Azure + Ansible

AUTOMATIZACIÓN DE TAREAS CON ANSIBLE Y ACCESO WEB REMOTO

Administración de Sistemas y Redes

Diego García, UO294255
Sergio García, UO294636

2024 - 2025

Escuela de Ingeniería Informática
Universidad de Oviedo

Índice

Automatización de la configuración de servidores de Minecraft y gestión de copias de seguridad mediante Ansible, con acceso remoto autenticado vía web 2
Automatización de creación de servidores Minecraft 3
Paso 1. Crear la plantilla base del servidor 3
Paso 2. Crear la estructura de Ansible 4
Paso 3. Configurar los archivos del sistema Ansible 5
Paso 4. Ejecución de Ansible 6
Automatización de backups 9
Paso 1. Crear la infraestructura 9
Paso 2. Modificar los ficheros de configuración 9
Paso 3. Ejecutar la copia de seguridad 10
Paso 4. Restaurar una copia de seguridad 11
Mejora mediante Scripts 13
Instalación de Azure CLI y autenticación 13
Script de creación de servers 14
Script de creación de backups 16
Script de restaurar un backup 17
Script de encendido y apagado del mundo 18
Autenticado y control de Servidores vía web. 20
Paso 1. Creación del proyecto. 20
Paso 2. Despliegue de la aplicación. 21

Automatización de la configuración de servidores de Minecraft y gestión de copias de seguridad mediante Ansible, con acceso remoto autenticado vía web

Esta segunda fase del proyecto constituye una evolución del trabajo previamente desarrollado, con el objetivo de mejorar la automatización, escalabilidad y facilidad de gestión de servidores de Minecraft. Los objetivos planteados en esta etapa son los siguientes:

- A partir de la infraestructura multiserver implementada anteriormente, se ha procedido a automatizar la creación y configuración de nuevos servidores mediante el uso de **Ansible**, una herramienta de automatización que permite definir despliegues reproducibles y eficientes.
- Se ha desarrollado un sistema de **copias de seguridad automatizadas y configurables**, también gestionado a través de Ansible, con el fin de garantizar la persistencia e integridad de los datos en caso de fallos o pérdidas.
- Con el propósito de mejorar la usabilidad del sistema, se han incorporado **scripts auxiliares** que facilitan la ejecución de tareas administrativas frecuentes, reduciendo así la intervención manual y minimizando errores.
- Asimismo, se ha diseñado e implementado una **interfaz web con autenticación** para usuarios administradores, que permite la gestión remota de los servidores, incluyendo operaciones como el encendido, apagado o supervisión del estado de estos.

Todas estas mejoras con el objetivo de facilitar la administración de servidores de Minecraft a través de herramientas modernas de automatización y control remoto, promoviendo buenas prácticas de mantenimiento y escalabilidad.

Lo primero que haremos será explicar que es Ansible y por que lo vamos a usar. **Ansible** es una herramienta de automatización de código abierto, que permite gestionar configuraciones y tareas en administración de sistemas informáticos de forma "remota y reproducible". Su funcionamiento se lleva a cabo a través de archivos ".yaml" (o ".yml"), a los que Ansible denomina como "*playbooks*", en los cuales se define los pasos que seguirá el proceso en cuestión que queramos automatizar. Alternativas a Ansible son **Puppet**, pero requiere de instalar agentes en cada nodo; **Chef**, muy parecido a Puppet, pero está basado en Ruby y es mucho más complejo; **SaltStak**, que esta sobre todo orientado a velocidad y escalabilidad; o incluso **Terraform**, que es una potente herramienta IaC (Infraestructura como Código) muy útil, pero que no tiene su enfoque en configuraciones software.

Para el desarrollo propuesto, hemos elegido Ansible debido a su facilidad de uso y su enfoque sin agentes, para hacer asequible a cualquier usuario interesado en el objetivo del proyecto, poder replicarlo sin requerir configuraciones complejas. Además, sabiendo que en casos regulares estaríamos trabajando con máquinas con recursos limitados, Ansible ofrece una buena potencia, permitiendo automatizar lo que queramos sin necesidad de instalar ningún software adicional en cada nodo, evitando la sobrecarga de trabajo y el consumo excesivo de recursos. También al funcionar con archivos YAML, facilita la legibilidad y favorece el mantenimiento. Sin duda Ansible es la herramienta que usaremos.

Automatización de creación de servidores Minecraft

Empezaremos con la automatización de creación de mundos. Para ello, arrancamos nuestra máquina virtual y nos conectamos a ella mediante *SSH* como vimos anteriormente. (Para este caso, es recomendado la creación de un archivo ".pem" para la conexión para algo que veremos próximamente).

```
PS C:\Users\diego> ssh -i .\servidorMinecraft_key.pem kupai@20.117.241.84
```

Una vez dentro, empezaremos a desarrollar la infraestructura del proyecto mejorado. El primer paso consistirá en `$ sudo apt update` y `$ sudo apt full-upgrade`. Una vez actualizada la máquina, instalamos Ansible, que es la herramienta de automatización con la que trabajaremos:

```
kupai@servidorMinecraft:~$ sudo apt install ansible default-jdk unzip wget -y
```

```
kupai@servidorMinecraft:~$ ansible --version
ansible [core 2.16.3]
  config file = None
  configured module search path = ['/home/kupai/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /home/kupai/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.12.3 (main, Feb  4 2025, 14:48:35) [GCC 13.3.0] (/usr/bin/python3)
  jinja version = 3.1.2
  libyaml = True
```

Paso 1. Crear la plantilla base del servidor

Para empezar, crearemos el directorio base del servidor, que será donde tendremos los archivos base desde los que generaremos los distintos mundos, funcionando como plantilla de estos. Ansible necesitara de este directorio junto a sus archivos configurados para poder clonarlo y generar cada nuevo mundo automáticamente.

El procedimiento es el siguiente:

```
kupai@servidorMinecraft:~$ mkdir ~/minecraft_server_base
kupai@servidorMinecraft:~$ ls
minecraft_server_base
```

Dentro, descargaremos el jar del servidor oficial dentro de este directorio, el cual nos permitirá jugar a la última versión de Minecraft:

```
wget https://piston-data.mojang.com/v1/objects/e6ec2f64e6080b9b5d9b471b291c33cc7f509733/server.jar
```

```
kupai@servidorMinecraft:~$ cd minecraft_server_base/
kupai@servidorMinecraft:~/minecraft_server_base$ wget https://piston-data.mojang.com/v1/objects/e6ec2f64e6080b9b5d9b471b291c33cc7f509733/server.jar
--2025-04-09 17:36:02--  https://piston-data.mojang.com/v1/objects/e6ec2f64e6080b9b5d9b471b291c33cc7f509733/server.jar
Resolving piston-data.mojang.com (piston-data.mojang.com)... 13.107.246.64, 2620:1ec:bdf::64
Connecting to piston-data.mojang.com (piston-data.mojang.com)|13.107.246.64|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 57269758 (55M) [application/octet-stream]
Saving to: 'server.jar'

server.jar               100%[===================================>]  54.62M  91.7MB/s    in 0.6s

2025-04-09 17:36:02 (91.7 MB/s) - 'server.jar' saved [57269758/57269758]

kupai@servidorMinecraft:~/minecraft_server_base$ ls
server.jar
kupai@servidorMinecraft:~/minecraft_server_base$
```

Con el "*server.jar*", ejecutamos el siguiente comando para que genere los archivos necesarios del mundo:

```
kupai@servidorMinecraft:~/minecraft_server_base$ java -Xmx1024M -Xms1024M -jar server.jar
```

```
kupai@servidorMinecraft:~/minecraft_server_base$ ls
eula.txt  libraries  logs  server.jar  server.properties  versions
kupai@servidorMinecraft:~/minecraft_server_base$
```

Ahora es necesario aceptar el "eula.txt", que es el **Acuerdo de Licencia de Usuario Final**:

```
#By changing the setting below to TRUE you are indicating your agreement to our EULA (https:
//aka.ms/MinecraftEULA).
#Wed Apr 09 17:39:08 UTC 2025
eula=true
~
```

Luego modificamos el fichero "*server.properties*" en caso de no estar configurado el puerto base ("*server-port=25565*"):

```
server-ip=
server-port=25565
simulation-distance=10
```

Con esto ya tendremos configurado el directorio plantilla. Podemos borrar el resto de los archivos puesto a que se regeneran con cada creación de un mundo:

```
kupai@servidorMinecraft:~/minecraft_server_base$ rm -rf world logs
kupai@servidorMinecraft:~/minecraft_server_base$ ls
eula.txt  libraries  server.jar  server.properties  versions
```

Paso 2. Crear la estructura de Ansible

Crearemos una estructura de directorios típica de una configuración con Ansible, la cual es necesaria para organizar los archivos y directorios para que Ansible pueda ejecutar comandos estructurados y reutilizables; crear nuevos mundos con nombres, puertos y servicios configurables; y escalar fácilmente las utilizades para implementar el resto de las mejoras propuestas.

