**Procedimiento para crear un VM en Azure y conectarnos a ella**

**Crear la VM desde el portal (sin usar comandos)**

1. Tenemos que ir a la pagina principal y le damos a create resource, y en lo que nos sale buscamos Virtual Machines

Interfaz de usuario gráfica, Aplicación, Word

El contenido generado por IA puede ser incorrecto.

1. Pestaña *Basics*
   * **Name**: milvus-vm
   * **Image**: *Ubuntu 22.04 LTS*
   * **Size**: para pruebas basta *Standard\_D4s\_v5* (4 vCPU / 16 GB).
   * **Authentication type**: *SSH public key*. Azure genera una clave publica que es la que vams a guardar en nuestra computadora. [learn.microsoft.com](https://learn.microsoft.com/en-us/azure/virtual-machines/linux-vm-connect?utm_source=chatgpt.com)
2. Avanza hasta *Networking* y deja la IP pública como “dinámica” (puedes cambiarla a estática después).
3. Pulsar **Review + Create → Create**. Tras ~2 min tendrás la VM y verás su **IP pública**.

**5- Después que creemos la virtual machine, entramos en ella para crear los puertos que Milvus necesita y que se consultaran a través de SSH, Ver imagen debajo**

**Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.**

Una vez que tenemos creada la VM y configurados lo puertos necesario para que milvus funcione entonces tenemos que ser capaz de conectarnos usando PowerShell en Windows. Para poder realizar esta conexión tenemos que tener localizada la llave publica que genero azure para poder acceder a la VM y además configurar los permisos de la llave para que solo un usuario pueda acceder a ella y leerla:

1. Vamos a donde tenemos localizada la llave publica, le damos click derecho -> propiedades -> Seguridad -> Advanzado -> Deshabilitar Herencia
   1. Esto lo que hace es eliminar cada uno de los usuarios que tenían acceso a este archivo
   2. Agregamos nuestro usuario
   3. Ponemos permisos de solo lectura

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

1. Ahora vamos a conectarnos a la VM mediante Poweshell
   1. Abrimos powershell como administrador y cambiamos la ruta a donde tenemos la llave publica:

1.  cd ruta\a\tu\llave\publica

* 1. Posteriormente ponemos el siguiente comando para poder acceder a la VM

1.ssh -i .\<**nombre\_tu\_llave\_publica.pem**> azureuser@<**tu\_IP**>

* 1. Con lo anterior debería aparcer algo así en la consola de powershell

Texto

El contenido generado por IA puede ser incorrecto.

**Comandos importantes**

//Se usa para verificar el estado del almacenamiento, cuanto esta usado y cuanto queda libre

1. df -h /

//Se usa para la actualización de los paquetes

1. sudo apt update

//Instalar los paquetes más recientes

1. sudo apt upgrade -y

//Limpiar dependencias obsoletas

1. sudo apt autoremove -y

//Reinicio seguro (si aparece /var/run/reboot-required)

1. sudo reboot

**Instalar Docker Engine y Docker Compose**

Docker Engine es el servicio que ejecuta contenedores; Docker Compose v2 es un complemento del propio cliente Docker que orquesta varios contenedores con un único archivo YAML [docs.docker.com](https://docs.docker.com/engine/install/ubuntu/?utm_source=chatgpt.com). Ambos se distribuyen a través del repositorio oficial de Docker para Ubuntu, por lo que primero añadiremos la clave GPG y el repositorio, después instalaremos los paquetes.

* **Requisitos previos**

//Estos paquetes permiten descargar la clave GPG y gestionar repositorios APT

1. sudo apt install ca-certificates curl gnupg lsb-release -y

* **Añadir la clave GPG y el repositorio de Docker**

//La clave autenticará los paquetes; el archivo .list apunta al repositorio estable de Docker para la versión Ubuntu

sudo install -m 0755 -d /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | \

sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

echo \

"deb [arch=$(dpkg --print-architecture) \

signed-by=/etc/apt/keyrings/docker.gpg] \

https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

1. sudo apt update

* **Instalar Docker Engine**

//docker-ce es la edición comunitaria, containerd.io el runtime subyacente docs.docker.comdigitalocean.com

