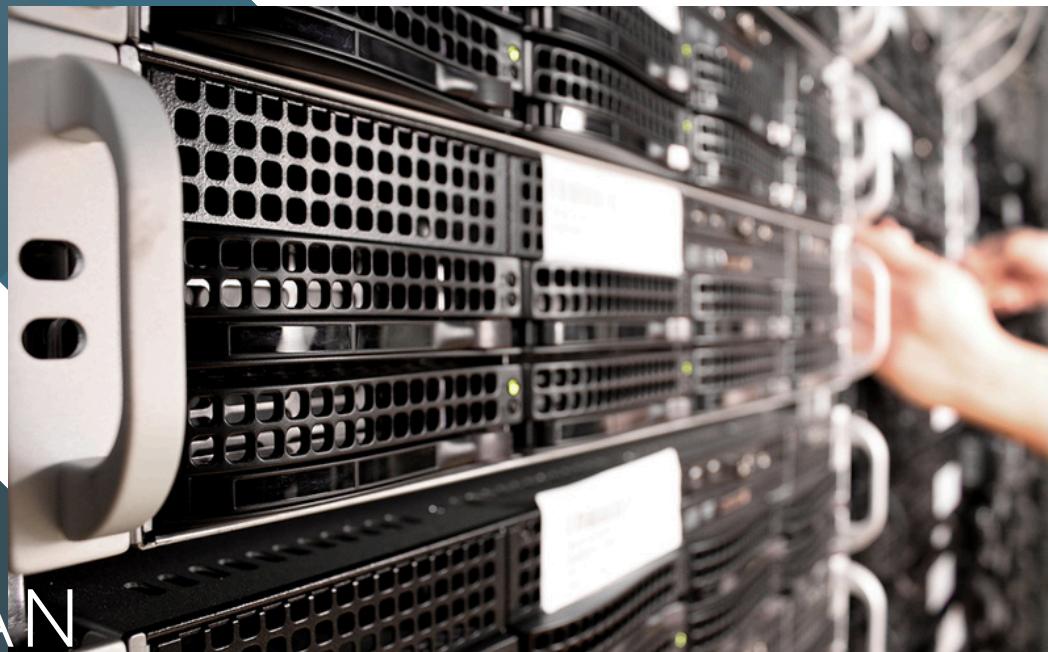


3A. EVALUACIÓN

Framework de adopción de Infraestructura
en la nube



ERICK IVÁN
CHÁVEZ
MONROY.

23/03/2025

CONTENIDO

Introducción	2
Implementación de la Máquina Virtual en Azure	3
Configuración de la Máquina Virtual	3
Versionamiento de la Infraestructura en la Nube.....	3
Scripts de Programación	5
Registro de Colaboración	9
El proceso se organizó de la siguiente manera:	9
Mapa de Recursos.....	10

INTRODUCCIÓN

La administración de infraestructuras en la nube ha evolucionado significativamente en los últimos años, pasando de configuraciones manuales a enfoques automatizados que garantizan mayor eficiencia y precisión. En este contexto, la Infraestructura como Código (IaC) surge como una metodología clave que permite definir y gestionar recursos mediante código, asegurando la consistencia y escalabilidad de los entornos tecnológicos.

Este proyecto explora la implementación de una máquina virtual en Microsoft Azure utilizando Azure Resource Manager (ARM). A través del uso de plantillas ARM en formato JSON, se logra una configuración automatizada y reproducible, reduciendo errores y facilitando la gestión de recursos en la nube.



A lo largo del documento, se abordará el proceso de implementación de la infraestructura, destacando aspectos fundamentales como el versionamiento del código, la utilización de scripts para la automatización de tareas y el registro de colaboración dentro del equipo. Asimismo, se presentará un esquema detallado de los recursos desplegados, proporcionando una visión clara de la arquitectura implementada.

IMPLEMENTACIÓN DE LA MÁQUINA VIRTUAL EN AZURE

La implementación de una máquina virtual en Microsoft Azure utilizando Infraestructura como Código (IaC) permite la creación y configuración de recursos de manera automatizada, asegurando coherencia y eficiencia en la gestión de la infraestructura en la nube.