Explicado el por qué, procederemos de la siguiente manera:

Crear la carpeta principal del proyecto Ansible (esto será el 'cerebro' de Ansible).

```
kupai@servidorMinecraft:~$ mkdir minecraft-ansible
kupai@servidorMinecraft:~$ ls
minecraft-ansible  minecraft_server_base
```

Una vez dentro, creamos la estructura básica.

```
kupai@servidorMinecraft:~/minecraft-ansible$ mkdir -p roles/create_instance/tasks
kupai@servidorMinecraft:~/minecraft-ansible$ mkdir -p roles/create_instance/templates
kupai@servidorMinecraft:~/minecraft-ansible$ ls
roles
```

Los "roles" agrupan tareas relacionadas, como "create_instance" que se encargara de crear el nuevo mundo de Minecraft.

Ahora crearemos los archivos vacíos iniciales:

```
kupai@servidorMinecraft:~/minecraft-ansible$ touch inventory.yml
kupai@servidorMinecraft:~/minecraft-ansible$ ls
inventory.yml  roles
```

"*inventory.yml*" >> Define los servidores.

```
kupai@servidorMinecraft:~/minecraft-ansible$ touch create_server.yml
kupai@servidorMinecraft:~/minecraft-ansible$ ls
create_server.yml  inventory.yml  roles
```

“*create_server.yml*” >> El playbook que lanza todo (como una especie de guion).

```
kupai@servidorMinecraft:~/minecraft-ansible$ cd roles/create_instance/tasks/
kupai@servidorMinecraft:~/minecraft-ansible/roles/create_instance/tasks$ touch
 main.yml
kupai@servidorMinecraft:~/minecraft-ansible/roles/create_instance/tasks$ ls
main.yml
```

“*tasks/main.yml*” >> Las acciones concretas (copiar, cambiar puerto, crear servicio…).

```
kupai@servidorMinecraft:~/minecraft-ansible$ cd roles/create_instance/templates/
kupai@servidorMinecraft:~/minecraft-ansible/roles/create_instance/templates$ touch
minecraft.service.j2
kupai@servidorMinecraft:~/minecraft-ansible/roles/create_instance/templates$ ls
minecraft.service.j2
kupai@servidorMinecraft:~/minecraft-ansible/roles/create_instance/templates$
```

“*templates/minecraft.service.j2*” >> Plantilla para el archivo systemd para cada mundo. Este archivo es una **plantilla de Jinja2**, que es un motor de plantillas que usa Ansible.

Con esto tendremos la estructura básica creada.

Paso 3. Configurar los archivos del sistema Ansible

1. “Inventory.yml”:

```yaml
all:
  hosts:
    localhost:
      ansible_connection: local
```

Esto define el inventario de servidores donde se ejecutarán los comandos. En este caso, como queremos que se ejecuten sobre la propia máquina, usaremos localhost y con “*ansible_connection: local*” indicamos que no se necesita de SSH.

2. “create_server.yml”:

```yaml
- name: Crear nuevo servidor de Minecraft
  hosts: localhost
  become: true
  vars:
    nuevo_servidor: minecraft_server_1
    puerto: 25565
    origen: ""  # dejar vacío para crear mundo nuevo, o poner por ejemplo: minecraft_server_1 para clonarlo
  roles:
    - create_instance
```

Esto es el playbook que lanza todo. Usa el rol “create_instance” para crear el servidor. También se declaran las 3 variables modificables, siendo estas “nuevo_servidor” para el nombre de la carpeta, puerto que usara o origen para clonar o crear uno de 0.

3. “main.yml”:

```yaml
- name: Clonar plantilla base
  copy:
    src: /home/kupai/minecraft_server_base/
    dest: "/home/kupai/{{ nuevo_servidor }}"
    remote_src: yes

- name: Cambiar el puerto
  lineinfile:
    path: "/home/kupai/{{ nuevo_servidor }}/server.properties"
    regexp: '^server-port='
    line: "server-port={{ puerto }}"

- name: Copiar mundo si se indica origen
  when: origen != ""
  copy:
    src: "/home/kupai/{{ origen }}/world"
    dest: "/home/kupai/{{ nuevo_servidor }}/world"
    remote_src: yes

- name: Asegurar permisos correctos en el nuevo servidor
  file:
    path: "/home/kupai/{{ nuevo_servidor }}"
    state: directory
    recurse: yes
    owner: kupai
    group: kupai

- name: Ejecutar el .jar unos segundos para generar archivos
  command: timeout 15s java -Xmx1024M -Xms1024M -jar server.jar nogui
  args:
    chdir: "/home/kupai/{{ nuevo_servidor }}"
  become: false
  register: salida_minecraft
  failed_when: false
```

```yaml
- name: Crear servicio systemd
  template:
    src: minecraft.service.j2
    dest: "/etc/systemd/system/{{ nuevo_servidor }}.service"

- name: Recargar systemd
  command: systemctl daemon-reexec

- name: Activar servidor
  systemd:
    name: "{{ nuevo_servidor }}"
    enabled: yes
    state: started
```

Automatiza la creación de un nuevo servidor Minecraft clonando una plantilla base, configurando su puerto, copiando un mundo si se indica, generando su servicio systemd y activándolo.

4. “minecraft.service.j2”:

```
[Unit]
Description=Minecraft Server {{ nuevo_servidor }}
After=network.target

[Service]
User=kupai
WorkingDirectory=/home/kupai/{{ nuevo_servidor }}
ExecStart=/usr/bin/java -Xmx1G -Xms1G -jar server.jar nogui
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

Se añade un archivo de unidad “systemd” que configura el arranque del servidor Minecraft. La sección “[Unit]” indica que el servicio se inicia tras la red. El usuario “kupai” ejecuta el servicio, con el directorio de trabajo asignado al nuevo servidor. El comando “ExecStart” lanza el archivo “.jar” del servidor. La opción “Restart=on-failure” asegura que el servicio se reinicie si falla. La sección “[Install]” permite habilitar el arranque automático con “systemctl enable”.

Paso 4. Ejecución de Ansible

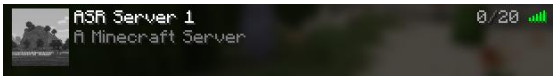
Con los archivos configurados, probamos a crear un nuevo mundo con la siguiente sentencia (recordar abrir el puerto en Azure):

```
kupai@servidorMinecraft:~/minecraft-ansible$ ansible-playbook create_server.yml -e "nuevo_servidor=minecraft_server_1 puerto=25565"
[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 [Crear nuevo servidor de Minecraft] ***************************************

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

TASK [create_instance : Clonar plantilla base] *********************************
changed: [localhost]

TASK [create_instance : Cambiar el puerto] *************************************
ok: [localhost]

TASK [create_instance : Copiar mundo si se indica origen] **********************
skipping: [localhost]

TASK [create_instance : Asegurar permisos correctos en el nuevo servidor] ******
changed: [localhost]

TASK [create_instance : Ejecutar el .jar unos segundos para generar archivos] ***
changed: [localhost]

TASK [create_instance : Crear servicio systemd] ********************************
changed: [localhost]

TASK [create_instance : Recargar systemd] **************************************
changed: [localhost]

TASK [create_instance : Activar servidor] **************************************
changed: [localhost]

PLAY RECAP *********************************************************************
localhost                  : ok=8    changed=6    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
```

Con esto tendremos desplegado un mundo. Podremos comprobarlo desde la máquina de la siguiente manera:

```
kupai@servidorMinecraft:~/minecraft-ansible$ sudo systemctl status minecraft_server_1
● minecraft_server_1.service - Minecraft Server minecraft_server_1
     Loaded: loaded (/etc/systemd/system/minecraft_server_1.service; enabled; preset: enabled)
     Active: active (running) since Wed 2025-04-09 20:25:33 UTC; 1min 0s ago
   Main PID: 23720 (java)
      Tasks: 37 (limit: 9459)
     Memory: 644.5M (peak: 645.0M)
        CPU: 45.204s
     CGroup: /system.slice/minecraft_server_1.service
             └─23720 /usr/bin/java -Xmx1G -Xms1G -jar server.jar nogui

