ADMINISTRACIÓN DE SERVIDOR DE MINECRAFT EN Azure

Administración de Sistemas y Redes

Diego García, UO294255
Sergio García, UO294636

2024 - 2025

Escuela de Ingeniería Informática
Universidad de Oviedo

Índice

Administración y despliegue de un servidor de Minecraft en una máquina virtual en Azure

Como parte del trabajo teórico, mi compañero y yo hemos ideado una manera original e interesante de poner en práctica los conceptos aprendidos en clase sobre la gestión de servidores en máquinas remotas. Dado que somos de la generación Z y WordPress no es precisamente nuestra herramienta favorita, decidimos pensar en un servidor que nos resultara útil instalar. Así que, de manera automática, pensamos en un servidor de Minecraft, donde podríamos crear un mundo para compartir experiencias divertidas y momentos graciosos con nuestros amigos.

Por ello, en este trabajo mostraremos el paso a paso y las explicaciones necesarias para llevar a cabo esta tarea.

Creación de la máquina virtual

El primer paso consiste en crear la máquina virtual donde desplegaremos toda la infraestructura del servidor. Aquí es donde surgen las primeras preguntas: ¿Dónde desplegamos la máquina virtual? ¿Qué características debe tener?

A continuación, explicaremos detenidamente cada decisión tomada.

Como plataforma donde desplegar utilizaremos Azure, puesto a que tanto Minecraft y Azure son productos de Microsoft, y aunque no tengan un impacto directo en la implementación de dicho servidor, puesto a que este depende de la máquina virtual, el sistema operativo y de los recursos asignados; puede ser más cómodo y eficiente la gestión del proyecto y el manejo de credenciales puesto a que están ambos en el mismo ecosistema; además de que ya hemos trabajado con anterioridad con ello sin terminal de darle una aplicación más practica que seguir un guion.

Otras opciones muy interesantes serian AWS, que es el producto de Amazon, que oferta de manera extensa instancias de computación y servicios adicionales como es el caso de S3 para el almacenamiento o CloudWatch para un mejor monitoreo. También otra opción que barajamos fue Oracle Cloud, que ofrece una capa gratuita, que tiene alta disponibilidad, pero debido a que un servidor de Minecraft puede llegar a forzarse mucho, temíamos que los recursos que ofrece puedan limitar la experiencia.

Una vez elegida la plataforma desde donde montaremos todo, Microsoft Azure, toca la parte de la creación de la maquina desde donde instalaremos los recursos necesarios.

Elección del SO

Para la creación de la máquina, es necesario tener en cuenta el cumplir con los requisitos necesarios para poder gestionarlo todo. Por ello hemos estudiado cuales son los componentes más rentables y los mostramos a continuación:

La primera decisión es si elegir un sistema operativo Ubuntu o un sistema Windows Server, ya que cada sistema operativo nos ofrece diferentes ventajas.

Desde Windows Server, contamos con una interfaz gráfica, desde donde tendremos una experiencia más fácil de gestionar. Además, solamente Windows cuenta con compatibilidad con “Minecrat Bedrock Edition”, y desde una maquina Ubuntu Linux no

contaríamos con este soporte oficial. Lo bueno de esto, es que podríamos hacer accesible el servidor para jugadores de diferentes dispositivos como lo son consolas o móvil (además de claramente desde ordenador) a través de la dirección del servidor.

En cambio, con un sistema operativo Linux, Ubuntu para ser exactos; tendríamos que implementar la versión "Minecraft Java Edition", con lo cual estaremos limitando el acceso a usuarios, pero como en nuestro caso será un servidor para amigos que todos poseemos los medios para tener la versión de Java, no nos importa. Además, al basarnos en esta edición, contaríamos con la última versión estable de Minecraft.

En conclusión, aunque Windows nos permite más accesibilidad además de más compatibilidad con software de Windows, que tampoco es nuestro caso, debido a que necesita de un mayor consumo de recursos (debido a que Windows consume más de RAM y CPU, limitando los recursos para el juego) y que las licencias de Windows son más caras que las de Linux, vamos a optar por una máquina Ubuntu.

Debido a que Ubuntu es un sistema operativo basado en Linux, consume menos recursos como hemos mencionado, lo que nos permite asignar más capacidad centrada a nuestro servidor, además que permite tener un mejor rendimiento en entornos de servidores por su eficiencia en el manejo de recursos, lo cual es clave para servidores de juegos, donde cada recurso cuenta para garantizar el mejor rendimiento posible. Como curiosidad añadida, Linux es el entorno más utilizado para el despliegue de servidores de Minecraft, debido además a su estabilidad y facilidad de mantenimiento, siendo consecuencia de esto que las herramientas y "plugins" más populares para servidores de Minecraft estén optimizados para Linux. También es interesante en temas de escalabilidad y flexibilidad, pero sabiendo que es para un grupo reducido de amigos, nos estaríamos saliendo ya mucho del tema.

Elección del tamaño de la máquina virtual

Una vez hemos elegido nuestro sistema operativo, la siguiente decisión será elegir el tamaño de la máquina virtual, puesto a que esto jugará un papel clave en los recursos disponibles, y muy importante en el costo de esta.

Para un servidor pequeño (de hasta 10 jugadores) lo recomendable es elegir una opción de tamaño de la serie B v2 (aunque la serie B también sirva, aunque con menor rendimiento), puesto a que sería suficiente y con su bajo costo ofrece adecuado rendimiento.

Para servidores con más personas (de entre 10 o 30 como máximo) se recomienda una configuración de tamaño de la serie D, como es la D2s v4, que ofrece un buen equilibrio entre CPU, memoria y rendimiento de almacenamiento, clave para que funcione sin problemas.

Si quisiéramos ya un servidor amplio, en donde el volumen de jugadores ya sea bastante más amplio, necesitaremos de mucho más procesamiento, entonces nos quedaríamos con una máquina de la serie F como F2s o F4s, por ejemplo, que proporcionan más núcleos de CPU.

Como ya hemos dicho que será para un grupo reducido de amigos, usaremos una máquina de tamaño de la serie B v2 como es b2as_v2, que nos sería suficiente.

Elección del almacenamiento

Necesitaremos de almacenamiento para los archivos de nuestro servidor de Minecraft, incluyendo el mundo u otros datos. La mejor opción es hacer uso de discos premium SSD, puesto a que un servidor de juegos tiene normalmente mucha interacción, generando lecturas y escrituras frecuentes en el disco, y este tipo de disco nos brinda de una baja latencia y son ideales para una alta carga de trabajo. Otras opciones serian hacer uso de un disco HDD, es decir, un disco duro tradicional, que son más económicos, pero que afectaría considerablemente en el rendimiento. Por último, está la opción de un disco Standard SSD, que tiene también un costo inferior a los premium y son la opción intermedia en cuanto a rendimiento, pero teniendo en cuenta el dato tratado anteriormente de que somos pocos usándolo, sería suficiente y se ajustaría a un presupuesto aceptable.