Para este proyecto, se utilizó Azure Resource Manager (ARM), una herramienta que permite definir los recursos en un formato declarativo mediante plantillas JSON o Bicep. A continuación, se detallan los pasos seguidos en la implementación de la máquina virtual, desde la descripción general del proceso hasta la configuración específica de la instancia creada.

CONFIGURACIÓN DE LA MÁQUINA VIRTUAL

- **Nombre de la VM:** ErickVM
- **Tipo de instancia:** B1s (750 horas/mes gratis en el nivel gratuito de Azure)
- **Sistema Operativo:** Ubuntu Server 20.04 LTS
- **Ubicación:** East US
- **Disco:** 30GB SSD
- **Grupo de recursos:** GR-Erick
- **Red Virtual:** vNet-Erick
- **Seguridad:** Regla de NSG para permitir SSH (puerto 22)

VERSIONAMIENTO DE LA INFRAESTRUCTURA EN LA NUBE

Para gestionar los cambios en la plantilla ARM y mantener un control ordenado del desarrollo de la infraestructura, se utilizó **GitHub como sistema de control de versiones**. Esto permitió almacenar de manera segura los archivos del proyecto, registrar cada modificación realizada y simular un entorno de trabajo colaborativo profesional.

El repositorio creado para este propósito se nombró **Azure-IaC-ErickVM**, y contiene la plantilla `template.json` junto con sus respectivas versiones y mejoras.

ESTRUCTURA DEL REPOSITORIO:

- **Repositorio Git:** Azure-IaC-ErickVM
- **Archivo principal:** `template.json`

ESTRATEGIA DE RAMAS UTILIZADA:

- `main`: rama principal con versiones estables del despliegue.

- **develop**: rama intermedia donde se prueban las integraciones antes de pasar a producción.
- **feature/ip-publica**: rama utilizada para probar la adición de dirección IP pública y otros ajustes.

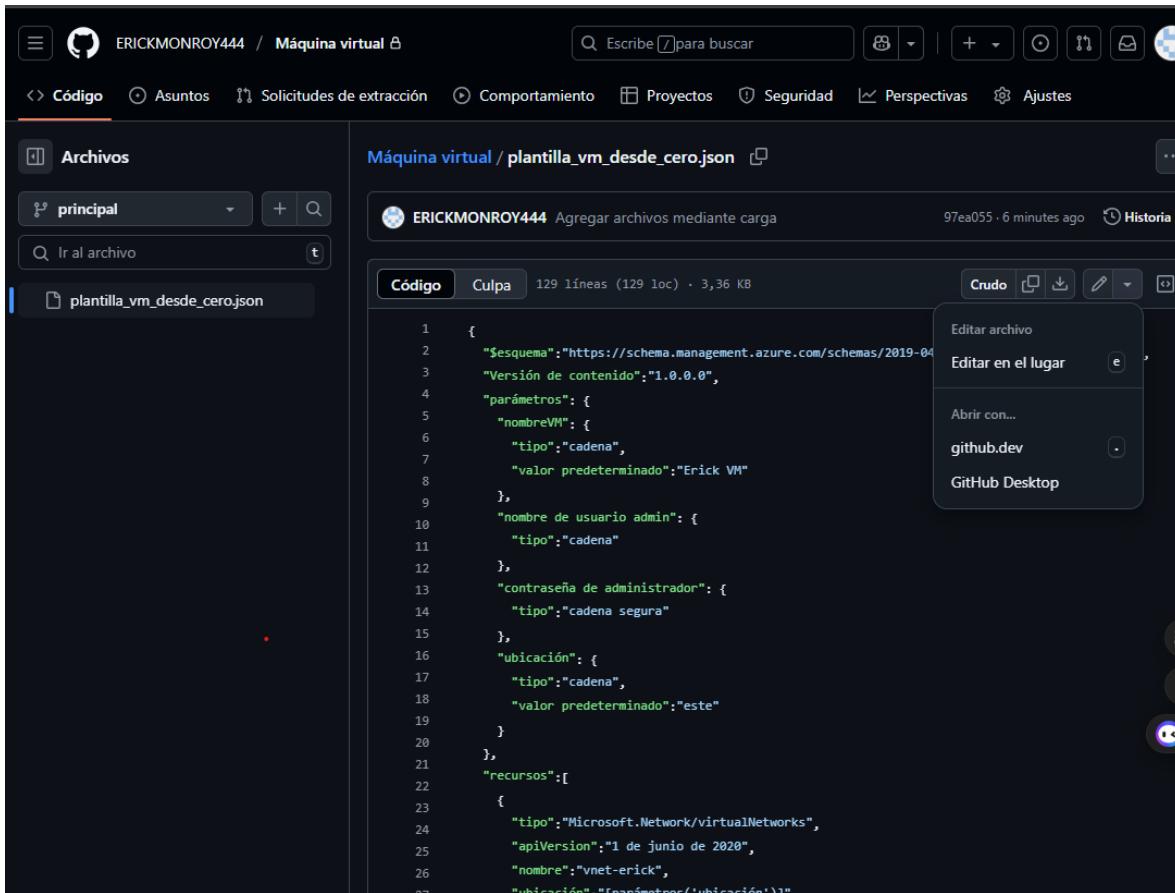
ETIQUETAS (TAGS) UTILIZADAS PARA MARCAR VERSIONES:

- **v1.0**: Implementación inicial de la máquina virtual.
- **v1.1**: Corrección de imagen del sistema operativo (de Ubuntu 20.04 a 18.04).
- **v1.2**: Inclusión de recursos adicionales como red virtual, NSG e IP pública.

EVIDENCIA DEL VERSIONAMIENTO:

A continuación, se presentan capturas de pantalla del repositorio:

- Vista general del repositorio, mostrando el archivo `template.json` y la estructura del proyecto.
- Historial de commits, donde se registran los cambios realizados a lo largo del desarrollo.
- Lista de etiquetas (`tags`) que representan versiones específicas del proyecto.
- Visualización de ramas, destacando la estrategia de desarrollo adoptada.



The screenshot shows a GitHub repository interface. The top navigation bar includes links for Código, Asuntos, Solicitudes de extracción, Comportamiento, Proyectos, Seguridad, Perspectivas, Ajustes, and a search bar. The main area displays a file named "Máquina virtual / plantilla_vm_desde_cero.json". The file content is as follows:

```

1  {
2      "$esquema": "https://schema.management.azure.com/schemas/2019-04-01/Microsoft.Resources/resourceDeploymentProperties.json#",
3      "versión de contenido": "1.0.0.0",
4      "parámetros": {
5          "nombreVM": {
6              "tipo": "cadena",
7              "valor predeterminado": "Erick VM"
8          },
9          "nombre de usuario admin": {
10             "tipo": "cadena"
11         },
12         "contraseña de administrador": {
13             "tipo": "cadena segura"
14         },
15         "ubicación": {
16             "tipo": "cadena",
17             "valor predeterminado": "este"
18         }
19     },
20     "recursos": [
21         {
22             "tipo": "Microsoft.Network/virtualNetworks",
23             "apiVersion": "1 de junio de 2020",
24             "nombre": "vnet-erick",
25             "ubicación": "[parámetros('ubicación')]"
26         }
27     ]
}