Apr 09 20:25:53 servidorMinecraft java[23720]: [20:25:53] [Worker-Main-1/INFO]: Preparing spawn area: 0%
Apr 09 20:25:53 servidorMinecraft java[23720]: [20:25:53] [Worker-Main-1/INFO]: Preparing spawn area: 0%
Apr 09 20:25:53 servidorMinecraft java[23720]: [20:25:53] [Worker-Main-1/INFO]: Preparing spawn area: 0%
Apr 09 20:25:53 servidorMinecraft java[23720]: [20:25:53] [Worker-Main-1/INFO]: Preparing spawn area: 0%
Apr 09 20:25:53 servidorMinecraft java[23720]: [20:25:53] [Worker-Main-1/INFO]: Preparing spawn area: 0%
Apr 09 20:25:53 servidorMinecraft java[23720]: [20:25:53] [Worker-Main-1/INFO]: Preparing spawn area: 0%
Apr 09 20:25:54 servidorMinecraft java[23720]: [20:25:53] [Worker-Main-1/INFO]: Preparing spawn area: 0%
Apr 09 20:25:54 servidorMinecraft java[23720]: [20:25:54] [Server thread/INFO]: Time elapsed: 8276 ms
Apr 09 20:25:54 servidorMinecraft java[23720]: [20:25:54] [Server thread/INFO]: Done (8.598s)! For help, type "help"
Apr 09 20:25:58 servidorMinecraft java[23720]: [20:25:58] [Server thread/WARN]: Can't keep up! Is the server overloaded? Running 2519ms or 50 ticks behind
```

Nos vamos ahora a Minecraft, e intentamos localizar el nuevo servidor:

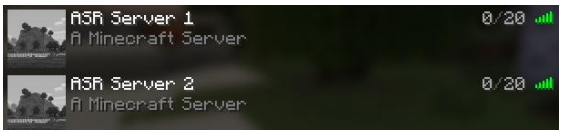


Si entramos vemos que el mundo funcióna a la perfección:

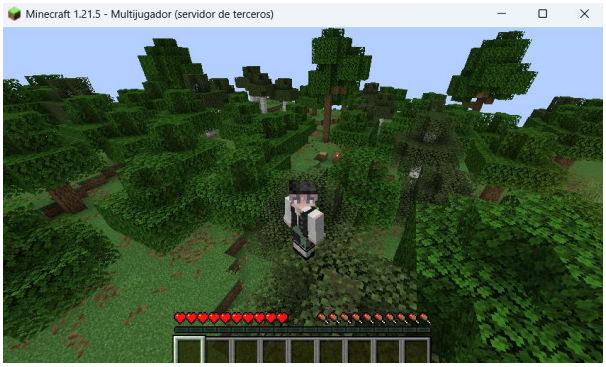
Ahora vamos a crear un nuevo mundo ejecutando nuevamente la automatización de ansible cambiando el puerto y el nombre (recordar abrir el puerto en Azure):

```
kupai@servidorMinecraft:~/minecraft-ansible$ ansible-playbook create_server.yml -e "nuevo_servidor=minecraft_server_2 puerto=25566"
[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'
```

Ahora, desde Minecraft, añadimos un nuevo servidor, con la misma IP, pero cambiando al puerto que introdujimos en el paso anterior:



Accedemos al mundo para ver que no es un clon del anterior, si no que es un mundo completamente nuevo:



A tener en cuenta

Para la creación de cada mundo, seguirá siendo necesario ir a Azure portal, a la configuración de nuestra máquina virtual y específicamente, al apartado de "Configuraciones de Red" y abrir cada puerto que estemos usando para el nuevo servidor.

Este paso se automatizará en el apartado de mejora a través de scripts.

Automatización de backups

Para esta segunda etapa del proyecto, buscaremos automatizar las backups para cada mundo de Minecraft, para facilitar la labor de los administradores en el control sobre los distintos servidores. Para ello conseguiremos que Ansible pueda guardar automáticamente una copia del mundo (el directorio "/world") en un formato ".tar.gz" dentro de una carpeta de backups.

Paso 1. Crear la infraestructura

El primer paso, al igual que hicimos en la parte anterior, es crear la estructura de directorios que Ansible utilizará para automatizar todo el proceso. Para ello, crearemos un nuevo rol llamado "backup_world". Además, crearemos el archivo "backup.yml", que será el "playbook" principal encargado de ejecutar ese rol, del mismo modo que hicimos anteriormente con "create_server.yml".

Para ello, ejecutamos las siguientes instrucciones en la maquina:

```
kupai@servidorMinecraft:~/minecraft-ansible$ mkdir -p roles/backup_world/tasks
kupai@servidorMinecraft:~/minecraft-ansible$ touch backup.yml
kupai@servidorMinecraft:~/minecraft-ansible$ ls
backup.yml  create_server.yml  inventory.yml  roles
kupai@servidorMinecraft:~/minecraft-ansible$ cd roles
kupai@servidorMinecraft:~/minecraft-ansible/roles$ ls
backup_world  create_instance
kupai@servidorMinecraft:~/minecraft-ansible/roles$ cd backup_world/tasks/
kupai@servidorMinecraft:~/minecraft-ansible/roles/backup_world/tasks$ touch main.yml
kupai@servidorMinecraft:~/minecraft-ansible/roles/backup_world/tasks$ ls
main.yml
kupai@servidorMinecraft:~/minecraft-ansible/roles/backup_world/tasks$
```

Paso 2. Modificar los ficheros de configuración

Modificaremos los dos nuevos ficheros de configuración para que haga las copias de seguridad automáticamente en cada mundo.

Modificaremos el playbook "backup.yml" primero. Su contenido es el siguiente:

```
- name: Backup de mundo de Minecraft
  hosts: localhost
  become: true
  vars:
    servidor: minecraft_server_1
  roles:
    - backup_world
```

Seguimos usando localhost debido a que los archivos se ejecutan sobre la misma máquina, y le definimos una variable servidor para saber sobre que mundo haremos el backup.

A continuación, vamos a definir las tareas del rol de backup. Para ello, editamos el archivo "roles/backup_world/tasks/main.yml" con el siguiente contenido:

```
- name: Crear carpeta de backups si no existe
  file:
    path: "/home/kupai/backups/{{ servidor }}"
    state: directory

- name: Comprimir el mundo
  command: >
    tar -czf /home/kupai/backups/{{ servidor }}/world_{{ ansible_date_time.date }}_{{ ansible_date_time.time }}.tar.gz
    /home/kupai/{{ servidor }}/world
```

La primera parte se encarga de crear (en caso de no existir), una carpeta por mundo dentro de "~/backups", y la segunda se encarga de crear un archivo ".tar.gz" con el formato "wolrd_FECHA_HORA.tar.gz", guardando el contenido de la carpeta en su interior.

Paso 3. Ejecutar la copia de seguridad

Por último, para probar su funcionamiento, ejecutaremos el siguiente comando que ejecutara la tarea que acabamos de crear:

```
kupai@servidorMinecraft:~/minecraft-ansible$ ansible-playbook backup.yml -e "servidor=minecraft_server_1"
[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 [Backup de mundo de Minecraft] ********************************************************************************

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

TASK [backup_world : Crear carpeta de backups si no existe] ********************************************************
changed: [localhost]

TASK [backup_world : Comprimir el mundo] ***************************************************************************
changed: [localhost]

PLAY RECAP *********************************************************************************************************
localhost                  : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

kupai@servidorMinecraft:~/minecraft-ansible$
```

Y comprobamos que se haya creado la carpeta de backups (que anteriormente no existía):

```
kupai@servidorMinecraft:~$ ls
backups  minecraft-ansible  minecraft_server_1  minecraft_server_2  minecraft_server_base
kupai@servidorMinecraft:~$
```

Y comprobamos su interior para ver que se haya creado la backup:

```
kupai@servidorMinecraft:~$ cd backups/
kupai@servidorMinecraft:~/backups$ ls
minecraft_server_1
kupai@servidorMinecraft:~/backups$ cd minecraft_server_1/
kupai@servidorMinecraft:~/backups/minecraft_server_1$ ls
world_2025-04-10_09:34:22.tar.gz
kupai@servidorMinecraft:~/backups/minecraft_server_1$
```

Ahí tenemos la copia de nuestro mundo creada correctamente. Ahora, para recuperar esta versión de la copia de seguridad del mundo, solo será necesario borrar la actual carpeta del mundo y descomprimir en el directorio del servidor en cuestión la copia de seguridad.

