo.

Por último la etiqueta *image*, será la encargada de indicar el sistema operativo que estamos usando para crear la máquina virtual.

```
- name: Create VM
  azure_rm_virtualmachine:
    resource_group: grupoRecursos
    name: nombrevm
    storage_account: nombrevm
    storage_container: nombrevm
    storage_blob: nombrevm.vhd ## ALMACENAMIENTO DEL SO
    network_interfaces: nombrevm
    vm_size: Standard_D2s_v3
    admin_username: usuario
    admin_password: contraseña
    image:
      offer: CentOS
      publisher: OpenLogic
      sku: '7.2'
      version: latest"
```

3.3.3. Creación del archivo Windows:

Para la creación del archivo Windows hay que tener en cuenta que únicamente deberemos cambiar una serie de características respecto al archivo que usamos para generar la máquina Linux.

Las tareas que se pueden reutilizar son las que tienen como funcionalidad:

1. Crear una cuenta de almacenamiento
2. Crear una IP pública
3. Crear una interfaz de red

El resto sufren cambios, en el caso de la creación del grupo de seguridad, se debe añadir una opción por la que se pueda realizar una conexión remota a la máquina mediante el comando **RCP**.

```
- name: Create security group that allows SSH and HTTP
```

```
  azure_rm_securitygroup:
```

```
    resource_group: grupoRecursos
```

```
    name: nombrevm
```

```
    rules:
```

```
      - name: 'allow_rdp'
```

```
        protocol: Tcp
```

```
        destination_port_range: 3389
```

```
        access: Allow
```

```
        priority: 101
```

```
        direction: Inbound
```

```
      - name: 'allow_web_traffic'
```

```
        protocol: Tcp
```

```
        destination_port_range:
```

```
          - 80
```

```
          - 443
```

```
        access: Allow
```

```
        priority: 102
```

```
        direction: Inbound
```

```
      - name: 'allow_powershell_remoting'
```

```
        protocol: Tcp
```

```
        destination_port_range:
```

```
          - 5985
```

```
          - 5986
```

```
        access: Allow
```

```
        priority: 103
```

```
      - name: SSH
```

```
        protocol: Tcp
```

```
        destination_port_range: 22
```

```
        access: Allow
```

```
        priority: 104
```

```
        direction: Inbound
```

Una vez se han ejecutado las tareas previas, se puede crear la máquina virtual. En este caso en la etiqueta *image* se indica que es una máquina Windows.

```
- name: provision new Azure virtual host
```

```
azure_rm_virtualmachine:
  admin_username: usuario
  admin_password: contraseña
  os_type: Windows
  resource_group: grupoRecursos
  name: nombrevm
  state: present
  vm_size: Standard_D2s_v3
  storage_account_name: nombrevm
  network_interfaces: nombrevm
  image:
    offer: WindowsServer
    publisher: MicrosoftWindowsServer
    sku: 2016-Datacenter
    version: latest
```

Por último para la creación de una máquina Windows, es necesario configurarla para que se pueda acceder a los comandos de la shell mediante la herramienta *WinRM*, para ello es la siguiente tarea.

En ella hay una variable que está indicada con el nombre de *WINRMVALUE*, cuyo valor es la codificación en Base64 del siguiente comando:

```
Invoke-Expression -Command ((New-Object System.Net.WebClient).DownloadString
('https://raw.githubusercontent.com/ansible/ansible/devel/examples/scripts/
ConfigureRemotingForAnsible.ps1')); Enable-WSManCredSSP -Role Server -Force
```

Una vez tenemos este valor, se puede completar los campos de la tarea.

```
- name: create Azure vm extension to enable HTTPS WinRM listener
  azure_rm_virtualmachine_extension:
    name: winrm-extension
    resource_group: grupoRecursos
    virtual_machine_name: nombrevm
    publisher: Microsoft.Compute
    virtual_machine_extension_type: CustomScriptExtension
    type_handler_version: 1.9
    settings: '{"commandToExecute": "powershell.exe -ExecutionPolicy ByPass
-EncodedCommand WINRMVALUE"}'
    auto_upgrade_minor_version: true
  with_items: output.instances
```

3.3.4. Creación del archivo PostgreSQL:

En el caso de que se quiera crear un Servidor PostgreSQL, se necesita como prerequisites tener un grupo de recursos. (Sección 3.3.1)

En este playbook como en los anteriores se empieza indicando los hosts sobre los que tiene que actuar. Una vez se tiene indicando este campo comienza la creación del servidor. Para ello se indica el grupo de recursos donde va a almacenar este servidor, el nombre que le damos y el nombre del mismo.