```

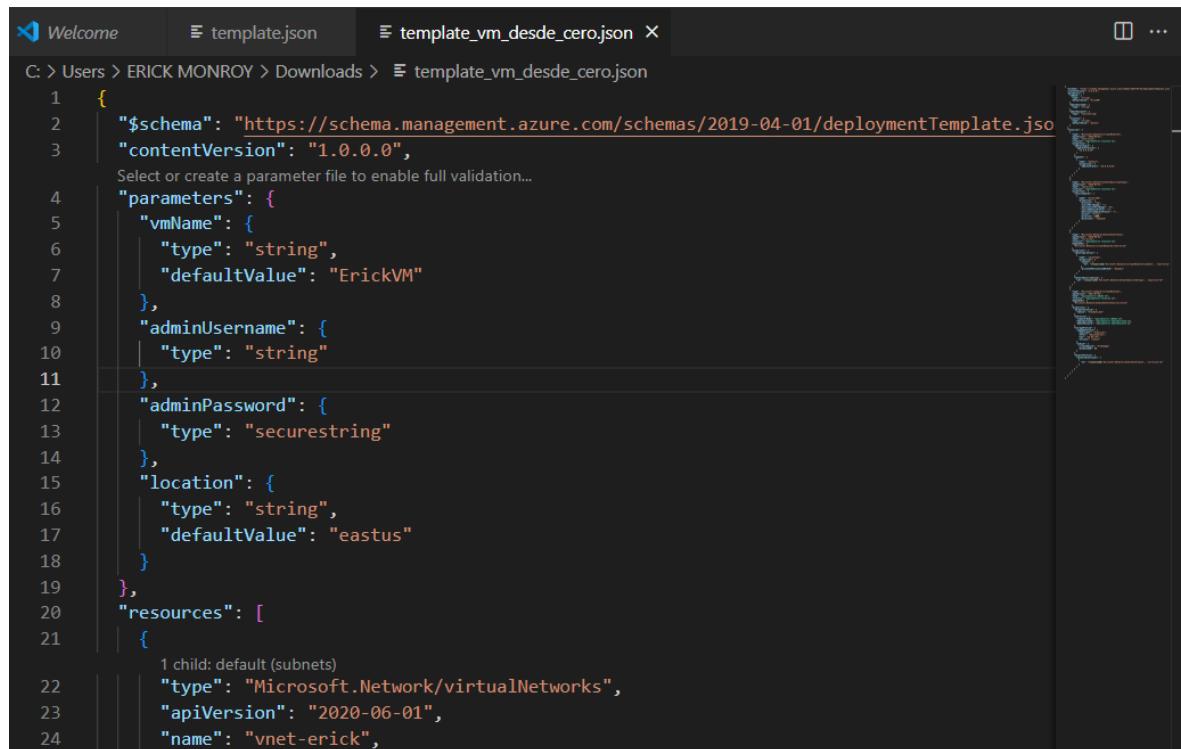
A context menu is open over the file content, showing options: Crudo, Editar archivo, Editar en el lugar, Abrir con..., github.dev, and GitHub Desktop.

El uso de GitHub permitió llevar un control preciso del avance del proyecto, identificar cambios importantes en el tiempo y facilitar la recuperación o comparación de versiones anteriores. Esta práctica es fundamental en entornos reales de DevOps e Infraestructura como Código (IaC), donde la trazabilidad y la automatización son claves.

SCRIPTS DE PROGRAMACIÓN

Se utilizaron scripts ARM en formato JSON para automatizar la creación de la infraestructura. Estos scripts definen los parámetros de la máquina virtual, red virtual, grupo de recursos, NSG, entre otros.

Ejemplo de fragmento del archivo `template.json`:

A screenshot of a code editor showing an ARM template JSON file. The file is titled "template.json" and has a tab for "template_vm_desde_cero.json". The code itself is a JSON object with several properties: "\$schema", "contentVersion", "parameters" (containing "vmName" and "adminUsername" objects), "adminPassword" (an object with type "securestring"), "location" (an object with type "string" and defaultValue "eastus"), and "resources" (an array containing one object for a virtual network).

```
1 {  
2     "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#"  
3     "contentVersion": "1.0.0.0",  
4     "parameters": {  
5         "vmName": {  
6             "type": "string",  
7             "defaultValue": "ErickVM"  
8         },  
9         "adminUsername": {  
10            "type": "string"  
11        },  
12         "adminPassword": {  
13            "type": "securestring"  
14        },  
15         "location": {  
16            "type": "string",  
17            "defaultValue": "eastus"  
18        }  
19    },  
20    "resources": [  
21        {  
22            "type": "Microsoft.Network/virtualNetworks",  
23            "apiVersion": "2020-06-01",  
24            "name": "vnet-erick",  
25            "properties": {  
26                "addressSpace": {  
27                    "addressPrefixes": ["10.0.0.0/16"],  
28                    "prefixLength": 16  
29                },  
30                "subnets": [  
31                    {  
32                        "name": "sub1",  
33                        "cidr": "10.0.0.0/24",  
34                        "prefixLength": 24  
35                    }  
36                ]  
37            }  
38        }  
39    ]  
40}
```