```
kupai@servidorMinecraft:~/backups/minecraft_server_1$ sudo systemctl stop minecraft_server_1
kupai@servidorMinecraft:~/backups/minecraft_server_1$ rm -rf /home/kupai/minecraft_server_1/world
kupai@servidorMinecraft:~/backups/minecraft_server_1$ tar -xzf /home/kupai/backups/minecraft_server_1/
world_2025-04-10_09:34:22.tar.gz -C /home/kupai/minecraft_server_1/
kupai@servidorMinecraft:~/backups/minecraft_server_1$ sudo systemctl start minecraft_server_1
kupai@servidorMinecraft:~/backups/minecraft_server_1$
```

Y comprobamos en Minecraft que el mundo sigue siendo el mismo:

El mundo sigue funcionando perfectamente incluso habiendo borrado la carpeta del mundo y volviendo a instaurar la del comprimido.

Paso 4. Restaurar una copia de seguridad

Vamos a automatizar el proceso de restauración de copiar en un *playbook* que detenga el servidor, elimine el mundo actual, descomprima el backup elegido y luego reinicie el servidor.

Para ello creamos un fichero “restore.yml” en el directorio base de Ansible (/minecraft-ansible) y un nuevo “rol” que sea “/restore_world”:

```
kupai@servidorMinecraft:~/minecraft-ansible$ mkdir -p roles/restore_world/tasks
kupai@servidorMinecraft:~/minecraft-ansible$ touch restore.yml
kupai@servidorMinecraft:~/minecraft-ansible$ touch roles/restore_world/tasks/main.yml
kupai@servidorMinecraft:~/minecraft-ansible$
```

Ahora configuramos dichos archivos:

“minecraft-ansible/restore.yml”:

```
- name: Restaurar un backup de mundo Minecraft
  hosts: localhost
  become: true
  vars:
    servidor: minecraft_server_1
    backup_file: world_2025-04-10_09:34:22.tar.gz
  roles:
    - restore_world
```

“minecraft-ansible/roles/restore_world/tasks/main.yml”:

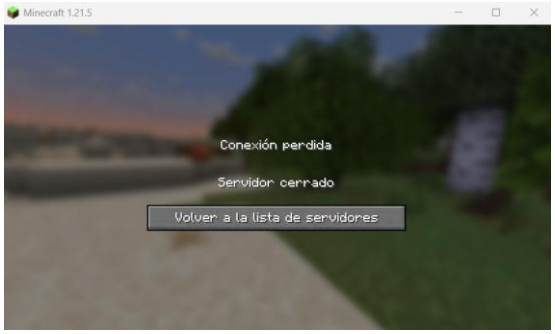
```
- name: Detener el servidor antes de restaurar
  systemd:
    name: "{{ servidor }}"
    state: stopped
    enabled: true

- name: Eliminar el mundo actual
  file:
    path: "/home/kupai/{{ servidor }}/world"
    state: absent

- name: Restaurar el backup
  unarchive:
    src: "/home/kupai/backups/{{ servidor }}/{{ backup_file }}"
    dest: "/home/kupai/{{ servidor }}"
    remote_src: yes
    extra_opts: ["--strip-components=3"]

- name: Reiniciar el servidor
  systemd:
    name: "{{ servidor }}"
    state: started
```

Ahora ejecutamos este proceso y revisamos que hemos vuelto a encima del árbol en el mundo de minecraft:

```
kupai@servidorMinecraft:~/minecraft-ansible$ ansible-playbook restore.yml -e "servidor=minecraft_server_1 backup_file=world_
2025-04-10_09:59:01.tar.gz"
[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 [Restaurar un backup de mundo Minecraft] **********************************************************************

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

TASK [restore_world : Detener el servidor antes de restaurar] ******************************************************
changed: [localhost]

TASK [restore_world : Eliminar el mundo actual] ********************************************************************
changed: [localhost]

TASK [restore_world : Restaurar el backup] *************************************************************************
changed: [localhost]

TASK [restore_world : Reiniciar el servidor] ***********************************************************************
changed: [localhost]

PLAY RECAP *********************************************************************************************************
localhost                  : ok=5    changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

kupai@servidorMinecraft:~/minecraft-ansible$
```

Antes de ejecutar la backup automatizada:

Mientras se ejecuta el servidor se cierra:



Después de la copia de seguridad:

No hay nada de lo que hice antes, funcionó correctamente.

Mejora mediante Scripts

El resultado de los cambios implementados hasta ahora es la posibilidad de crear nuevos servidores de Minecraft automáticamente. Al crearlos automáticamente crea también un servicio, lo que permite que el servidor se ejecute como servicio y este funcionando mediante "enable" desde que se arranca la máquina virtual, permitiendo tener los mundos corriendo sin necesidad de encenderlos manualmente y más cómodo que como lo teníamos en la anterior entrega.

Instalación de Azure CLI y autenticación

Con esto, conseguiremos que, una vez creado un nuevo servidor, el proceso de abrir el puerto correspondiente se ejecute de forma automática. Para ello instalamos:

```
kupai@servidorMinecraft:~$ curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash
```

Y revisamos que se haya instalado correctamente con `$ az versión`:

```
kupai@servidorMinecraft:~$ az version
{
  "azure-cli": "2.71.0",
  "azure-cli-core": "2.71.0",
  "azure-cli-telemetry": "1.1.0",
  "extensions": {}
}
```

Ahora tendremos que registrarnos con nuestras credenciales la primera vez con `$ az login`.

```
kupai@servidorMinecraft:~$ az login
To sign in, use a web browser to open the page https://microsoft.com/devicelogin and enter the code G6'    4JU to authenticate.
```

En el navegador de nuestra máquina introducimos el código, y seleccionamos en la siguiente pestaña la cuenta que usaremos registrándonos en ella (no se incluirá esta parte por motivos obvios de seguridad):

Una vez hecho, volvemos al cmd y comprobamos que se haya iniciado sesión correctamente (datos confidenciales han sido censurados):

```
kupai@servidorMinecraft:~$ az login
To sign in, use a web browser to open the page https://microsoft.com/devicelogin and enter the code G6V99G4JU to authenticate.

Retrieving tenants and subscriptions for the selection...

[Tenant and subscription selection]

No     Subscription name    Subscription ID                       Tenant
-----  -------------------  ------------------------------------  --------------------
[1] *  Azure for Students   b                                  4  Universidad de Oviedo

The default is marked with an *; the default tenant is 'Universidad de Oviedo' and subscription is 'Azure for Students' (b                                  .c
4).

Select a subscription and tenant (Type a number or Enter for no changes):

Tenant: Universidad de Oviedo
Subscription: Azure for Students (b                                  4)

[Announcements]
With the new Azure CLI login experience, you can select the subscription you want to use more easily. Learn more about it and its configuration at https://g
o.microsoft.com/fwlink/?linkid=2271236

If you encounter any problem, please open an issue at https://aka.ms/azclibug

[Warning] The login output has been updated. Please be aware that it no longer displays the full list of available subscriptions by default.
```

Script de creación de servers

Lo que haremos ahora será, dejar todo el proceso de creación de mundo completamente automatizado solo con la ejecución de un único script, al cual llamaremos "crear_mundo.sh". Este lo ubicaremos en la carpeta home (es decir, el directorio base, por mantener una separación entre *playbooks* y scripts) y su contenido es:

```bash
#!/bin/bash

# Preguntar si quiere nombre personalizado
read -p "Nombre personalizado para el nuevo servidor (dejar vacío para autogenerar): " NOMBRE_INPUT

if [ -z "$NOMBRE_INPUT" ]; then
  # Sin nombre personalizado → buscar el siguiente número libre
  NUM=1
  while [ -d "$HOME/servidor_minecraft_$NUM" ]; do
    ((NUM++))
  done
  NOMBRE_SERVIDOR="servidor_minecraft_$NUM"
else
  # Si escribió nombre y no empieza por "servidor_minecraft_", añadirlo
  if [[ "$NOMBRE_INPUT" != servidor_minecraft_* ]]; then
    NOMBRE_SERVIDOR="servidor_minecraft_$NOMBRE_INPUT"
  else
    NOMBRE_SERVIDOR="$NOMBRE_INPUT"
  fi
fi
```

```bash
# Buscar puerto libre comenzando en 25565
PUERTO=25565
while ss -tuln | grep -q ":$PUERTO "; do
  ((PUERTO++))
done