En la etiqueta *sku* se indican las características de almacenamiento del servidor. La etiqueta *name* indica el tipo de servidor que es, hay un catálogo de servidores para elegir el que más convenga, no todos funcionan en todas las localizaciones. La etiqueta *tier*, indica el nivel del servidor pudiendo tener los valores *Basic* o *Standard*. Por último la etiqueta *capacity*, que indicará la capacidad de escalado del servidor.

El resto de etiquetas indican, localización donde se alojará el servidor, permiso SSL, usuario, contraseña y tamaño de almacenamiento.

```
---
- hosts: localhost
  tasks:

    - name: Create PostgreSQL Server
      azure_rm_postgresqlserver:
        resource_group: grupoRecursos
        name: nombreServidor
        sku:
          name: B_Gen5_1
          tier: Basic
          capacity: 1
        location: location
        enforce_ssl: True
        admin_username: usuario
        admin_password: contraseña
        storage_mb: 51200
```

La siguiente parte del archivo es una tarea para inicializar una base de datos dentro del servidor creado.

```
- name: Create instance of PostgreSQL Database
  azure_rm_postgresqldatabase:
    resource_group: grupoRecursos
    server_name: nombreServidor
    name: nombreBaseDatos
```

Esta última tarea es fundamental para poder acceder al servidor, sirve para crear un firewall y permitir la conexión sobre un rango de IPs, básicamente se indica sobre que servidor se actúa y los rangos de IPs permitidos, una vez sea ejecutado permitirá a los usuarios acceder, con las credenciales que se han usado para la configuración del servidor.

```
- name: Create (or update) PostgreSQL firewall rule
  azure_rm_postgresqlfirewallrule:
    resource_group: grupoRecursos
    server_name: nombreServidor
    name: rule1
```

```
start_ip_address: 0.0.0.0
end_ip_address: 255.255.255.255"
```

3.3.5. Creación del archivo MySQL:

Para este archivo YAML se usa la misma estructura que se ha usado anteriormente con el archivo PostgreSQL; es decir que primero se crea el servidor MySQL, posteriormente se crea la base de datos dentro del servidor y por último se configura el firewall que permitirá las conexiones a las IPs permitidas.

```
---
- hosts: localhost
  tasks:

    - name: Create MySQL Server
      azure_rm_mysqlserver:
        resource_group: grupoRecursos
        name: nombreServidor
        sku:
          name: GP_Gen5_2
          tier: GeneralPurpose
        location: location
        version: 5.6
        enforce_ssl: True
        admin_username: usuario
        admin_password: contraseña
        storage_mb: 51200

    - name: Create instance of MySQL Database
      azure_rm_mysqlatabase:
        resource_group: grupoRecursos
        server_name: nombreServidor
        name: nombreBaseDatos

    - name: Open firewall to access MySQL Server from outside
      azure_rm_resource:
        resource_group: grupoRecursos
        provider: dbformysql
        resource_type: servers
        resource_name: nombreServidor
        subresource:
          - type: firewallrules
            name: externalaccess
        body:
          properties:
            startIpAddress: 0.0.0.0
            endIpAddress: 255.255.255.255
```

4. Ejecuciones:

4.1. Prerrequisitos de las máquinas virtuales

El resultado de la ejecución correcta de la creación de vnet y subnet (sección 3.3.1) es: En el caso de que la

```

azureuser@gruporecursoslmvl:~/nuevaCarpeta$ ansible-playbook creacionNetSubnet.yml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'

PLAY [CREATE VNET AND SUBNET] *****

TASK [Create a virtual network] *****
changed: [localhost]

TASK [Create a subnet] *****
changed: [localhost]

PLAY RECAP *****
localhost                : ok=2    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

```

Figura 6: Ejecución correcta

ejecución fuera errónea, el fallo se mostraría de una manera similar a esta, el fallo se muestra en la etiqueta que está fallando.

```

azureuser@gruporecursoslmvl:~/nuevaCarpeta$ ansible-playbook creacionNetSubnet.yml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'

PLAY [CREATE VNET AND SUBNET] *****

TASK [Create a virtual network] *****
ok: [localhost]

TASK [Create a subnet] *****
fatal: [localhost]: FAILED! => {"changed": false, "msg": "Error creating or updating subnet subnetgr2_pr - Azure Error: ResourceNotFound\nMessage: The Resource 'Microsoft.Network/virtualNetworks/cosaInexistente' under resource group 'grupo_recursos2_westeurope' was not found. For more details please go to https://aka.ms/ARMResourceNotFoundFix"}

PLAY RECAP *****
localhost                : ok=1    changed=0    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0

