

# MANUAL PARA CONFIGURACIÓN VPS LINUX PARA BACK-END .NET

---

Este manual técnico guía paso a paso el proceso completo para desplegar una Web API .NET en un entorno de producción real utilizando una VPS Linux conectada a un dominio personalizado. Desde la adquisición del servicio, configuración del sistema operativo, instalación de .NET, MySQL y SQL Server, hasta la publicación de la API mediante NGINX como proxy inverso, con seguridad HTTPS y despliegue automatizado a través de GitHub Actions (CI/CD).

A lo largo del documento se explican con claridad y profundidad los siguientes aspectos:

Qué es una VPS, cómo funciona y por qué es necesaria para ejecutar proyectos .NET Core

Instalación y configuración segura del sistema (SSH, firewall, actualización de paquetes)

Instalación de MySQL Server y configuración de usuarios locales/remotos

Instalación y configuración de SQL Server, ejecución normal o por medio de Docker

Publicación de una Web API .NET como servicio de sistema (systemd)

Uso de NGINX como proxy inverso para exponer la API a internet

Conexión de la VPS a un dominio usando registros DNS

Configuración de certificados HTTPS gratuitos con Certbot + Let's Encrypt

Automatización del despliegue con GitHub Actions CI/CD

Administración de múltiples Web APIs en una misma VPS

Listado de comandos clave para mantenimiento y referencia rápida

Referencias para aquellos que necesiten ahondar en algún tema

Este recurso está pensado para desarrolladores backend, devops junior o equipos técnicos que deseen montar servicios en producción con control total del entorno, sin depender de plataformas costosas como Azure o configuraciones cerradas de hosting tradicional.

1. ¿Qué es una VPS?	6
1.1 ¿Por qué es necesaria una VPS para .Net?	6
1.2 ¿Quién presta el servicio de VPS?	7
1.3 ¿Qué sistema operativo instalar?	7
2. Adquisición del servicio	8
2.1 Opciones de VPS en Hostinger	8
2.2 Instalación del sistema operativo y acceso por consola	9
2.3 Acceso vía consola (SSH)	9
3. Configuración del sistema operativo	11
4. Instalación de MySql Server.	12
4.1 Instalar MySQL Server	12
4.1.1 Asegurar la instalación	12
4.1.2 Verificar el estado del servicio	14
4.2 Creación de usuarios MySQL	14
4.3 ¿Cómo conectarse a MySQL desde XAMPP o MySQL Workbench?	16
4.4 Crear un usuario local para uso exclusivo de la Web API	16
4.4.1 Crear base de datos y usuario local	16
4.4.2 Cadena de conexión (Connection String) en .NET	17
5. Instalación de SQL Server	17
5.1 Importar la clave GPG y agregar el repositorio	17
5.2 Instalar SQL Server	18
5.2.1 Configurar SQL Server	18
5.2.2 Verificar que el servicio esté corriendo	18
5.2.3 (Opcional) Instalar herramientas de línea de comandos	19
5.2.4 Probar conexión con sqlcmd	20
5.2.5 Permitir conexiones remotas (opcional)	20
5.2.6 Cadena de conexión en .NET	20
5.3 Alternativa: Ejecutar SQL Server con Docker (recomendado para Ubuntu 24.04+)	21
5.3.1 ¿Por qué usar Docker?	21
5.3.2 Instalar Docker	21
5.3.3 Ejecutar SQL Server en un contenedor	21
5.3.4 Verificaciones y conexión	22
6. Instalación de .Net	22
6.1 Descargar e instalar los paquetes oficiales de Microsoft	22
6.2 Instalar el runtime de ASP.NET Core	23
6.3 Verificar la instalación	23
7. Configuración de inicio de sesión sin contraseña (opcional, recomendado para CI/CD)	23