1. sudo apt install docker-ce docker-ce-cli containerd.io -y

//Habilitar el servicio

sudo systemctl enable docker

sudo systemctl start docker

sudo systemctl status docker

* **Ejecutamos Docker sin sudo**

// Esto añade tu cuenta al grupo docker, permitiendo usar el cliente sin privilegios de administrador

2. sudo usermod -aG docker $USER

**Al hacer este paso debemos salir de la sesión con exit y volver a ejecutarla y hacemos:**

//Con esto comprobamos que Docker este corriendo perfectamente

1. docker run hello-world

* **Instalamos Docker Compose v2**

El paquete docker-compose-plugin integra docker compose en el CLI estándar [docs.docker.com](https://docs.docker.com/compose/install/linux/?utm_source=chatgpt.com) [docs.docker.com](https://docs.docker.com/compose/install/?utm_source=chatgpt.com).

1. sudo apt install docker-compose-plugin -y

// Comprobamos la version

1. docker compose version

* **Probar Docker Compose**

Crea un directorio de prueba:

1. mkdir ~/compose-test && cd ~/compose-test

Escribe un archivo docker-compose.yml minimalista:

1. nano docker-compose.yml

Contenido:

1. services:

2. web:

3. image: nginx:stable

4. ports:

5. - "8080:80"

Arranca el servicio:

1. docker compose up -d

2. docker compose ps # debe mostrar el contenedor “web” HEALTHY

**Abre http://<IP\_PUBLICA>:8080 en tu navegador; verás la página de inicio de Nginx.**

**Texto

El contenido generado por IA puede ser incorrecto.**

Detén y elimina todo:

1. docker compose down

**Desplegar Milvus 2.5.13 con Docker Compose**

A continuación desplegaremos Milvus 2.5.13 Stand-alone con Docker Compose, verificaremos que quede “healthy” y probaremos la conexión con PyMilvus desde la VM. El proceso se resume en: descargar el docker-compose.yml oficial, arrancar los contenedores, comprobar su salud y ejecutar un script de prueba.

**1. Colocar el archivo docker-compose.yml oficial**

cd ~

wget https://github.com/milvus-io/milvus/releases/download/v2.5.13/milvus-standalone-docker-compose.yml \

-O docker-compose.yml

La versión v2.5.13 (10-jun-2025) es la estable más reciente y corrige fallos detectados en v2.5.12 [github.com](https://github.com/milvus-io/milvus/releases?after=v0.10.0&utm_source=chatgpt.com)[github.com](https://github.com/milvus-io/milvus/issues/42479?utm_source=chatgpt.com). El YAML monta un volumen ./volumes para persistir datos y expone los puertos 19530 (gRPC), 9091 (REST) y 9092 (HTTP UI) [milvus.io](https://milvus.io/docs/install_standalone-docker-compose.md?utm_source=chatgpt.com)[milvus.io](https://milvus.io/docs/install_standalone-docker.md?utm_source=chatgpt.com).

**Inspeccionar el archivo**

1. grep -A3 'image:' docker-compose.yml # verás milvusdb/milvus:v2.5.13-\*

**2. Arrancar Milvus**

1. docker compose up -d # descarga y lanza Milvus, etcd y MinIO

2. docker compose ps # espera a que los tres aparezcan "healthy"

Si alguno parpadea en “Restarting”, consulta docker compose logs -f milvus para ver detalles [github.com](https://github.com/milvus-io/milvus/issues/41231?utm_source=chatgpt.com).

**3. Verificar desde dentro de la VM**

**3.1 Ping de salud por REST**

1. curl http://localhost:9091/healthz

Respuesta # → {"status":"ok"}

**3.2 Prueba con PyMilvus**

python3 -m venv venv && source venv/bin/activate

pip install pymilvus==2.5.11

python - <<'PY'

from pymilvus import connections, utility

connections.connect(host="localhost", port="19530")

print("Colecciones existentes:", utility.list\_collections())

PY

Una salida Colecciones existentes: [] confirma la conexión [milvus.io](https://milvus.io/docs?utm_source=chatgpt.com)[milvus.io](https://milvus.io/blog/how-to-get-started-with-milvus.md?utm_source=chatgpt.com).