Luego estaría la opción de usar el almacenamiento en Azure, pero es más recomendable su uso para almacenar copias de seguridad de mundos, o incluso mods; que no necesitan estar directamente en el servidor.

Respecto a la capacidad de almacenamiento, es un tema delicado, puesto a que un mundo de Minecraft (espacio de juego), cuando más rodado este, más va a crecer su tamaño. Por lo tanto, hay que tener muy en cuenta este factor a la hora de elegir la capacidad que vamos a soportar. Teniendo esto en cuenta, un disco de 30GB puede resultar como un muy buen punto de partida, el cual podremos ampliar más adelante si se da el caso, pero debería ser más que suficiente. Esto aplica al resto de recursos, como CPU o RAM, el cual podremos ampliar posteriormente si es necesario.

Configuración de red

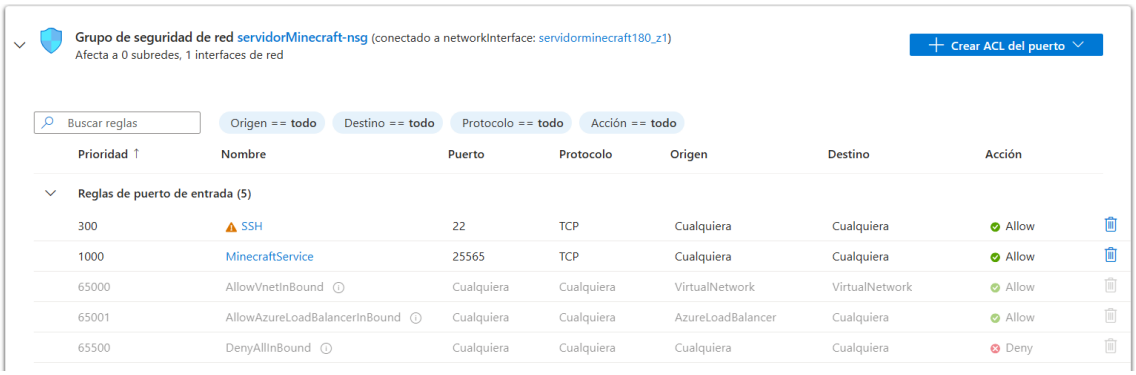
Sera necesario abrir el puerto 25565 en el firewall de Azure para permitir la conexión al servidor de Minecraft desde el exterior, además de tener una IP pública.

También será necesario configurar el acceso remoto mediante SSH para poder gestionar la VM desde cualquier lugar.

Una vez acabada la configuración para la creación de nuestra máquina virtual, la crearemos, y pasaremos a la fase de instalación de todo el material.

Fase de Instalación

Una vez creada la máquina virtual, será necesario abrir el puerto 25565 para ofrecer el servicio del servidor a nuestros amigos o demás usuarios.



Una vez hecho este paso previo, arrancamos la máquina y nos conectamos vía SSH:

```
PS C:\Users\diego> ssh kupai@20.108.32.97
```

Una vez dentro, el primer paso es, debido a que es una maquina recién creada, vamos a ejecutar `$ sudo apt update` y `$ sudo apt full-upgrade` para actualizar todos los paquetes actuales a su última versión. Posteriormente, instalar Java para jugar a Minecraft Java Edition, quien lo diría. Para ello ejecutamos lo siguiente:

```
kupai@servidorMinecraft:~$ sudo apt install openjdk-21-jdk-headless
```

Con esto instalamos la versión más optimizada y con soporte extendido para nuestro caso, que nos permita jugar a la última versión de Minecraft, junto a que no incluya la parte grafica con “headless”, debido a que los servidores no la necesitan puesto que sea por la línea de comandos.

Podemos revisar que se haya instalado correctamente de la siguiente manera:

```
kupai@servidorMinecraft:~$ java -version
openjdk version "21.0.6" 2025-01-21
OpenJDK Runtime Environment (build 21.0.6+7-Ubuntu-124.04.1)
OpenJDK 64-Bit Server VM (build 21.0.6+7-Ubuntu-124.04.1, mixed mode, sharing)
```

Debido a que ahora no vamos a meternos en temas complejos de modificaciones del mundo como mods, paquetes de texturas y demás puesto a que no van con la temática del trabajo, vamos a instalar únicamente la última versión disponible de Minecraft con el siguiente comando:

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

En caso de realizar esto en tiempos futuros, revisar el ultimo enlace junto a la última versión disponibles en la página oficial de minecraft.net https://www.minecraft.net/es-es/download/server. El resultado de la ejecución del comando es la siguiente:

```
kupai@servidorMinecraft:~$ wget https://piston-data.mojang.com/v1/objects/e6ec2
f64e6080b9b5d9b471b291c33cc7f509733/server.jar
--2025-03-25 20:18:29--  https://piston-data.mojang.com/v1/objects/e6ec2f64e608
0b9b5d9b471b291c33cc7f509733/server.jar
Resolving piston-data.mojang.com (piston-data.mojang.com)... 13.107.246.64, 262
0:1ec:bdf::64
Connecting to piston-data.mojang.com (piston-data.mojang.com)|13.107.246.64|:44
3... connected.
HTTP request sent, awaiting response... 200 OK
Length: 57269758 (55M) [application/octet-stream]
Saving to: ‘server.jar’

server.jar          100%[===================>]  54.62M   140MB/s    in 0.4s

2025-03-25 20:18:29 (140 MB/s) - ‘server.jar’ saved [57269758/57269758]
```

Y con el “.jar” instalado, iniciamos el servidor con:

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

Seguramente de traza del error, porque, aunque se generan los archivos, no están configurados correctamente. El que deberemos modificar seguro es el “eula.txt”, que es el **Acuerdo de Licencia de Usuario Final** (*End User License Agreement*), el cual deberos aceptar para desplegar nuestro servidor. Se hace de la siguiente manera:

```
#By changing the setting below to TRUE you are indicating your agreement to our
 EULA (https://aka.ms/MinecraftEULA).
#Tue Mar 25 20:22:34 UTC 2025
eula=false
~
```

```
#By changing the setting below to TRUE you are indicating your agreement to our
 EULA (https://aka.ms/MinecraftEULA).
#Tue Mar 25 20:22:34 UTC 2025
eula=true
```

Una vez cambiado eso, debería estar todo.

(En caso de que ocurra algún otro error, seria revisar el fichero "*server.properties*", porque puede que este desajustado a nuestras preferencias).