7.1 Generar clave SSH desde el equipo local	23
7.2 Copiar la clave pública a la VPS	24
7.3 Configurar la clave pública en la VPS	24
7.4 Verificar acceso sin contraseña	25
8. Despliegue manual de la Web API .NET en la VPS	26
8.1 Publicar el proyecto localmente	26
8.2 Crear la carpeta del proyecto en la VPS	27
8.3 Subir los archivos a la VPS	27
8.4 Ejecutar la API en la VPS	27
8.5 Probar un endpoint desde la VPS	27
8.6 Actualizar el proyecto	28
8.7 Limitaciones del despliegue manual	28
8.8 Recomendación: Automatizar con CI/CD	29
9. Despliegue automatizado de Web API .NET en la VPS (CI/CD)	29
9.1 Crear carpeta del proyecto en la VPS	29
9.2 Configurar los Repository Secrets en GitHub	29
9.3 Crear el archivo deploy.yml	31
9.4 Probar el despliegue automático	32
10. Configuración del servicio para inicio automático (Systemd)	33
10.1 Crear el archivo del servicio	33
10.2 Activar el servicio	34
10.3 Verificar que el servicio esté en ejecución	34
10.4 Finalizar el Deploy si está usando GitHub Actions	35
10.5 Verificar el endpoint desde la VPS	36
11. Configuración de NGINX como Proxy para solicitudes externas	36
11.1 Instalar NGINX	36
11.2 Crear el archivo de configuración para su proyecto	37
11.3 Activar el sitio y reiniciar NGINX	37
11.4 Probar el acceso desde el exterior	37
11.4 Sobre el uso de HTTPS	38
11.5 ¿Y si tengo múltiples Web APIs?	38
12. Conexión entre VPS y Dominio	38
12.1 Diferencias entre Dominio, Hosting, y VPS.	39
12.1.1 ¿Qué es un Dominio?	39
12.1.2 ¿Qué es un Hosting?	39
12.1.3 ¿Qué es una VPS?	39
12.1.4 Costos y consideraciones	40

12.2 Conexión de dominio a la VPS y configuración HTTPS con Certbot	41
12.2.1 ¿Por qué se necesita un registro DNS?	41
12.2.2 Crear registro DNS en Hostinger	41
12.2.3 ¿Y si tengo múltiples Web APIs?	43
12.3 Configurar NGINX para responder a solicitudes desde el dominio + Certificado HTTPS	45
12.3.1 Editar el archivo de configuración de NGINX	45
12.3.2 Activar la configuración en NGINX	46
12.3.3 Habilitar HTTPS con Certbot (Let's Encrypt)	46
12.3.4 Verificar renovación automática	47
13. Comandos más usados.	47
13.1 Creación de un servicio systemd para Web API .NET	47
13.2 NGINX como Proxy Inverso	48
13.3 Activar y recargar configuración NGINX	48
13.4 Eliminar sitio NGINX	49
13.5 Generación de certificados HTTPS con Certbot	49
13.6 Subir carpeta publicada desde el repositorio	49
14. Conclusión	49
14.1. ¿Qué logró al finalizar este proceso?	49
15. Referencias	50
15.1. Documentación y artículos técnicos	50
15.2 Videos recomendados	51

## **1. ¿Qué es una VPS?**

En este manual se explica desde cero la adquisición y configuración de una VPS Linux para ejecutar Web Api .Net como servicio conectado a un dominio.

Una VPS (Virtual Private Server) es una solución de infraestructura que forma parte del modelo IaaS (Infraestructura como Servicio). Consiste en un servidor físico que se divide mediante virtualización en múltiples entornos independientes, donde cada entorno es una VPS con recursos dedicados de procesamiento, memoria RAM y almacenamiento. Aunque comparten el mismo hardware físico, cada VPS funciona como un servidor autónomo, permitiendo instalar un sistema operativo propio, ejecutar aplicaciones, servicios backend y conectarse a internet, ya sea para alojamiento web, despliegue de software o cualquier otro propósito.

En términos prácticos, una VPS es como tener un computador en la nube, con la ventaja de que el proveedor se encarga del mantenimiento físico del servidor. El usuario, por su parte, tiene control total sobre su entorno virtual, incluyendo la configuración del sistema operativo. Al proteger el acceso con credenciales seguras, solo el usuario puede ingresar, lo que garantiza la confidencialidad de los datos y evita accesos no autorizados, incluso por parte del proveedor.