# Crear la regla de puerto en Azure
az network nsg rule create \
  --resource-group servidorMinecraft_group \
  --nsg-name servidorMinecraft-nsg \
  --name "Allow_${PUERTO}_server" \
  --priority $((PUERTO - 25200)) \
  --direction Inbound \
  --access Allow \
  --protocol Tcp \
  --destination-port-ranges $PUERTO \
  --source-address-prefixes '*' \
  --destination-address-prefixes '*' >/dev/null

# Ejecutar playbook de creación de servidor
cd ~/minecraft-ansible || { echo "❌ No se encontró la carpeta de Ansible"; exit 1; }
ansible-playbook create_server.yml -e "nuevo_servidor=$NOMBRE_SERVIDOR puerto=$PUERTO"

echo "✅ Servidor '$NOMBRE_SERVIDOR' creado y disponible en el puerto $PUERTO."
```

Guardamos el nuevo fichero, le damos permisos de ejecución con `$ chmod +x ~/crear_mundo.sh` y ahora comprobaremos que funciona con el comando siguiente: `$ ./crear_mundo.sh`

```
kupai@servidorMinecraft:~$ ./crear_mundo.sh
Nombre del servidor (dejar vacío para autogenerar):
```

Lo dejaremos vacío para demostrar su funcionamiento.

El resultado es:

```
kupai@servidorMinecraft:~$ ./crear_mundo.sh
Nombre del servidor (dejar vacío para autogenerar):
➕ Nombre generado automáticamente: servidor_minecraft_1
🔌 Puerto asignado: 25567
[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 [Crear nuevo servidor de Minecraft] ***************************************************************************************************************************

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

TASK [create_instance : Clonar plantilla base] *********************************************************************************************************************
changed: [localhost]

TASK [create_instance : Cambiar el puerto] *************************************************************************************************************************
changed: [localhost]

TASK [create_instance : Copiar mundo si se indica origen] **********************************************************************************************************
skipping: [localhost]

TASK [create_instance : Asegurar permisos correctos en el nuevo servidor] ******************************************************************************************
changed: [localhost]

TASK [create_instance : Ejecutar el .jar unos segundos para generar archivos] **************************************************************************************
changed: [localhost]

TASK [create_instance : Crear servicio systemd] ********************************************************************************************************************
changed: [localhost]

TASK [create_instance : Recargar systemd] **************************************************************************************************************************
changed: [localhost]

TASK [create_instance : Activar servidor] **************************************************************************************************************************
changed: [localhost]

PLAY RECAP *********************************************************************************************************************************************************
localhost                  : ok=8    changed=7    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
```


```
{
  "access": "Allow",
  "description": "Regla automática para servidor_minecraft_1",
  "destinationAddressPrefix": "*",
  "destinationAddressPrefixes": [],
  "destinationPortRange": "25567",
  "destinationPortRanges": [],
  "direction": "Inbound",
  "etag": "W/\"9129bf97-4958-4b41-99a3-a08549d3565e\"",
  "id": "/subscriptions/bd5ce024-e8bd-4784-8bbc-e4975cc1f1c4/resourceGroups/servidorMinecraft_group/providers/Microsoft.Network/networkSecurityGroups/servidorMinecraft-nsg/securityRules/Allow_25567_server",
  "name": "Allow_25567_server",
  "priority": 312,
  "protocol": "Tcp",
  "provisioningState": "Succeeded",
  "resourceGroup": "servidorMinecraft_group",
  "sourceAddressPrefix": "*",
  "sourceAddressPrefixes": [],
  "sourcePortRange": "*",
  "sourcePortRanges": [],
  "type": "Microsoft.Network/networkSecurityGroups/securityRules"
}
✅ Mundo 'servidor_minecraft_1' creado y puerto 25567 abierto en Azure
kupai@servidorMinecraft:~$
```

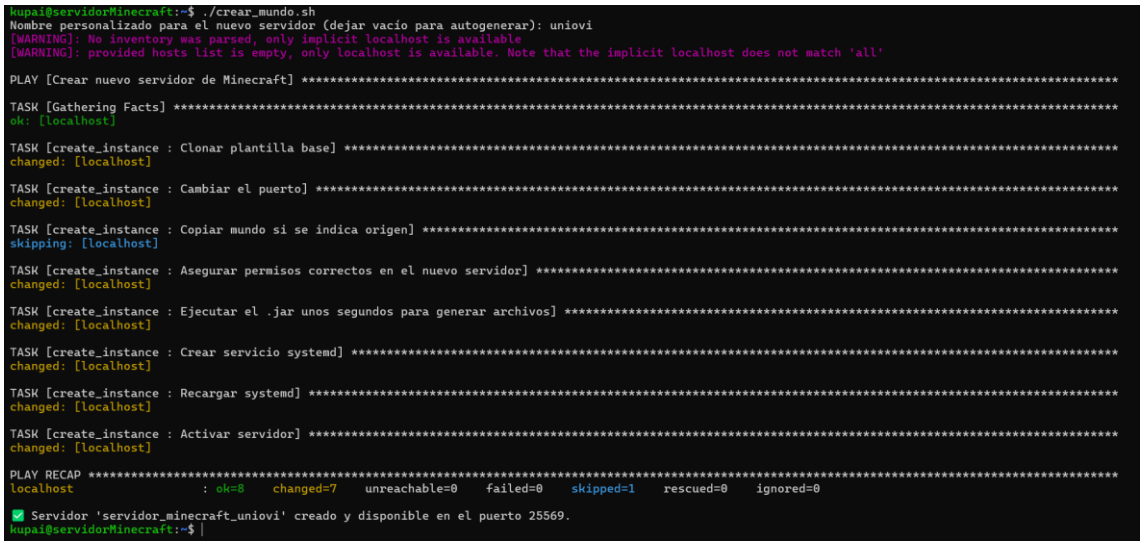

Vemos que se ha creado la carpeta correspondiente:

```
kupai@servidorMinecraft:~$ ls
ElGaitero  backups  crear_mundo.sh  minecraft-ansible  minecraft_server_1  minecraft_server_2  minecraft_server_base  servidor_minecraft_1
kupai@servidorMinecraft:~$
```

Vemos que se ha abierto el puerto en Azure:

Y vemos que el servidor está ya en funcionamiento:

Si introdujéramos un nombre personalizado, el resultado es el siguiente:

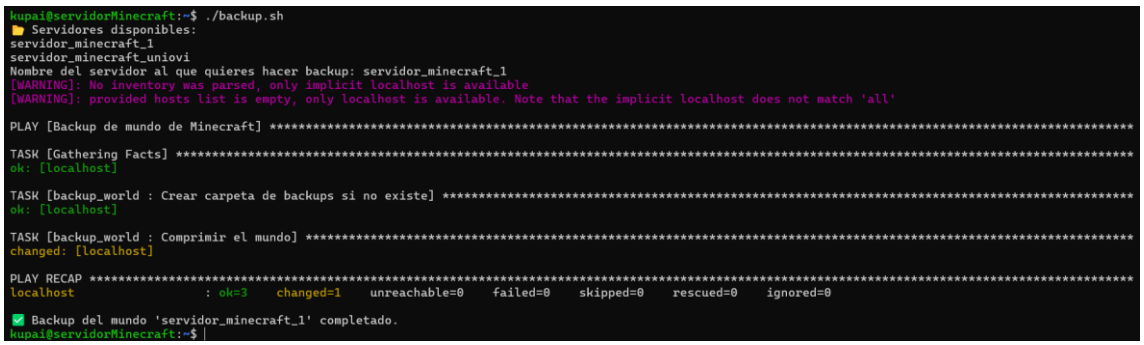


Script de creación de backups

Para este nuevo script, creamos también en el directorio raíz un nuevo fichero llamado backup.sh siendo su contenido:

```
#!/bin/bash

echo "📂 Servidores disponibles:"
ls -d ~/servidor_minecraft_* 2>/dev/null | xargs -n 1 basename

read -p "Nombre del servidor al que quieres hacer backup: " SERVIDOR

# Validación: ¿existe el mundo?
if [ ! -d "$HOME/$SERVIDOR/world" ]; then
  echo "❌ El servidor '$SERVIDOR' no existe o no tiene un mundo válido."
  exit 1
fi

cd ~/minecraft-ansible || { echo "❌ No se encontró la carpeta de Ansible"; exit 1; }
ansible-playbook backup.yml -e "servidor=$SERVIDOR"

echo "✅ Backup del mundo '$SERVIDOR' completado."
```

Esto muestra al usuario los servidores disponibles, y una vez introducido el nombre del servidor, avisa si se ha podido crear o no. Por último, le damos permisos de ejecución con la instrucción `$ chmod +x ~/backup.sh` y lo ejecutamos con `$ ./backup.sh`.



Si no existe muestra:

```
kupai@servidorMinecraft:~$ ./backup.sh
📂 Servidores disponibles:
servidor_minecraft_1
Nombre del servidor al que quieres hacer backup: servidor_minecraft_noexiste
❌ El servidor 'servidor_minecraft_noexiste' no existe o no tiene un mundo válido.
kupai@servidorMinecraft:~$
```

Script de restaurar un backup

Nuevamente desde el directorio raíz, creamos un nuevo archivo restaurar_backup.sh con el siguiente contenido:

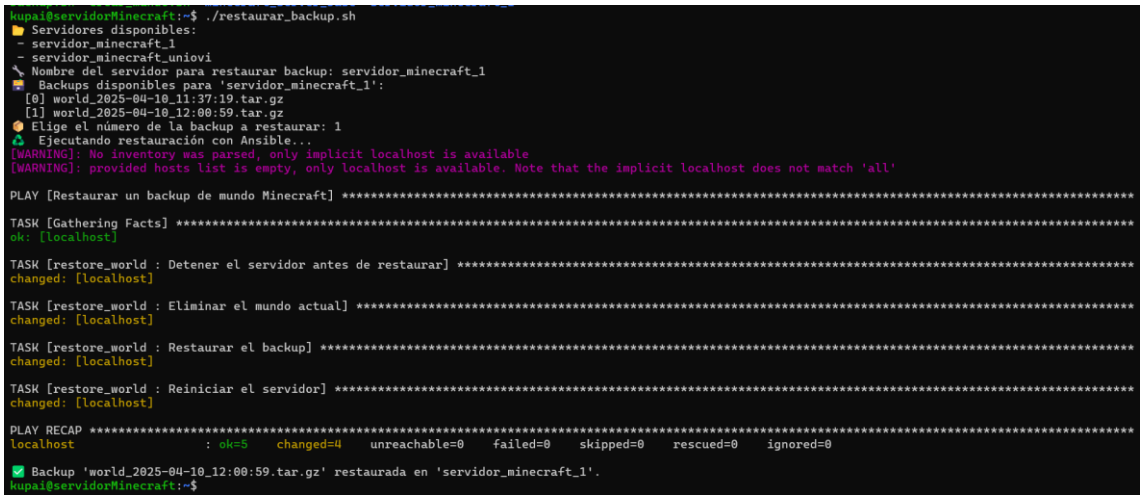
```
#!/bin/bash

echo "📂 Servidores disponibles:"
SERVIDORES=($(ls -d ~/servidor_minecraft_* 2>/dev/null | xargs -n 1 basename))

if [ ${#SERVIDORES[@]} -eq 0 ]; then
  echo "❌ No hay servidores disponibles."
  exit 1
fi

for S in "${SERVIDORES[@]}"; do echo " - $S"; done

read -p "🔧 Nombre del servidor para restaurar backup: " SERVIDOR

DIR="$HOME/$SERVIDOR"
BACKUP_DIR="$HOME/backups/$SERVIDOR"

if [ ! -d "$DIR" ]; then
  echo "❌ El servidor '$SERVIDOR' no existe."
  exit 1
fi

if [ ! -d "$BACKUP_DIR" ]; then
  echo "❌ No hay carpeta de backups para '$SERVIDOR'."
  exit 1
fi

BACKUPS=($(ls "$BACKUP_DIR"/*.tar.gz 2>/dev/null | sort))
if [ ${#BACKUPS[@]} -eq 0 ]; then
  echo "❌ No hay backups disponibles para '$SERVIDOR'."
  exit 1
fi

echo "🗃️  Backups disponibles para '$SERVIDOR':"
for i in "${!BACKUPS[@]}"; do
  echo "  [$i] $(basename "${BACKUPS[$i]}")"
done
```

```
read -p "📦 Elige el número de la backup a restaurar: " ELECCION

if ! [[ "$ELECCION" =~ ^[0-9]+$ ]] || [ "$ELECCION" -lt 0 ] || [ "$ELECCION" -ge "${#BACKUPS[@]}" ]; then
  echo "❌ Opción inválida."
  exit 1
fi

BACKUP_FILE=$(basename "${BACKUPS[$ELECCION]}")

echo "🚀 Ejecutando restauración con Ansible..."

cd ~/minecraft-ansible || { echo "❌ No se encontró la carpeta de Ansible"; exit 1; }
ansible-playbook restore.yml -e "servidor=$SERVIDOR backup_file=$BACKUP_FILE"

echo "✅ Backup '$BACKUP_FILE' restaurada en '$SERVIDOR'."
```

Esto hace que, en caso de introducir mal el servidor, no tenga backups asociadas o elija una backup incorrecta finalice controladamente. En caso de estar todo correcto crea una.

Como siempre le damos permisos de ejecución `$ chmod +x ~/restaurar_backup.sh` y ejecutamos `$ ./restaurar_backup.sh`.

El resultado es:



Script de encendido y apagado del mundo

Nuevamente creamos un fichero nuevo llamado gestionar_servidor.sh, cuyo contenido será:

```
#!/bin/bash

# Listar servicios correctamente sin símbolos raros
echo "📂 Servidores disponibles:"
SERVICIOS=($(systemctl list-units --type=service --all --no-legend --plain | grep servidor_minecraft_ | awk '{print $1}' | sed 's/.service$//'))

if [ ${#SERVICIOS[@]} -eq 0 ]; then
  echo "❌ No hay servidores registrados como servicios."
  exit 1
fi

# Mostrar servidores numerados
for i in "${!SERVICIOS[@]}"; do
  echo " [$i] ${SERVICIOS[$i]}"
done

read -p "🧩 Elige el número del servidor: " ELECCION

# Validar selección
if ! [[ "$ELECCION" =~ ^[0-9]+$ ]] || [ "$ELECCION" -lt 0 ] || [ "$ELECCION" -ge "${#SERVICIOS[@]}" ]; then
  echo "❌ Opción inválida."
  exit 1
fi
```

```
SERVIDOR="${SERVICIOS[$ELECCION]}"

# Mostrar acciones
echo "   ¿Qué quieres hacer con '$SERVIDOR'?"
echo " [1] Encender"
echo " [2] Apagar"
read -p "Elige opción: " ACCION

if [ "$ACCION" = "1" ]; then
  sudo systemctl start "$SERVIDOR"
  echo "✅ Servidor '$SERVIDOR' encendido."
elif [ "$ACCION" = "2" ]; then
  sudo systemctl stop "$SERVIDOR"
  echo "🔴 Servidor '$SERVIDOR' apagado."
else
  echo "❌ Acción inválida."
  exit 1
fi

# Mostrar estado final del servidor
systemctl status "$SERVIDOR" --no-pager
```

Le damos permisos de ejecución `$ chmod +x ~/gestionar_servidor.sh` y lo ejecutamos con la instrucción `$ ./gestionar_servidor.sh`:

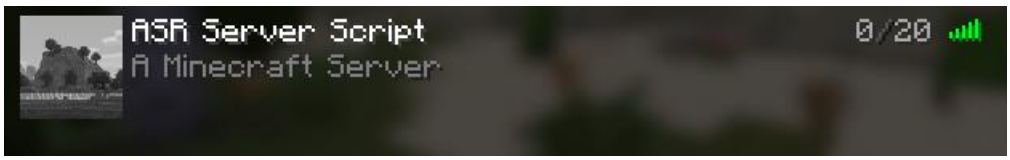
```
kupai@servidorMinecraft:~$ ./gestionar_servidor.sh
📂 Servidores disponibles:
 [0] servidor_minecraft_1
 [1] servidor_minecraft_uniovi
🧩 Elige el número del servidor: 0
⚙️  ¿Qué quieres hacer con 'servidor_minecraft_1'?
 [1] Encender
 [2] Apagar
Elige opción: 2
🔴 Servidor 'servidor_minecraft_1' apagado.
× servidor_minecraft_1.service - Minecraft Server servidor_minecraft_1
     Loaded: loaded (/etc/systemd/system/servidor_minecraft_1.service; enabled; preset: enabled)
     Active: failed (Result: exit-code) since Thu 2025-04-10 12:18:04 UTC; 8ms ago
   Duration: 7min 49.358s
    Process: 10338 ExecStart=/usr/bin/java -Xmx1G -Xms1G -jar server.jar nogui (code=exited, status=143)
   Main PID: 10338 (code=exited, status=143)
        CPU: 35.564s