Con todo esto, volvemos a intentar arrancar el servidor:

```
kupai@servidorMinecraft:~$ java -Xmx1024M -Xms1024M -jar server.jar
Starting net.minecraft.server.Main
```

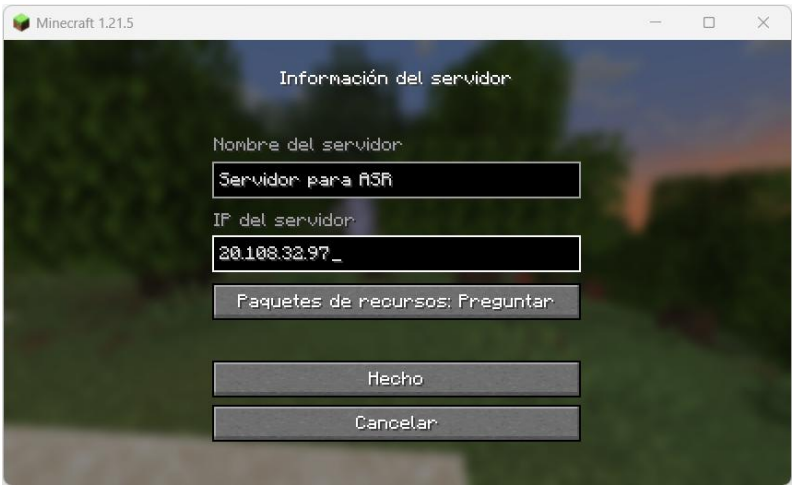
```
kupai@servidorMinecraft:~$ java -Xmx1024M -Xms1024M -jar server.jar
Starting net.minecraft.server.Main
[20:28:49] [ServerMain/INFO]: Environment: Environment[sessionHost=https://sess
ionserver.mojang.com, servicesHost=https://api.minecraftservices.com, name=PROD
]
[20:28:50] [ServerMain/INFO]: No existing world data, creating new world
[20:28:51] [ServerMain/INFO]: Loaded 1373 recipes
[20:28:51] [ServerMain/INFO]: Loaded 1484 advancements
[20:28:52] [Server thread/INFO]: Starting minecraft server version 1.21.5
[20:28:52] [Server thread/INFO]: Loading properties
[20:28:52] [Server thread/INFO]: Default game type: SURVIVAL
[20:28:52] [Server thread/INFO]: Generating keypair
[20:28:52] [Server thread/INFO]: Starting Minecraft server on *:25565
[20:28:52] [Server thread/INFO]: Using epoll channel type
[20:28:52] [Server thread/INFO]: Preparing level "world"
[20:29:03] [Server thread/INFO]: Preparing start region for dimension minecraft
:overworld
[20:29:03] [Worker-Main-2/INFO]: Preparing spawn area: 2%
[20:29:03] [Worker-Main-2/INFO]: Preparing spawn area: 2%
[20:29:04] [Worker-Main-2/INFO]: Preparing spawn area: 2%
[20:29:04] [Worker-Main-2/INFO]: Preparing spawn area: 2%
[20:29:05] [Worker-Main-2/INFO]: Preparing spawn area: 2%
[20:29:05] [Worker-Main-2/INFO]: Preparing spawn area: 2%
[20:29:06] [Worker-Main-2/INFO]: Preparing spawn area: 2%
[20:29:06] [Worker-Main-2/INFO]: Preparing spawn area: 2%
[20:29:07] [Worker-Main-2/INFO]: Preparing spawn area: 2%
[20:29:07] [Worker-Main-2/INFO]: Preparing spawn area: 2%
[20:29:08] [Worker-Main-2/INFO]: Preparing spawn area: 2%
[20:29:08] [Worker-Main-2/INFO]: Preparing spawn area: 2%
[20:29:09] [Worker-Main-2/INFO]: Preparing spawn area: 2%
[20:29:09] [Worker-Main-2/INFO]: Preparing spawn area: 2%
[20:29:10] [Worker-Main-2/INFO]: Preparing spawn area: 2%
[20:29:10] [Worker-Main-2/INFO]: Preparing spawn area: 2%
[20:29:11] [Worker-Main-2/INFO]: Preparing spawn area: 2%
[20:29:11] [Server thread/INFO]: Time elapsed: 8380 ms
[20:29:11] [Server thread/INFO]: Done (19.031s)! For help, type "help"
```

Para quien no lo entienda le hago un resumen de todo lo que muestra. Se está arrancando el servidor con la versión 1.21.5 que instalamos previamente, y como es un servidor nuevo, "No existing world data, creating new world" quiere decir que está generando el nuevo mundo junto a la zona de aparición. Esto sería lo que se vería en la primera instancia de arranque del servidor.

Conexión con el servidor

Por último, será necesario conectarnos al servidor, para ello, abrimos Minecraft Launcher en nuestro ordenador, tanto portátil como de sobremesa es válido. Desde ahí, vamos a seccionar la versión de "Java Edition", puesto que es la que elegimos que soportase nuestro servidor; y escogemos la última versión disponible (en nuestro caso la 1.21.5) y le damos a "Jugar". Con esto entraremos en la pestaña del juego como tal.

Una vez dentro del juego, vamos a "Multijugador" y le damos a la opción de "Añadir servidor", nos pedirá un nombre para asignarle (el que nosotros queramos) y la dirección para la conexión, lo añadimos y le damos a "Hecho", y ya la propia aplicación buscará la disponibilidad del servidor.



Posteriormente, tendremos desde la terminal del servidor, control de lo que sucede en el servidor en todo momento:

```
[21:28:32] [Server thread/INFO]: DieguKupai[/79.116.131.88:16934] logged in wit
h entity id 91 at (-5.799352625826562, 76.0, 43.75293101324583)
[21:28:32] [Server thread/INFO]: DieguKupai joined the game
[21:30:44] [User Authenticator #2/INFO]: UUID of player Zehioo is 8e568d0f-ace5
-4d62-8cb2-5e5536c73303
[21:30:44] [Server thread/INFO]: Zehioo[/83.54.203.156:57390] logged in with en
tity id 422 at (-7.5, 77.0, 53.5)
[21:30:44] [Server thread/INFO]: Zehioo joined the game
```

Para apagar el servidor de Minecraft, bastara con usar *Control + C* y se pausara automáticamente. Este proceso tarda unos segundos debido a que se apaga de forma controlada, es decir, realizara un proceso de guardado de datos.

Una vez hemos arrancado por primera vez el servidor, miramos los archivos que se han generado con `ls`:

```
kupai@servidorMinecraft:~$ ls
banned-ips.json      libraries  server.jar         versions
banned-players.json  logs       server.properties  whitelist.json
eula.txt             ops.json   usercache.json     world
```

Vemos que el mundo es un directorio, que contiene todos sus archivos asociados. Lo mismo pasa con las librerías (que es donde estarían incluidos los mods en caso de haber alguno). El resto de ficheros son propiedades del servidor, interesante solo la “whitelist.json”, que permite almacenar los jugadores que tienen permitidos unirse al servidor y “banned-players.json” junto a “banned-ips.json” que como su nombre indica, permite evitar el acceso a algún usuario. También está el directorio logs que es donde se guardan todos los logs de lo que sucede en el mundo, lo cual es vital para llevar un control que facilite el mantenimiento.