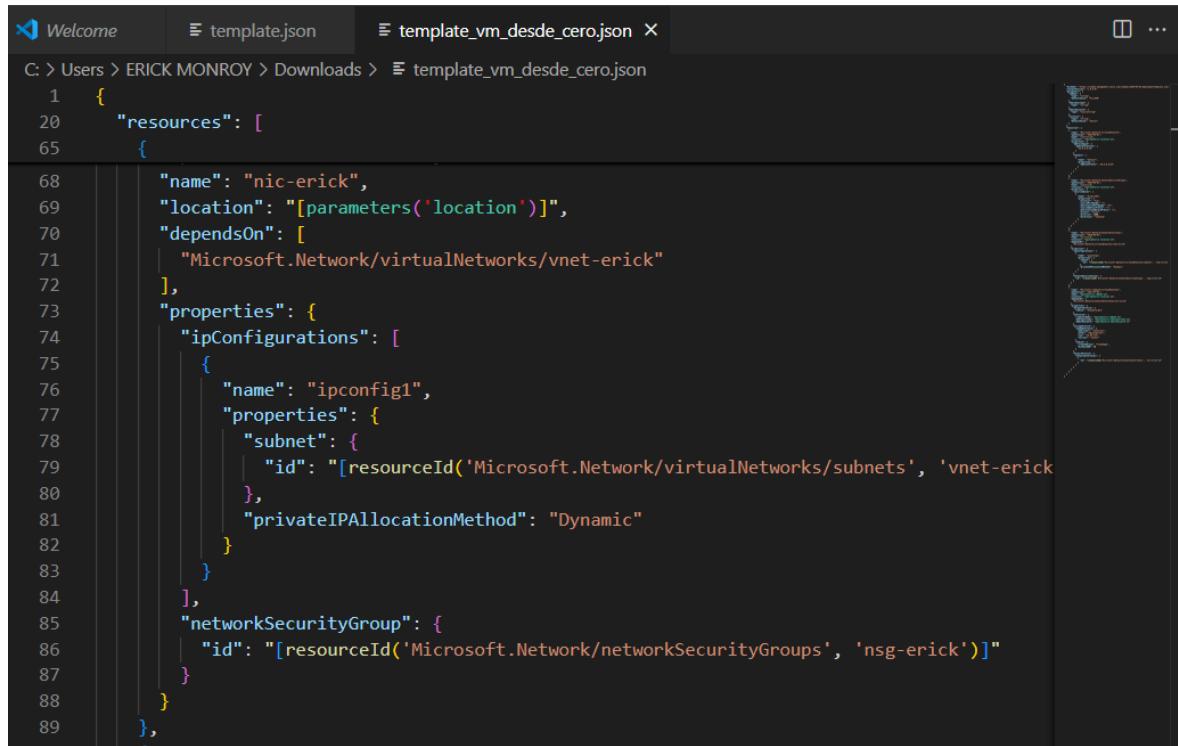
A screenshot of a code editor showing a JSON template file. The file contains configuration for a virtual network. It includes a resources section with a network security group (NSG) and a virtual network (VNet). The NSG has a single rule allowing SSH traffic. The VNet has one subnet with a specific address prefix.

```
C: > Users > ERICK MONROY > Downloads > template_vm_desde_cero.json
1  {
20   "resources": [
21     {
25       "location": "[parameters('location')]",
26       "properties": {
27         "addressSpace": {
28           "addressPrefixes": [
29             "10.0.0.0/16"
30           ]
31         },
32         "subnets": [
33           {
34             "name": "default",
35             "properties": {
36               "addressPrefix": "10.0.0.0/24"
37             }
38           }
39         ]
40       }
41     },
42     {
43       "type": "Microsoft.Network/networkSecurityGroups",
44       "apiVersion": "2020-06-01",
45       "name": "nsg-erick",
46       "location": "[parameters('location')]"

```

A screenshot of a code editor showing a JSON template file. The file contains configuration for a network interface. It includes a resources section with a network interface (NIC) and a security group (NSG). The NIC is associated with a specific virtual machine (VM).