Apr 10 12:10:32 servidorMinecraft java[10338]: [12:10:32] [Worker-Main-1/INFO]: Preparing spawn area: 0%
Apr 10 12:10:32 servidorMinecraft java[10338]: [12:10:32] [Worker-Main-1/INFO]: Preparing spawn area: 0%
Apr 10 12:10:32 servidorMinecraft java[10338]: [12:10:32] [Server thread/INFO]: Time elapsed: 6965 ms
Apr 10 12:10:32 servidorMinecraft java[10338]: [12:10:32] [Server thread/INFO]: Done (7.319s)! For help, type "help"
Apr 10 12:11:32 servidorMinecraft java[10338]: [12:11:32] [Server thread/INFO]: Server empty for 60 seconds, pausing
Apr 10 12:18:03 servidorMinecraft systemd[1]: Stopping servidor_minecraft_1.service - Minecraft Server servidor_minecraft_1...
Apr 10 12:18:04 servidorMinecraft systemd[1]: servidor_minecraft_1.service: Main process exited, code=exited, status=143/n/a
Apr 10 12:18:04 servidorMinecraft systemd[1]: servidor_minecraft_1.service: Failed with result 'exit-code'.
Apr 10 12:18:04 servidorMinecraft systemd[1]: Stopped servidor_minecraft_1.service - Minecraft Server servidor_minecraft_1.
Apr 10 12:18:04 servidorMinecraft systemd[1]: servidor_minecraft_1.service: Consumed 35.564s CPU time, 665.3M memory peak, 0B memory swap peak.
kupai@servidorMinecraft:~$
```

Todo fue a la perfección, y en Minecraft podemos comprobar que se apagó:

Ahora lo volveremos a encender:

```
kupai@servidorMinecraft:~$ ./gestionar_servidor.sh
📂 Servidores disponibles:
 [0] servidor_minecraft_1
 [1] servidor_minecraft_uniovi
🧩 Elige el número del servidor: 0
⚙️  ¿Qué quieres hacer con 'servidor_minecraft_1'?
 [1] Encender
 [2] Apagar
Elige opción: 1
✅ Servidor 'servidor_minecraft_1' encendido.
● servidor_minecraft_1.service - Minecraft Server servidor_minecraft_1
     Loaded: loaded (/etc/systemd/system/servidor_minecraft_1.service; enabled; preset: enabled)
     Active: active (running) since Thu 2025-04-10 12:24:21 UTC; 14ms ago
   Main PID: 10570 (java)
      Tasks: 2 (limit: 9442)
     Memory: 2.0M (peak: 2.0M)
        CPU: 6ms
     CGroup: /system.slice/servidor_minecraft_1.service
             └─10570 /usr/bin/java -Xmx1G -Xms1G -jar server.jar nogui

Apr 10 12:24:21 servidorMinecraft systemd[1]: Started servidor_minecraft_1.service - Minecraft Server servidor_minecraft_1.
kupai@servidorMinecraft:~$
```

Y en Minecraft vemos que funcionó:



Autenticado y control de Servidores vía web.

En este punto lo que hemos hecho es, básicamente, usar conceptos aprendidos en otras asignaturas como puede ser Sistemas Distribuidos e Internet para realizar una aplicación en Spring-Boot donde se gestionen los servidores creados y realice una autenticación sencilla.

Paso 1. Creación del proyecto.

Como IDE usamos IntelliJ en su versión de pago. Y creamos una plantilla básica de aplicación desarrollada en Spring-Boot. Una vez hecho esto, simplemente hicimos el "frontend" y "backend" de los 3 "endpoints" que íbamos a necesitar, uno referente a la autenticación, otro referente a la creación de y activación de servidores, y el último encargado de apagar los servidores.

Realmente aquí lo único complejo es en la capa de servicio, decirle la forma en la que trata el apagado y la creación de los servidores. En nuestro caso en el "endpoint":

```
POST /api/servers/{id}/start
```

centralizamos tanto la creación como el inicio de los mundos. Esto se debe a que ejecuta un playbook de Ansible llamado create_server.yml, al cual se le pasa como variable el identificador del mundo (nuevo_servidor={id}). El comportamiento del playbook es:

- Si el mundo no existe, se crea un nuevo directorio clonado desde una plantilla base (minecraft_server_base).
- Se genera un servicio systemd para gestionarlo automáticamente con el nombre
- Se inicia dicho servicio con systemctl start.

De esta forma obtenemos un resultado óptimo y sin necesidad de crear un endpoint distinto.

Mientras que el apago se realiza desde este otro endpoint:

```
POST /api/servers/{id}/stop
```

Esto, ejecuta un "`$ systemctl stop servidor_minecraft_{id}.service`" para apagar el servidor, no sin antes haber comprobado si efectivamente estaba activo con "systemctl is-active".

Para la autenticación, como la aplicación va a ser de uso personal, hemos creado solamente un usuario administrador que será el único con el que se permita la autenticación.

Por último, creamos las vistas y les ponemos estilo para que se pueda ver desde un navegador como Chrome.

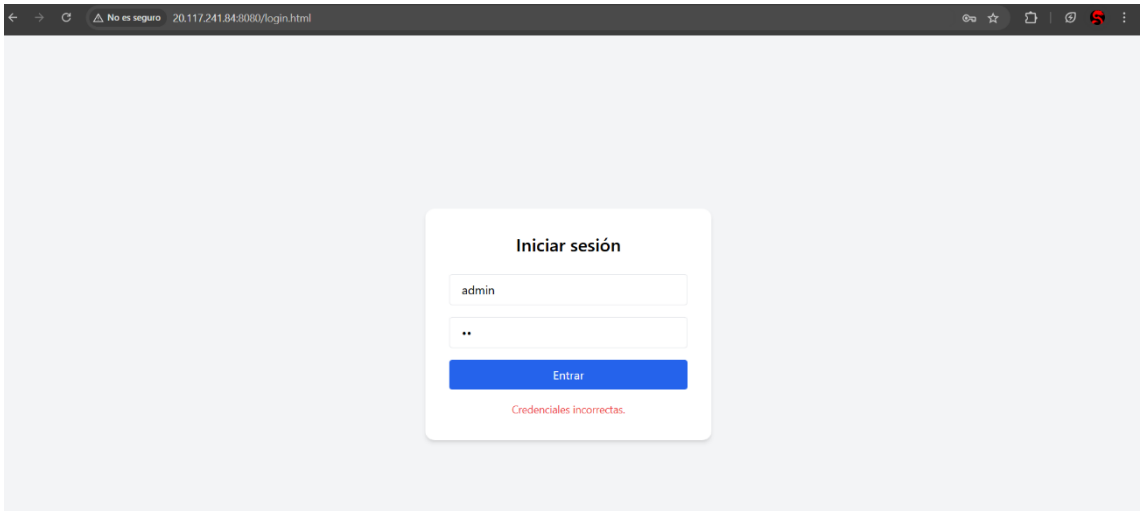
Paso 2. Despliegue de la aplicación.

Copiamos el proyecto en la máquina virtual. En nuestro caso, lo hemos hecho mediante ssh y con el comando "`$ scp /ruta/local/al/proyecto kupai@IP_DEL_SERVIDOR:~/ElGaitero`".

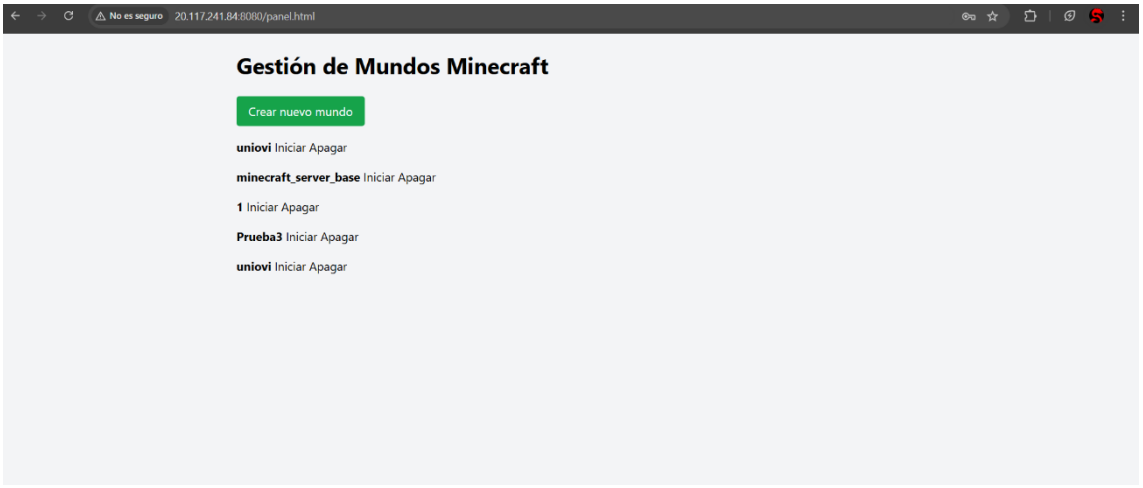
A continuación, nos situamos en la carpeta y realizamos "`$ ./mvnw clean package`" para compilar el proyecto y crear así un .jar que será el que despliegue el proyecto gracias al comando "`$ java -jar target/ElGaitero-0.0.1-SNAPSHOT.jar`". Una vez lanzada la aplicación vemos lo siguiente:

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.4.4)

2025-04-17T11:55:20.433Z  INFO 2081 --- [ElGaitero] [           main] c.uniovi.elgaitero.ElGaiteroApplication  : Startin
g ElGaiteroApplication v0.0.1-SNAPSHOT using Java 21.0.6 with PID 2081 (/home/kupai/ElGaitero/target/ElGaitero-0.0.1-SNA
PSHOT.jar started by kupai in /home/kupai/ElGaitero/target)
2025-04-17T11:55:20.439Z  INFO 2081 --- [ElGaitero] [           main] c.uniovi.elgaitero.ElGaiteroApplication  : No acti
ve profile set, falling back to 1 default profile: "default"
2025-04-17T11:55:21.802Z  INFO 2081 --- [ElGaitero] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat
initialized with port 8080 (http)
2025-04-17T11:55:21.827Z  INFO 2081 --- [ElGaitero] [           main] o.apache.catalina.core.StandardService   : Startin
g service [Tomcat]
2025-04-17T11:55:21.828Z  INFO 2081 --- [ElGaitero] [           main] o.apache.catalina.core.StandardEngine    : Startin
g Servlet engine: [Apache Tomcat/10.1.39]
2025-04-17T11:55:21.865Z  INFO 2081 --- [ElGaitero] [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initial
izing Spring embedded WebApplicationContext
2025-04-17T11:55:21.868Z  INFO 2081 --- [ElGaitero] [           main] w.s.c.ServletWebServerApplicationContext : Root We
bApplicationContext: initialization completed in 1314 ms
2025-04-17T11:55:22.384Z  INFO 2081 --- [ElGaitero] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat
started on port 8080 (http) with context path '/'
2025-04-17T11:55:22.406Z  INFO 2081 --- [ElGaitero] [           main] c.uniovi.elgaitero.ElGaiteroApplication  : Started
 ElGaiteroApplication in 2.666 seconds (process running for 3.316)
```