Una vez que queramos volver a conectarnos, volvemos a ejecutar el mismo comando que usamos anteriormente, la diferencia es que el mundo ya estará creado, y esta vez mostrará esto por pantalla:

```
kupai@servidorMinecraft:~$ java -Xmx1024M -Xms1024M -jar server.jar
Starting net.minecraft.server.Main
[21:34:18] [ServerMain/INFO]: Environment: Environment[sessionHost=https://sess
ionserver.mojang.com, servicesHost=https://api.minecraftservices.com, name=PROD
]
[21:34:21] [ServerMain/INFO]: Loaded 1373 recipes
[21:34:21] [ServerMain/INFO]: Loaded 1484 advancements
[21:34:21] [Server thread/INFO]: Starting minecraft server version 1.21.5
[21:34:21] [Server thread/INFO]: Loading properties
[21:34:21] [Server thread/INFO]: Default game type: SURVIVAL
[21:34:21] [Server thread/INFO]: Generating keypair
[21:34:21] [Server thread/INFO]: Starting Minecraft server on *:25565
[21:34:21] [Server thread/INFO]: Using epoll channel type
[21:34:22] [Server thread/INFO]: Preparing level "world"
[21:34:22] [Server thread/INFO]: Preparing start region for dimension minecraft
:overworld
[21:34:27] [Worker-Main-1/INFO]: Preparing spawn area: 0%
[21:34:27] [Worker-Main-1/INFO]: Preparing spawn area: 0%
[21:34:27] [Worker-Main-1/INFO]: Preparing spawn area: 0%
[21:34:27] [Worker-Main-1/INFO]: Preparing spawn area: 0%
[21:34:27] [Worker-Main-1/INFO]: Preparing spawn area: 0%
[21:34:27] [Worker-Main-1/INFO]: Preparing spawn area: 0%
[21:34:27] [Worker-Main-1/INFO]: Preparing spawn area: 0%
[21:34:27] [Worker-Main-1/INFO]: Preparing spawn area: 0%
[21:34:27] [Worker-Main-1/INFO]: Preparing spawn area: 0%
[21:34:27] [Worker-Main-1/INFO]: Preparing spawn area: 0%
[21:34:27] [Worker-Main-1/INFO]: Preparing spawn area: 0%
[21:34:27] [Server thread/INFO]: Time elapsed: 5524 ms
[21:34:27] [Server thread/INFO]: Done (5.936s)! For help, type "help"
```

Como se aprecia, al ya existir los elementos de “world” no los genera nuevamente, el resto de los elementos los prepara para su uso por los usuarios.

Mientras tanto, los jugadores ya pueden estar disfrutando del servicio:

Mejoras del servidor

Una propuesta muy interesante sería un par de mejoras, que a nivel de rendimiento no tienen influencia, puesto a que el rendimiento con la configuración anteriormente estudiada y aplicada ya es más que suficiente; tienen una influencia a nivel de usuario.

Servicio DNS

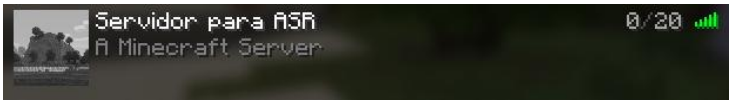
Para compartir nuestro servidor, será necesario de compartir la dirección IP para conectarnos a la máquina que está ofreciendo el servicio, que pasa, que una dirección IP debido a su naturaleza numérica y abstracta, son inherentemente difíciles de recordad. Este fenómeno se debe a la limitación cognitiva humana, y una forma de solventarlo, es asociándole un nombre de servidor para evitar que los usuarios tengan que copiar la dirección IP o buscarla cada vez.

Para ello vemos las alternativas reales para generar un dominio. Al estar utilizando Azure como plataforma para nuestra máquina Linux, este nos ofrece un servicio de dominios completamente gratuito y temporal, es decir, el dominio estará activo mientras este activa la dirección IP que se asocia. Al ser gratuita puede ser una opción, lo que sucede con este dominio es que cumple con un formato lo suficientemente largo como para seguir siendo difícil de recordar. El formato es "*tunombre*.*region*.*cloudapp.azure.com*", y como hay opciones mejores mostrare otras. (En el material auxiliar se adjunta el proceso para configurar el servicio DNS desde Azure).

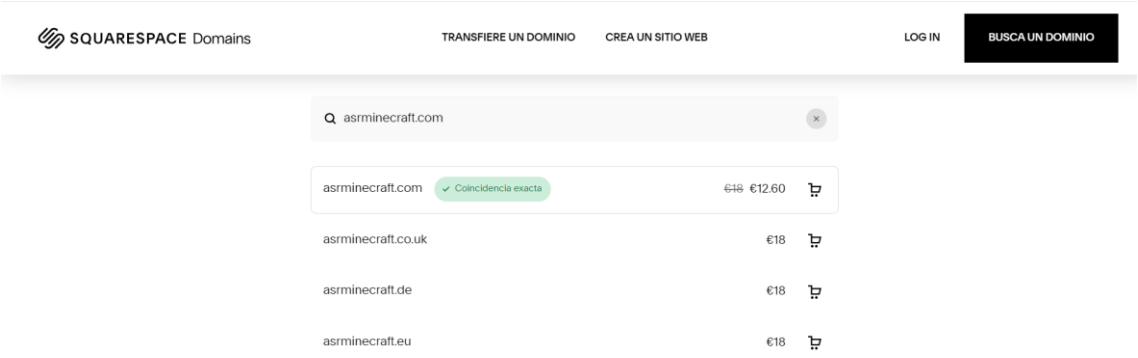
Otra alternativa consistiría en utilizar un dominio de https://www.duckdns.org/, que ofrece también un servicio gratuito. Crearlo es una tarea simple, solo iniciar sesión con Google, GitHub u otra opción de las que dispone, y luego creas el dominio directamente en "add domain":

Será necesario actualizar la IP de "current ip" a la de la máquina virtual donde está instalado nuestro servidor de Minecraft. Lo malo de este dominio es nuevamente, que, al ser gratuito, es un tanto complejo nuevamente, pero es una mejor alternativa al producto de Azure.

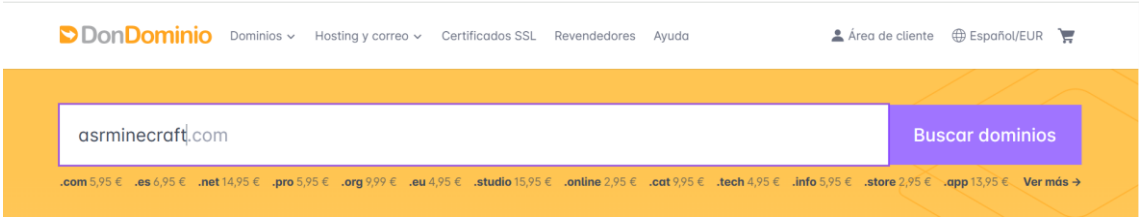
Y a la hora de "Añadir servidor", agregamos en vez de la dirección IP, escribimos "asrminecraft.duckdns.org" y al agregar servidor, veremos que hemos hecho una conexión de igual manera que directamente con la dirección IP.