Además, una VPS puede escalar verticalmente: si se requiere mayor capacidad de procesamiento, memoria o almacenamiento, es posible aumentar los recursos contratados pagando por el incremento.

### **1.1 ¿Por qué es necesaria una VPS para .Net?**

Los servicios de hosting compartido tradicionales no suelen ofrecer soporte nativo para aplicaciones desarrolladas en .NET, especialmente para Web APIs basadas en .NET Core o .NET 6+. En muchos casos, incluso pagando extras o aplicando configuraciones avanzadas, la ejecución de backend .NET no es posible. Por ejemplo, en plataformas populares como Hostinger, comúnmente utilizadas por pequeñas y medianas empresas, el soporte para .NET es limitado o inexistente.

Aunque Azure ofrece una plataforma completamente integrada para ejecutar aplicaciones .NET, incluyendo servicios como App Services y Azure SQL, su costo puede resultar elevado para proyectos personales o soluciones de bajo presupuesto.

Una alternativa eficiente y flexible es el uso de una VPS, donde el usuario puede instalar el runtime de .NET, configurar su propia Web API, y administrar el entorno completo según sus necesidades. Además, es posible asociar un nombre de dominio a la dirección IP pública de la VPS para atender solicitudes web de manera profesional. Al tratarse de un sistema operativo completo, también se pueden instalar múltiples servicios adicionales, como bases

de datos, gestores de archivos, certificados SSL, y servidores de correo o FTP, funcionando tanto como servidor backend como hosting web.

*Nota: Realmente una VPS Linux no es la única alternativa rentable para ejecutar proyectos Net, también existen opciones como IIS (Internet Information Services), este manual no aborda esas opciones, pero se le recomienda revisar las otras opciones y no quedarse con la primera opción.*

## **1.2 ¿Quién presta el servicio de VPS?**

Actualmente, existen múltiples proveedores que ofrecen servicios de VPS, entre los más reconocidos se encuentran Hostinger, GoDaddy, DigitalOcean, AWS, Vultr y Linode, entre otros. Cada uno presenta distintas configuraciones, precios y niveles de soporte.

Para este manual se utilizará Hostinger como proveedor de referencia, ya que al momento de su redacción ofrece una de las mejores relaciones calidad-precio del mercado, siendo una opción accesible y confiable para desplegar servicios backend con .NET.

## **1.3 ¿Qué sistema operativo instalar?**

Una VPS permite instalar prácticamente cualquier sistema operativo, incluyendo distribuciones de Linux y versiones de Windows Server. Sin embargo, es importante tener en cuenta que los recursos de la VPS (procesamiento, memoria RAM y almacenamiento) suelen estar limitados según el plan contratado, y que aumentar dichos recursos incrementa el costo del servicio.

Por este motivo, se recomienda evitar sistemas operativos que consuman demasiados recursos, como es el caso de Windows, que puede ocupar una cantidad considerable de RAM y CPU incluso en estado inactivo, debido a los procesos del sistema y la interfaz gráfica.

En su lugar, se sugiere utilizar una distribución ligera de Linux, que aprovecha mejor los recursos disponibles y ofrece un entorno estable y eficiente para desplegar servicios backend. En este manual se utilizará Ubuntu, una distribución ampliamente utilizada en entornos de producción por su estabilidad, compatibilidad y comunidad activa.