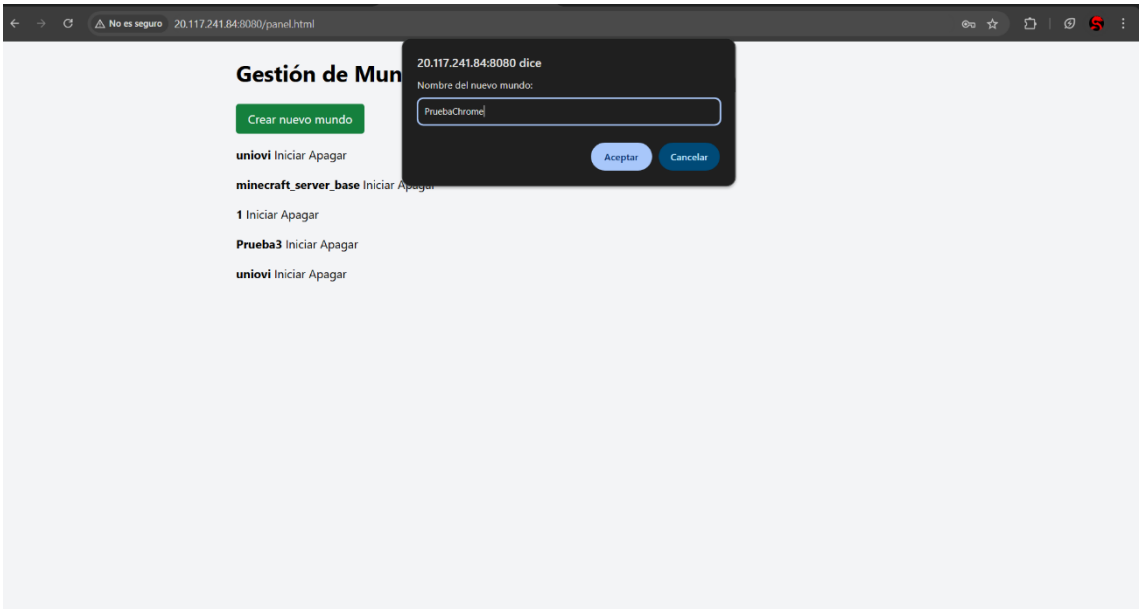
Con la aplicación iniciada, simplemente accedemos mediante un navegador para comprobar su funcionamiento.



Como vemos, no nos permite acceder si las credenciales no son las correctas.



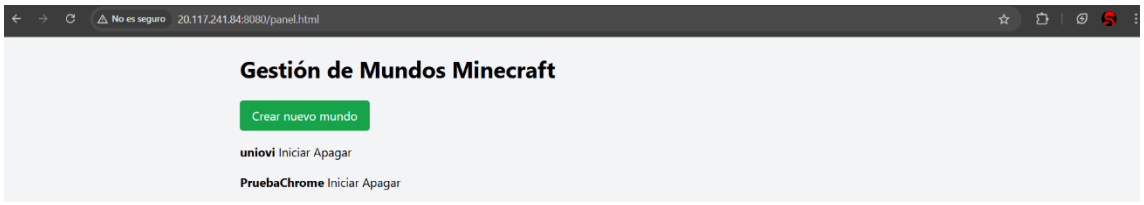
Una vez accedido con las credenciales correctas, vemos un listado de los mundos disponibles además de dos botones a su lado, uno para iniciar y otro para apagar el mundo. Además, vemos otro botón con el que podemos crear un nuevo mundo.



Probamos a crear un mundo con el nombre PruebaChrome.

```
[ANSIBLE] [WARNING]: No inventory was parsed, only implicit localhost is available
[ANSIBLE] [WARNING]: provided hosts list is empty, only localhost is available. Note that
[ANSIBLE] the implicit localhost does not match 'all'
[ANSIBLE]
[ANSIBLE] PLAY [Crear nuevo servidor de Minecraft] ************************************
[ANSIBLE]
[ANSIBLE] TASK [Gathering Facts] ******************************************************
[ANSIBLE] ok: [localhost]
[ANSIBLE]
[ANSIBLE] TASK [create_instance : Clonar plantilla base] ******************************
[ANSIBLE] changed: [localhost]
[ANSIBLE]
[ANSIBLE] TASK [create_instance : Cambiar el puerto] **********************************
[ANSIBLE] ok: [localhost]
[ANSIBLE]
[ANSIBLE] TASK [create_instance : Copiar mundo si se indica origen] *******************
[ANSIBLE] skipping: [localhost]
[ANSIBLE]
[ANSIBLE] TASK [create_instance : Asegurar permisos correctos en el nuevo servidor] ******
[ANSIBLE] changed: [localhost]
[ANSIBLE]
[ANSIBLE] TASK [create_instance : Ejecutar el .jar unos segundos para generar archivos] ***
[ANSIBLE] changed: [localhost]
[ANSIBLE]
[ANSIBLE] TASK [create_instance : Crear servicio systemd] *****************************
[ANSIBLE] changed: [localhost]
[ANSIBLE]
[ANSIBLE] TASK [create_instance : Recargar systemd] ***********************************
[ANSIBLE] changed: [localhost]
[ANSIBLE]
[ANSIBLE] TASK [create_instance : Activar servidor] ***********************************
[ANSIBLE] changed: [localhost]
[ANSIBLE]
[ANSIBLE] PLAY RECAP *****************************************************************
[ANSIBLE] localhost                  : ok=8    changed=6    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
```

Desde el lado del servidor se ejecuta el playbook tal como esperamos, y tras esperar un poco, se crea correctamente y aparece en el listado.



Con esta interfaz web creada, podríamos implementar todos los servicios que quisiéramos asociar a nuestro control de servidores.