Otras alternativas, de pago, que son la mejor opción en caso de hacer un servidor a nivel profesional, para que miles de personas de todo el mundo puedan acceder sería utilizar un servicio como Google Domains (actualmente se llama Squarespace Domains), pero tiene un coste elevado que no valdría la pena para servidores a pequeñas escalas.



Otra alternativa muy interesante es DonDominio, que es una empresa española que ofrece dominios a un precio mucho más económico que Google Domains.



Con ambas opciones de pago o incluso infinidad de ellas más que se encuentran disponibles en la web, se podrá crear un dominio fácil de aprender, como podría ser "*asrminecraft.com*". Es importante recalcar que dependiendo del TLD (Top-Level Domain, como ".es" o ".com") elegido, el precio puede variar.

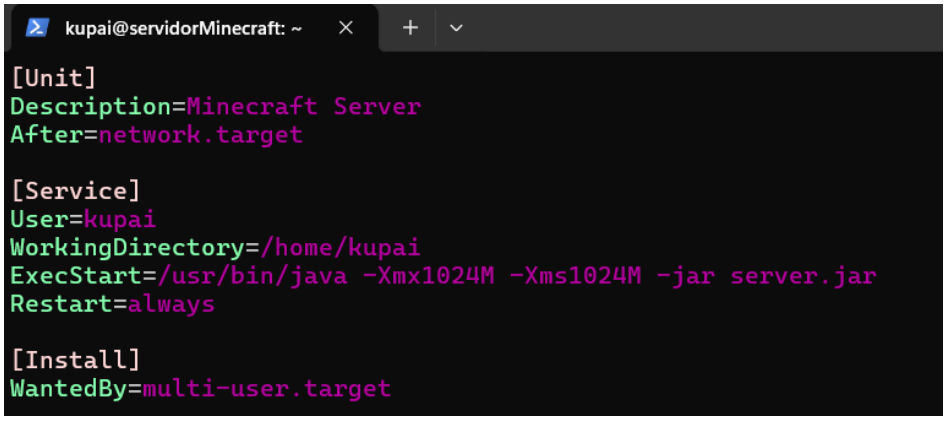
Mantener el Servidor Activo sin Conexión Remota

El principal problema de este servidor es que, al tener que arrancar la máquina desde ssh, es necesario que el anfitrión o dueño del servidor lo despliegue manualmente, resultando en que el servidor sea dependiente de que este alguien manejándolo. Para solucionarlo, vamos a configurar la máquina virtual para que, una vez se encienda, arranque directamente el servidor.

Para ello:

Crear un archivo de servicio “systemd” con la siguiente configuración:

```
sudo vi /etc/systemd/system/minecraft.service
```



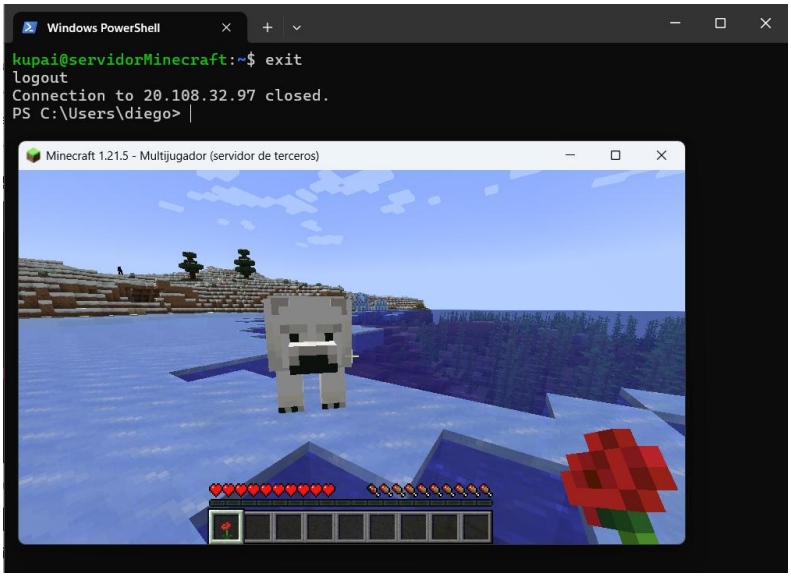
Recargar los servicios de systemd para que el sistema reconozca el archivo del servicio:

```
kupai@servidorMinecraft:~$ sudo systemctl daemon-reload
```

Y habilitamos el servicio para que se inicie automáticamente al arrancar la máquina:

```
kupai@servidorMinecraft:~$ sudo systemctl enable minecraft
Created symlink /etc/systemd/system/multi-user.target.wants/minecraft.service →
/etc/systemd/system/minecraft.service.
```

Ahora, aunque yo cierre o apague el ordenador, el servidor seguirá funcionando sin necesidad de tener la conexión ssh activada:



Multiservidor de Minecraft

Hasta ahora estábamos corriendo un único servidor de Minecraft, es decir, solo ofrecemos acceso a un único mundo a nuestros usuarios. Por lo tanto, ¿Cómo se puede hacer para desde la misma máquina, tener más de un servidor disponible?

Esto es un proceso relativamente fácil, lo que tenemos que hacer es clonar el directorio donde teníamos el servidor junto a sus archivos de propiedades, a otro directorio nuevo.

```
kupai@servidorMinecraft:~$ ls
minecraft_server_1  minecraft_server_2
kupai@servidorMinecraft:~$ cd minecraft_server_1
kupai@servidorMinecraft:~/minecraft_server_1$ ls
banned-ips.json      eula.txt   logs      server.jar         usercache.json  whitelist.json
banned-players.json  libraries  ops.json  server.properties  versions        world
kupai@servidorMinecraft:~/minecraft_server_1$ cd ../minecraft_server_2
kupai@servidorMinecraft:~/minecraft_server_2$ ls
banned-ips.json      eula.txt   logs      server.jar         usercache.json  whitelist.json
banned-players.json  libraries  ops.json  server.properties  versions        world
kupai@servidorMinecraft:~/minecraft_server_2$
```

Lo que tendremos que hacer ahora será modificar el fichero “server.properties” del “minecraft_server_2” para que no estén ambos usando el mismo puerto:

`vi server.properties`

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

```
server-ip=
server-port=25566
simulation-distance=10
```

Es IMPORTANTE abrir el puerto nuevo en el firewall de entrada desde Azure:

1000	MinecraftService	25565	TCP	Cualquiera	Cualquiera	Allow
1010	MinecraftSecondService	25566	TCP	Cualquiera	Cualquiera	Allow

Y podemos añadir ambos servidores a `/etc/systemd/system/minecraft.service` para que arranque ambos servidores a la vez:

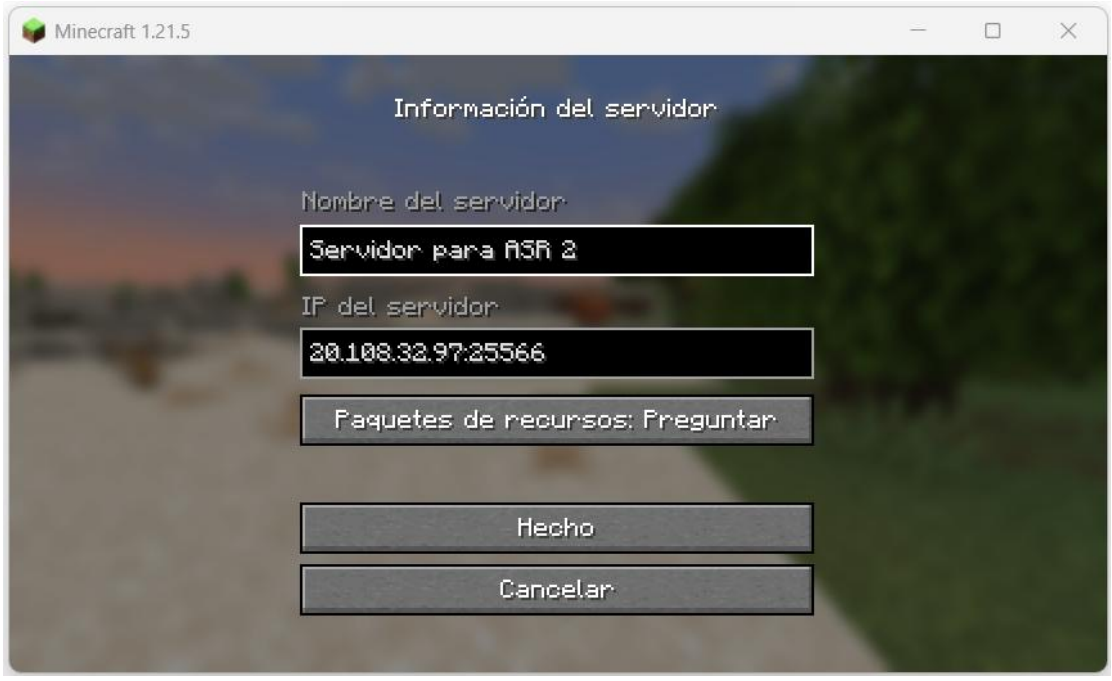
```
kupai@servidorMinecraft: ~/r
[Unit]
Description=Minecraft Servers
After=network.target

[Service]
User=kupai
WorkingDirectory=/home/kupai

# Arrancar primer servidor
ExecStart=/usr/bin/java -Xmx1024M -Xms1024M -jar minecraft_server_1/server.jar

# Arrancar segundo servidor
ExecStartPost=/usr/bin/java -Xmx1024M -Xms1024M -jar minecraft_server_2/server.jar

Restart=always

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

(Recordar recargar los servicios de systemd, además de reiniciar el servicio Minecraft igual que mostramos con anterioridad al modificar este servicio). Y con esto deberíamos tener dos servidores funcionando. Ahora para conectarnos, será necesario proporcionar la dirección IP junto al puerto de la siguiente manera (el DNS aquí es todavía más interesante):

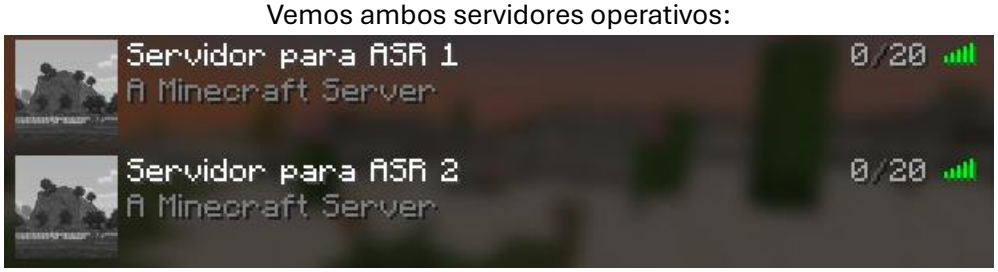
Dirección_IP:Puerto

Conectamos al primer servidor:

Conectamos al segundo servidor:



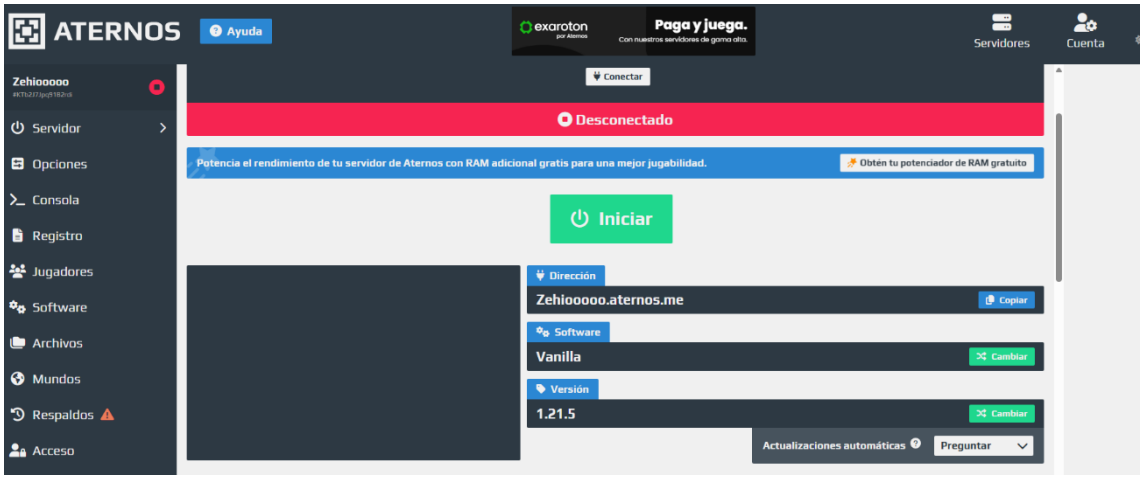
Vemos ambos servidores operativos:



Alternativas a la máquina virtual privada

Si una persona ajena a la Universidad de Oviedo desease replicar este proyecto, tendría que asumir los costes derivados del uso de un servidor en Azure. En nuestro caso, al ser estudiantes, contamos con un saldo inicial de 100 dólares proporcionado por el acuerdo con la universidad, lo que facilita alojar pequeños servidores mediante