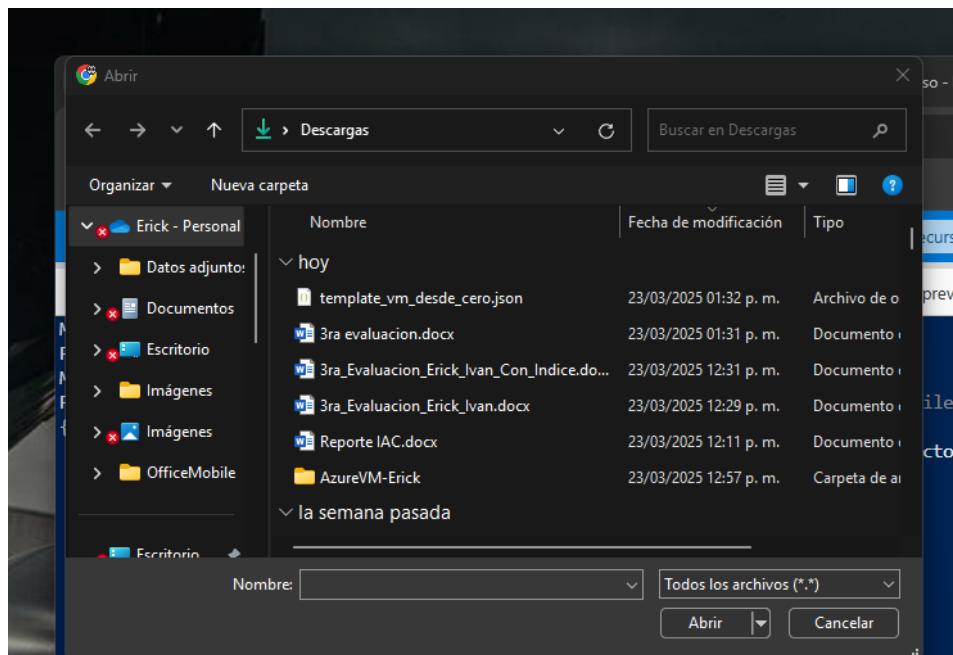
```
C: > Users > ERICK MONROY > Downloads > template_vm_desde_cero.json
1  {
20   "resources": [
21     {
26       "location": "[parameters('location')]",
27       "properties": {
28         "securityRules": [
29           {
30             "name": "Allow-SSH",
31             "properties": {
32               "protocol": "Tcp",
33               "sourcePortRange": "*",
34               "destinationPortRange": "22",
35               "sourceAddressPrefix": "*",
36               "destinationAddressPrefix": "*",
37               "access": "Allow",
38               "priority": 1000,
39               "direction": "Inbound"
40             }
41           }
42         ]
43       }
44     },
45     {
46       "type": "Microsoft.Network/networkInterfaces",
47       "apiVersion": "2020-06-01",
48       "name": "nic-erick",
49       "location": "[parameters('location')]"
50       "dependsOn": [
51         "[resourceId('Microsoft.Network/networkSecurityGroups/nsg-erick', parameters('location'))]"
52       ],
53       "properties": {
54         "ipConfigurations": [
55           {
56             "name": "ipconfig-erick",
57             "properties": {
58               "subnet": "[resourceId('Microsoft.Network/virtualNetworks/vnet-erick', parameters('location'))/subnets/default]"
59             }
60           }
61         ]
62       }
63     }
64   ],
65   "outputs": {}
66 }
```



The screenshot shows the Visual Studio Code interface with a dark theme. In the center, there is a code editor window displaying an Azure ARM template (JSON) file. The file contains code for creating a network interface card (NIC) named 'nic-erick' within a virtual network 'vnet-erick'. It specifies an IP configuration with a dynamic private IP allocation method, a subnet from 'nsg-erick', and a network security group. The code uses Azure Resource Manager syntax, including resource IDs and parameters. The code editor has syntax highlighting and line numbers. On the right side of the screen, there is a vertical panel showing a preview or another part of the template.

```
1  {
2    "resources": [
3      {
4        "name": "nic-erick",
5        "location": "[parameters('location')]",
6        "dependsOn": [
7          "Microsoft.Network/virtualNetworks/vnet-erick"
8        ],
9        "properties": {
10          "ipConfigurations": [
11            {
12              "name": "ipconfig1",
13              "properties": {
14                "subnet": {
15                  "id": "[resourceId('Microsoft.Network/virtualNetworks/subnets', 'vnet-erick')]"
16                },
17                "privateIPAllocationMethod": "Dynamic"
18              }
19            }
20          ],
21          "networkSecurityGroup": {
22            "id": "[resourceId('Microsoft.Network/networkSecurityGroups', 'nsg-erick')]"
23          }
24        }
25      }
26    ]
27  }
```