```

Figura 7: Ejecución Incorrecta

4.2. Creación de máquinas Linux

El resultado de la ejecución correcta de la creación de la máquina Linux (sección 3.3.2) es:

```
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that
the implicit localhost does not match 'all'

PLAY [CREATE VM PLAYBOOK] *****

TASK [Create storage account] *****
changed: [localhost]

TASK [Create security group that allows SSH and HTTP] *****
changed: [localhost]

TASK [Create public IP address] *****
changed: [localhost]

TASK [Create NIC] *****
[DEPRECATION WARNING]: Setting ip configuration flatten is deprecated and will
be removed. Using ip_configurations list to define the ip configuration. This
feature will be removed in version [2, 9]. Deprecation warnings can be disabled
by setting deprecation_warnings=False in ansible.cfg.
changed: [localhost]

TASK [Create VM] *****
[WARNING]: Module did not set no_log for ssh_password_enabled
changed: [localhost]

PLAY RECAP *****
localhost : ok=5  changed=5  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0
```

Figura 8: Ejecución correcta

En el caso de que la ejecución fuera errónea, el fallo se mostraría de una manera similar a esta, los fallos provienen de una mala ejecución de cualquiera de las tareas.

```
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'

PLAY [CREATE VM PLAYBOOK] *****

TASK [Create storage account] *****
ok: [localhost]

TASK [Create security group that allows SSH and HTTP] *****
ok: [localhost]

TASK [Create public IP address] *****
fatal: [localhost]: FAILED! => ("changed": false, "msg": "Error retrieving resource group grupo_inexistente - Resource group 'grupo_inexistente' could not be
found.")

PLAY RECAP *****
localhost : ok=2  changed=0  unreachable=0  failed=1  skipped=0  rescued=0  ignored=0
```

Figura 9: Ejecución Incorrecta

4.3. Creación de máquinas Windows

El resultado de la ejecución correcta de la creación de la máquina virtual Windows (sección 3.3.3) es:

```
PLAY [CREATE VM PLAYBOOK] *****
TASK [create Azure storage account] *****
ok: [localhost]

TASK [Create security group that allows SSH and HTTP] *****
changed: [localhost]

TASK [Create public IP address] *****
ok: [localhost]

TASK [Create NIC] *****
[DEPRECATION WARNING]: Setting ip_configuration flatten is deprecated and will be removed. Using ip_configurations list to define the ip configuration. This feature will be removed in version [2, 9]. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
ok: [localhost]

TASK [provision new Azure virtual host] *****
[WARNING]: Module did not set no_log for ssh_password_enabled
changed: [localhost]

TASK [create Azure vm extension to enable HTTPS WinRM listener] *****
changed: [localhost] => (item=output.instances)
[WARNING]: The value "1.9" (type float) was converted to "1.9" (type string). If this does not look like what you expect, quote the entire value to ensure it does not change.
[DEPRECATION WARNING]: The 'azure_rm_virtualmachines_extension' module has been renamed to 'azure_rm_virtualmachineextension'. This feature will be removed in version 2.12. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
PLAY RECAP *****
localhost : ok=6 changed=3 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
```

Figura 10: Ejecución correcta

En el caso de que la ejecución fuera errónea, el fallo se mostraría de una manera similar a esta, los fallos provienen de una mala ejecución de cualquiera de las tareas.

```
PLAY [CREATE VM PLAYBOOK] *****
TASK [create Azure storage account] *****
ok: [localhost]

TASK [Create security group that allows SSH and HTTP] *****
changed: [localhost]

TASK [Create public IP address] *****
changed: [localhost]

TASK [Create NIC] *****
[DEPRECATION WARNING]: Setting ip_configuration flatten is deprecated and will be removed. Using ip_configurations list to define the ip configuration. This feature will be removed in version [2, 9]. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
changed: [localhost]

TASK [provision new Azure virtual host] *****
[WARNING]: Module did not set no_log for ssh_password_enabled
fatal: [localhost]: FAILED! => ("changed": false, "msg": "Error creating or updating virtual machine maquinaWin0 - Azure Error: TargetDiskBlobAlreadyExists\nMessage: Blob https://maquinawin0.blob.core.windows.net/vhds/maquinaWin0.vhd already exists. Please provide a different blob URI as target for disk 'maquinaWin0.vhd'.")
PLAY RECAP *****
localhost : ok=4 changed=3 unreachable=0 failed=1 skipped=0 rescued=0 ignored=0
```