maquinas virtuales. Dado a que este servidor esta destinado a un numero reducido de usuarios y permanecerá inactivo varias horas a la semana, los gastos semanales serán moderados. Sin embargo, en caso de escalar el servidor para un publico mas amplio y disponibilidad continua, los costes podrían incrementarse considerablemente. Es por ello, que nos parece interesante ofrecer también una opción diferente para "hostear" este tipo de servidores:

Una opción conocida es la plataforma Aternos, que ofrece la posibilidad de crear servidores personalizados de manera parcial, y lo mas importante, de forma gratuita. Esta característica la convierte en la opción mas viable actualmente a nivel de mercado.



Uno de los principales inconvenientes de Aternos, es la presencia de publicidad (adware) en su página, lo que puede resultar molesto e incluso presentar riesgos potenciales para la seguridad (nada es gratis). Sin embargo, el problema mas significativo al utilizar plataformas gratuitas como Aternos radica en la latencia elevada del servidor, causada principalmente por dos factores. El primero, es que un servidor en Azure privado reserva todos los recursos como son la CPU, RAM y ancho de banda para nuestra máquina virtual, asegurando un buen rendimiento. En cambio, en Aternos, los recursos son compartidos con otros usuarios, lo que ocasiona múltiples caídas de rendimiento frecuentes. El segundo inconveniente es la estabilidad y escalabilidad, puesto a que en nuestra maquina podemos escalar los recursos en tiempo real según la demanda, mientras que en Aternos están limitados.