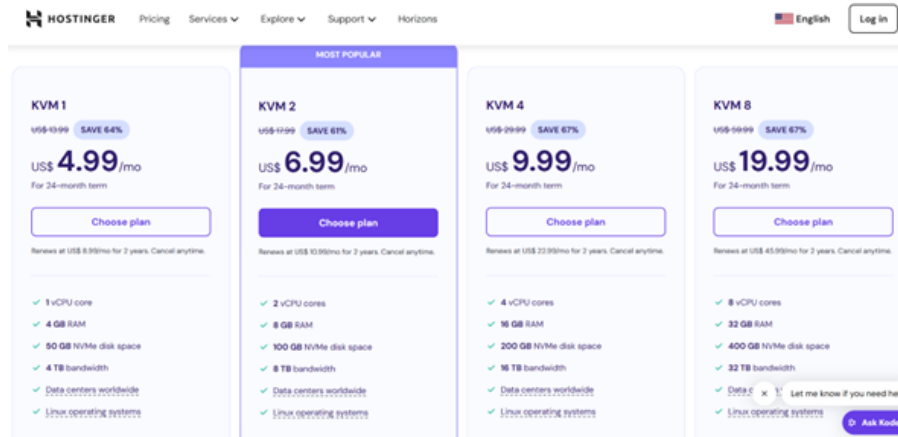
Además, no se instalará una interfaz gráfica, ya que trabajar desde la consola o terminal es el enfoque estándar en entornos profesionales, y reduce significativamente el consumo de recursos innecesarios.

## 2. Adquisición del servicio

En este manual se utilizará como referencia el servicio de VPS Linux ofrecido por Hostinger. Puede acceder a la página de planes disponibles a través del siguiente enlace:

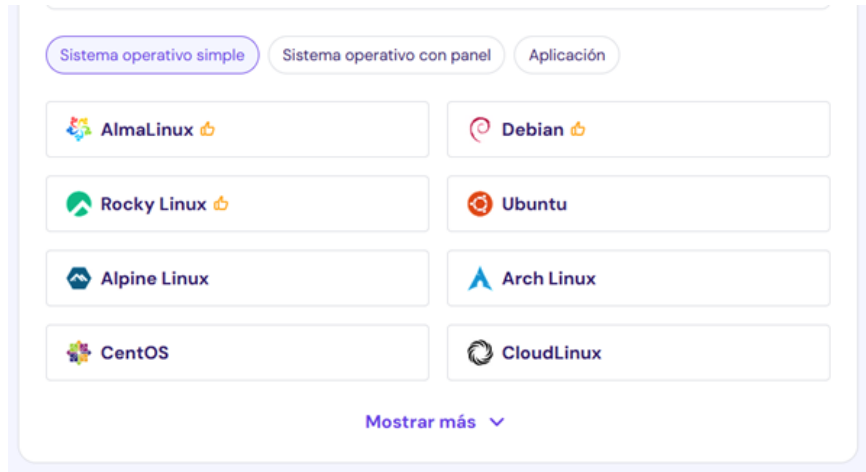
<https://www.hostinger.com/vps/linux-hosting#pricing>

### 2.1 Opciones de VPS en Hostinger



Seleccione el plan que mejor se adapte a sus necesidades según los recursos requeridos por sus aplicaciones. Si su objetivo es realizar pruebas o desplegar uno o varios proyectos pequeños, se recomienda el plan KVM 1, el cual ofrece una configuración básica pero suficiente. Más adelante, si requiere mayor capacidad de procesamiento, memoria o almacenamiento, podrá escalar a un plan superior sin pérdida de datos ni complicaciones técnicas.

Al momento de la compra, Hostinger permite elegir entre tres modalidades de pago: mensual, anual y bianual. Cuanto mayor sea el periodo contratado, mayor será el descuento aplicado al precio total, por lo que se recomienda considerar esta opción si planea usar la VPS a mediano o largo plazo.



## 2.2 Instalación del sistema operativo y acceso por consola

Durante el proceso de configuración inicial, deberá elegir la distribución de Linux que desea instalar en su VPS. Para este manual se utilizará Ubuntu 22.04 LTS.

Evite seleccionar versiones no LTS (Long Term Support), ya que las versiones LTS garantizan mayor estabilidad y soporte por parte de la comunidad y los desarrolladores oficiales durante varios años.

A continuación, deberá ingresar sus datos de facturación y seguir el proceso de compra hasta que el sistema le solicite establecer una contraseña para el usuario root.

El usuario root es el administrador del sistema en Linux, y su contraseña será requerida para acceder a la VPS vía consola (SSH). Asegúrese de utilizar una contraseña robusta y segura.