Figura 11: Ejecución Incorrecta

4.4. Creación de servidores PostgreSQL

El resultado de la ejecución correcta de la creación del servidor PostgreSQL (sección 3.3.4) es:

```
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that
the implicit localhost does not match 'all'

PLAY [localhost] *****

TASK [Gathering Facts] *****
ok: [localhost]

TASK [Create PostgreSQL Server] *****
changed: [localhost]

TASK [Create instance of PostgreSQL Database] *****
changed: [localhost]

TASK [Create (or update) PostgreSQL firewall rule] *****
changed: [localhost]

PLAY RECAP *****
localhost                : ok=4    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Figura 12: Ejecución correcta

En el caso de que la ejecución fuera errónea, el fallo se mostraría de una manera similar a esta, los fallos provienen de una mala ejecución de cualquiera de las tareas.

```
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that
the implicit localhost does not match 'all'

PLAY [localhost] *****

TASK [Gathering Facts] *****
ok: [localhost]

TASK [Create PostgreSQL Server] *****
fatal: [localhost]: FAILED! => {"changed": false, "msg": "Error creating the PostgreSQL Server instance: Azure Error: ServerNameAlreadyExists\n\nMessage: Specified server name 'server' is already used. Please use a different name and try again."}

PLAY RECAP *****
localhost                : ok=1    changed=0    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0
```

Figura 13: Ejecución Incorrecta

4.5. Creación de servidores MySQL

El resultado de la ejecución correcta de la creación del servidor MySQL (sección 3.3.5) es:

```
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that
the implicit localhost does not match 'all'

PLAY [localhost] *****
TASK [Gathering Facts] *****
ok: [localhost]

TASK [Create MySQL Server] *****
[WARNING]: The value "5.6" (type float) was converted to "'5.6'" (type string).
If this does not look like what you expect, quote the entire value to ensure it
does not change.
changed: [localhost]

TASK [Create instance of MySQL Database] *****
changed: [localhost]

TASK [Open firewall to access MySQL Server from outside] *****
[WARNING]: Azure API profile latest does not define an entry for
GenericRestClient
changed: [localhost]

PLAY RECAP *****
localhost : ok=4    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Figura 14: Ejecución correcta

En el caso de que la ejecución fuera errónea, el fallo se mostraría de una manera similar a esta, los fallos provienen de una mala ejecución de cualquiera de las tareas.

```
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'

PLAY [localhost] *****
TASK [Gathering Facts] *****
ok: [localhost]

TASK [Create MySQL Server] *****
[WARNING]: The value "5.6" (type float) was converted to "'5.6'" (type string). If this does not look like what you expect, quote the entire value to ensure
it does not change.
fatal: [localhost]: FAILED! => ("changed": false, "msg": "Error retrieving resource group None - Parameter 'resource_group_name' can not be None.")

PLAY RECAP *****
localhost : ok=1    changed=0    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0
```

Figura 15: Ejecución Incorrecta

5. Entrega:

Todo lo explicado se resume en un directorio de Github propio, donde he creado un repositorio el cual almacena todos los documentos que he utilizado. El link al repositorio es el siguiente:

<https://github.com/albruiz/practicansibleazure>

Referencias

- [1] *AJAX* - *Wikipedia*. 2020. URL: <https://es.wikipedia.org/wiki/AJAX>.
- [2] Lorenzo Atareao. *El inventario de Ansible*. 2020. URL: <https://atareao.es/tutorial/ansible/el-inventario-de-ansible/>.
- [3] *Bootstrap* - *Wikipedia*. 2021. URL: [https://es.wikipedia.org/wiki/Bootstrap_\(framework\)](https://es.wikipedia.org/wiki/Bootstrap_(framework)).
- [4] *CentOS* - *Wikipedia*. 2021. URL: <https://es.wikipedia.org/wiki/CentOS>.
- [5] *Github* - *community Playbooks*. 2021. URL: <https://github.com/Azure-Samples/ansible-playbooks>.
- [6] *Github* - *tutorial Ansible-Azure*. 2021. URL: https://github.com/MicrosoftLearning/AZ400-DesigningandImplementingMicrosoftDevOpsSolutions/blob/master/Instructions/Labs/AZ400_M14_Ansible_with_Azure.md.
- [7] Bradley Schacht. *Azure PowerShell – List Data Center Locations*. 2019. URL: <http://www.bradleyschacht.com/azure-powershell-list-data-center-locations/>.