Por lo tanto, para un servidor como el que pusimos de ejemplo, para jugar con amigos, si no es un caso en donde realmente necesites asegurar una buena latencia junto a un buen rendimiento, compensa infinitamente mas Aternos frente a un servidor propio. En cambio, si se quisiera hacer un servidor profesional, como los ofrecidos globalmente que juegan millones de jugadores al día, una opción gratuita no es viable.

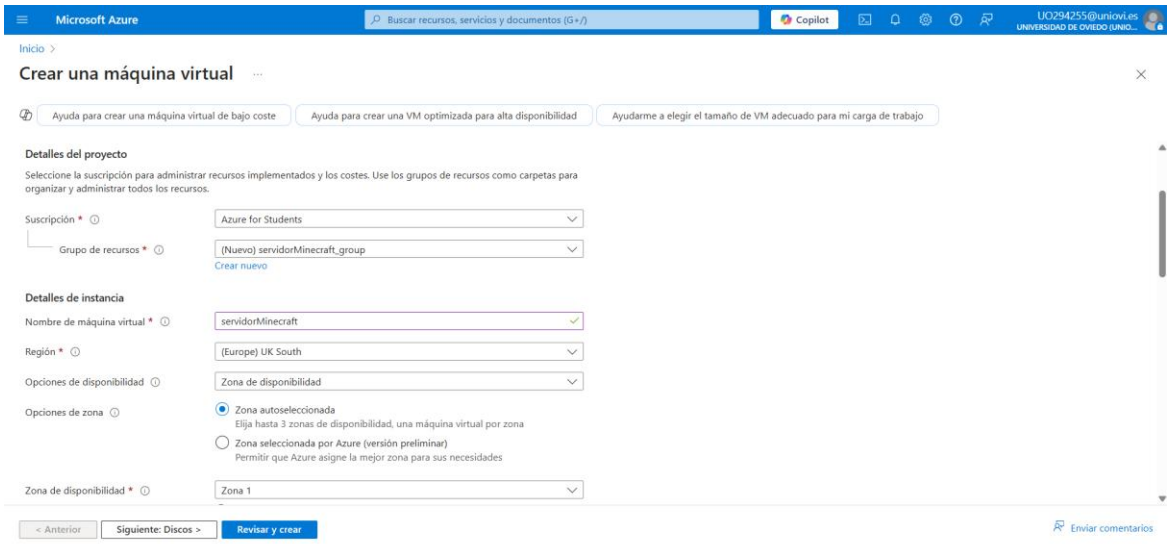
Otra alternativa gratuita es Minehut. También hay opciones similares de pago que prometen un mejor rendimiento como Shockbyte o Apex Hosting, pero para pagar de todas formas, me quedo con mi propia máquina virtual, y no pagar extras por el servicio.

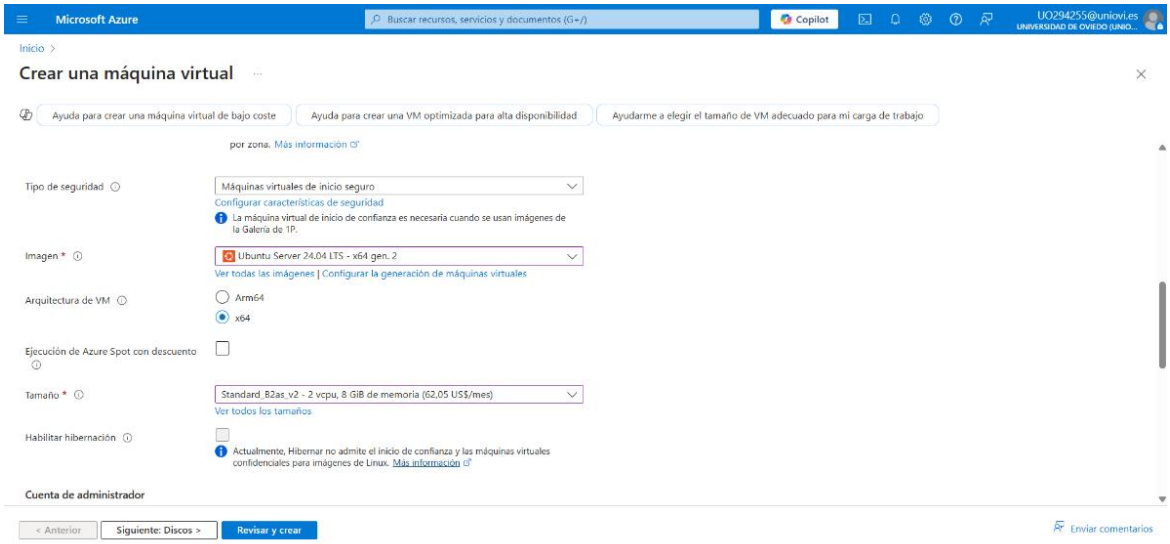
Material auxiliar

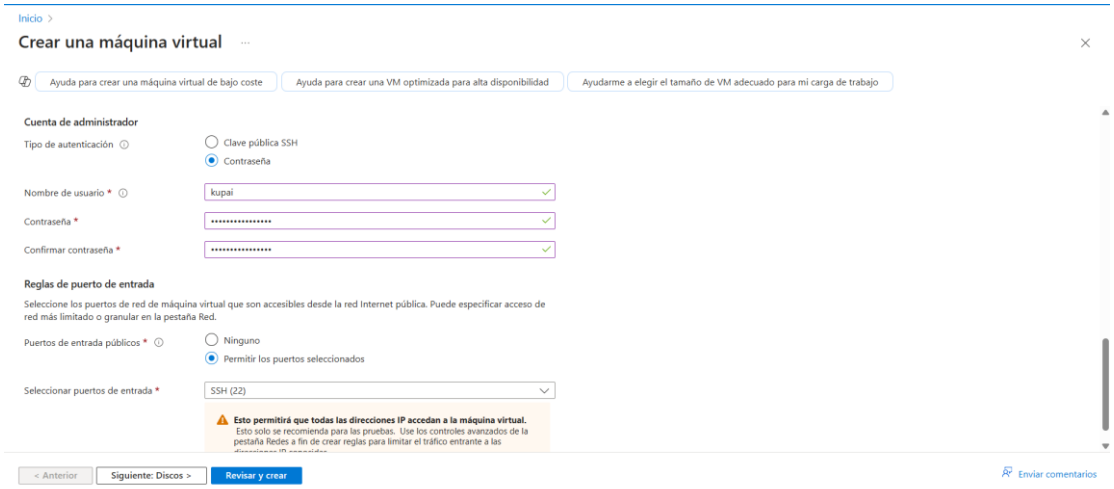
Para ayudar al proceso de instalación, se adjuntan capturas de los distintos procesos realizados en la instalación del servidor:

Configuración máquina virtual

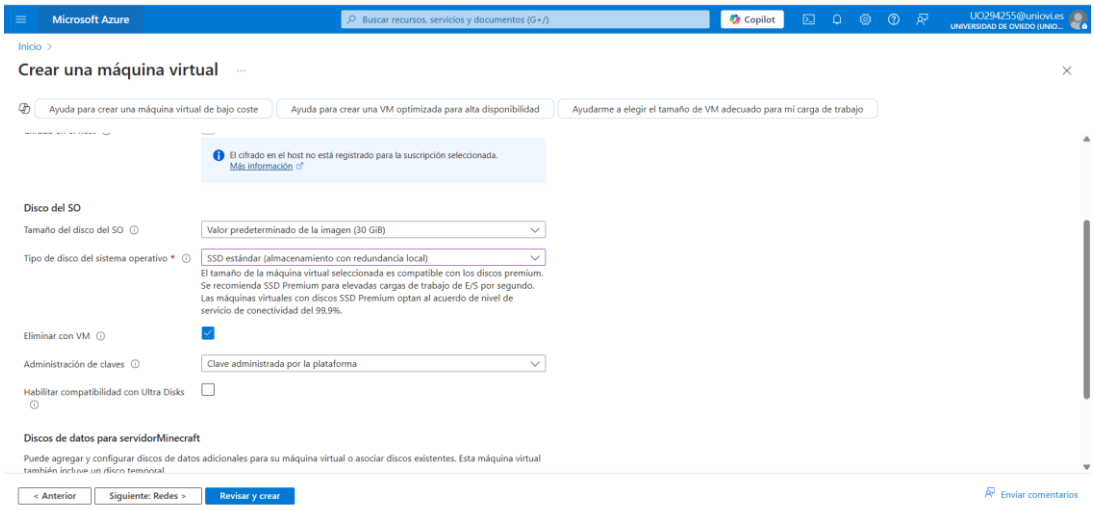
Todos los detalles de la instancia:



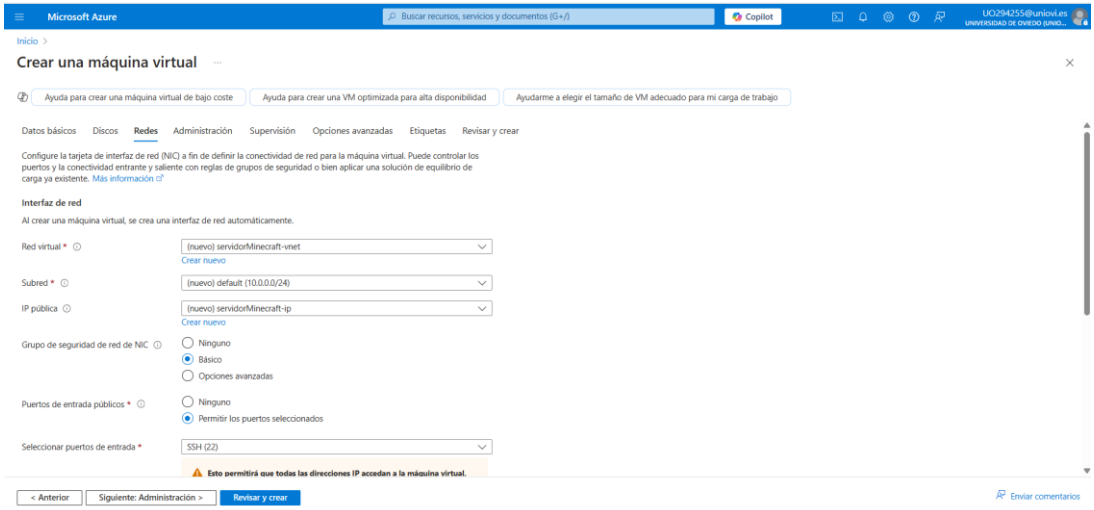




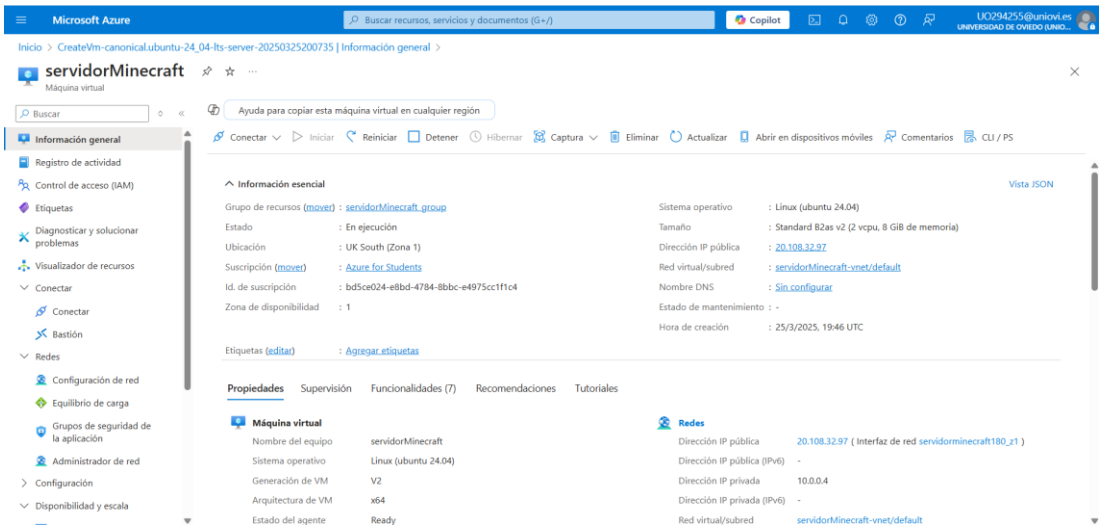
Configuración del disco:



Configuración red:



Resumen de la máquina:



Configuración DNS Azure