**4. Verificar desde la PC o WSL**

pip install pymilvus

python - <<'PY'

from pymilvus import connections, db

connections.connect(host="20.84.56.57", port="19530")

print(db.list\_collections())

PY

| **Tarea** | **Comando / Acción** | **Referencia** |
| --- | --- | --- |
| Ver uso de RAM/CPU de Milvus | 1. docker stats milvus-standalone | ([github.com](https://github.com/milvus-io/milvus?utm_source=chatgpt.com)) |
| Actualizar a la próxima versión | 1. docker compose pull && docker compose up -d | ([github.com](https://github.com/milvus-io/milvus/releases?utm_source=chatgpt.com)) |
| Arranque rápido sin YAML (script oficial) | 1. curl -sfL https://raw.githubusercontent.com/milvus-io/milvus/master/scripts/standalone\_embed.sh -o standalone\_embed.sh && bash standalone\_embed.sh start | ([milvus.io](https://milvus.io/docs/install_standalone-docker.md?utm_source=chatgpt.com)) |

**Ingesta de Datos con Milvus**

Para eliminar una colección de milvus

python - <<'PY'

from pymilvus import connections, utility

connections.connect("default", host="135.237.82.231", port="19530")

if utility.has\_collection("inbio\_plinian\_qa"):

utility.drop\_collection("inbio\_plinian\_qa")

print("Colección eliminada")

PY

**EJEMPLO DE COMO ESCRIBIR Y GUARDAR UN ARCHIVO**

1. **Crear/abrir el archivo**

1. nano docker-compose.yml

Esto abre *nano*; en la 3.ª línea verás el cursor listo para escribir. [linuxize.com](https://linuxize.com/post/how-to-use-nano-text-editor/?utm_source=chatgpt.com)

1. **Pegar el contenido (usa el ratón o Shift+Insert si tu terminal soporta pegar).**

1. services:

2. web:

3. image: nginx:stable

4. ports:

5. - "8080:80"

Ejemplo mínimo válido tomado de un tutorial de Compose con Nginx. [medium.com](https://medium.com/%40s_a_faruque/simplest-way-to-setup-nginx-with-docker-compose-9b19c702783a?utm_source=chatgpt.com)[geshan.com.np](https://geshan.com.np/blog/2024/03/nginx-docker-compose/?utm_source=chatgpt.com)

1. **Guardar (“Write Out”)**
   * En *nano* significa pulsar **Ctrl + O**. Aparece una línea que pregunta por el nombre de archivo. [askubuntu.com](https://askubuntu.com/questions/477603/how-to-save-a-file-using-nano?utm_source=chatgpt.com)[superuser.com](https://superuser.com/questions/1145534/how-to-save-file-in-nano-fast?utm_source=chatgpt.com)
   * Simplemente presiona Enter para confirmar docker-compose.yml.
   * Verás en la parte inferior “Wrote X lines”; eso confirma que se guardó. [nano-editor.org](https://www.nano-editor.org/dist/latest/nano.html?utm_source=chatgpt.com)
2. **Salir de nano**
   * Pulsa **Ctrl + X**. Regresarás al prompt de la terminal. [linuxize.com](https://linuxize.com/post/how-to-use-nano-text-editor/?utm_source=chatgpt.com)
3. **Verificar que el archivo quedó bien**

1. cat docker-compose.yml

El contenido debe mostrarse en pantalla tal cual lo escribiste.

Atajos útiles en nano

* **Ctrl+O** = Guardar («Write Out»).
* **Ctrl+X** = Salir.
* **Ctrl+W** = Buscar dentro del archivo.  
  Estas combinaciones aparecen en la barra al pie de nano: **el símbolo ^ significa Ctrl**. [askubuntu.com](https://askubuntu.com/questions/477603/how-to-save-a-file-using-nano?utm_source=chatgpt.com)