Una vez iniciado sesión en nuestra cuenta de Azure se procede a subir el archivo que se ha creado en **Visual Studio Code** como se muestra en la imagen:



Podemos comprobar que el archivo se ha subido exitosamente con el comando ls como se muestra a continuación:

```
PS /home/erick> ls
Microsoft template.json template_vm_desde_cero.json
```

Ahora para ejecutar la creación de la máquina virtual se procede a meter el siguiente comando:

```
PS /home/erick> az deployment group create --resource-group GR-Proyecto --template-file template_vm_desde_cero.json --parameters adminUsername=erickadmin adminPassword=Monroy2153+{
  "id": "/subscriptions/53fd1e10-fb9f-4011-96aa-0b7165c91216/resourceGroups/GR-Proyecto/providers/Microsoft.Resources/deployments/template_vm_desde_cero",
  "location": null,
```

Si todo ha salido bien nos dirigimos a este apartado en donde ya podemos apreciar la creación de la maquina virtual:

The screenshot shows the Microsoft Azure Infrastructure blade under the 'Máquinas virtuales' section. It displays a single VM named 'ErickVM' with the following details:

Nombre	Suscripción	Grupo de recursos	Ubicación	Estado	Sistema operativo	Tamaño	Dirección IP pública	Discos
ErickVM	Azure for Students	GR-Proyecto	East US	En ejecución	Linux	Standard_B1s	-	1

Podemos visualizar en el Grupo de Recueros (**GR-proyecto**) de lo que se ha creado:

The screenshot shows the Microsoft Azure Resource Groups blade for the 'GR-Proyecto' group. It displays the following resources:

Nombre	Tipo	Ubicación
ErickVM	Máquina virtual	East US
ErickVM_disk1_91c6635eb0a04f8fa14d833d99873ac7	Disco	East US
nic-erick	Interfaz de red	East US
nsg-erick	Grupo de seguridad de red	East US
vnet-erick	Red virtual	East US

Incluso podemos mediante PowerShell verificar el estatus de nuestra máquina virtual:

```
PS /home/erick> az vm get-instance-view --name ErickVM --resource-group GR-Proyecto --query "instanceView.statuses[?starts_with(code,'PowerState/')].displayStatus" --output tsv
VM running
PS /home/erick>
```

REGISTRO DE COLABORACIÓN

Aunque el desarrollo de este proyecto fue realizado de forma individual, se simuló un entorno colaborativo utilizando herramientas propias del trabajo en equipo profesional, como sistemas de control de versiones y flujos de integración continua.

EL PROCESO SE ORGANIZÓ DE LA SIGUIENTE MANERA:

Plataforma de colaboración: GitHub fue utilizado para alojar la plantilla ARM (template.json), gestionar versiones y simular revisión de cambios.

Control de versiones: Se trabajó con una estrategia de ramas (main, develop, feature/*) que permitió simular un entorno profesional y controlado.

Flujo de trabajo simulado:

Se realizaron pull requests desde ramas feature/* hacia develop para validar los cambios de manera controlada.

Se asumieron distintos roles durante el desarrollo:

- **Desarrollador de Infraestructura (IaC):** encargado de crear y depurar la plantilla ARM.
- **Validador:** responsable de ejecutar el despliegue y revisar errores en Cloud Shell.
- **Documentador:** encargado de capturar las evidencias del proceso y generar este informe técnico.

MAPA DE RECURSOS

Vista del grupo de recursos GR-Proyecto en el portal de Azure. En la imagen se muestran todos los componentes desplegados mediante la plantilla ARM, incluyendo la máquina virtual, red virtual, interfaz de red, disco y grupo de seguridad. Esta vista valida que la infraestructura fue creada correctamente y que los recursos están organizados de forma estructurada.