Una vez completado este proceso, Hostinger comenzará la instalación del sistema operativo. Esta operación suele tardar entre 2 y 5 minutos. Al finalizar, se le proporcionará una dirección IP pública, que usará para conectarse a su VPS de forma remota.

## 2.3 Acceso vía consola (SSH)

Desde su computadora personal, abra una consola o terminal (en Windows puede usar PowerShell o una terminal como Windows Terminal).

Ejecute el siguiente comando, reemplazando ip por la dirección IP pública que le asignó Hostinger:

```
ssh root@ip
```



El sistema le pedirá la contraseña que configuró anteriormente. Si todo fue realizado correctamente, accederá a su VPS de forma remota a través de la consola, y verá un entorno similar al siguiente:

```
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-60-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:        https://ubuntu.com/pro

System information as of Sun May 18 01:51:08 UTC 2025

System load:  0.0               Processes:            105
Usage of /:   6.7% of 47.39GB    Users logged in:     0
Memory usage: 20%              IPv4 address for eth0: 31.97.12.69
Swap usage:   0%               IPv6 address for eth0: 2a02:4780:2d:90aa::1

* Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
  just raised the bar for easy, resilient and secure K8s cluster deployment.

  https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Sun May 18 01:40:19 2025 from 169.254.0.1
root@srv830133:~#
```

*Nota:*

*SSH (Secure Shell) es un protocolo de red que permite establecer conexiones seguras y cifradas entre dos computadoras. En términos simples, le permite abrir una sesión de consola en un servidor remoto desde su propia computadora local, como si estuviera trabajando directamente en el sistema remoto.*

*A través de SSH podrá administrar el sistema operativo, instalar software y realizar cualquier configuración necesaria en su VPS.*

*Si bien Hostinger ofrece una terminal web integrada desde su panel de usuario para acceder a la VPS, no se recomienda su uso en entornos de trabajo colaborativo, ya que no permite una gestión adecuada de accesos. Para equipos de trabajo, lo más adecuado es crear usuarios individuales dentro de la VPS y que cada uno se conecte por SSH de forma segura.*

### 3. Configuración del sistema operativo

Una vez haya accedido a su VPS mediante SSH, lo primero que debe hacer es actualizar los paquetes del sistema operativo para asegurarse de contar con las versiones más recientes y seguras del software.

Ejecute los siguientes comandos:

```
apt update
```

Este comando actualiza la lista de paquetes disponibles desde los repositorios configurados.

```
apt upgrade -y
```

El comando anterior actualiza todos los paquetes instalados en el sistema. La opción `-y` responde automáticamente “yes” a todas las preguntas del proceso de actualización, lo que permite realizarlo sin intervención manual.

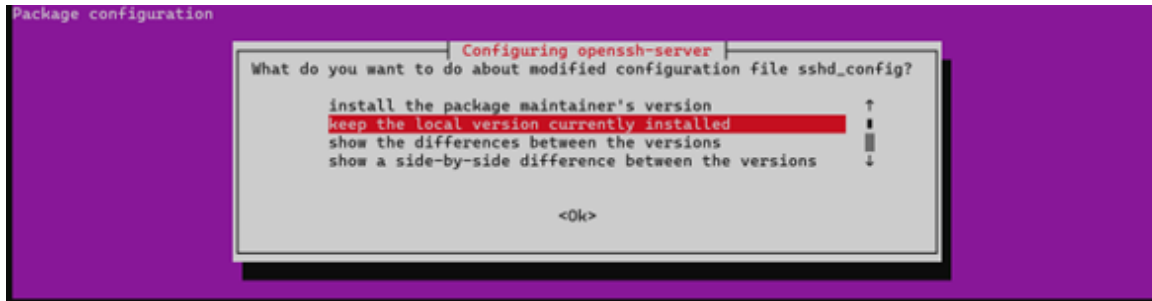
Si prefiere revisar los cambios antes de que se apliquen, puede omitir `-y` y ejecutar simplemente `apt upgrade`. En ese caso, el sistema le preguntará si desea continuar antes de aplicar las actualizaciones.

Importante:

Durante el proceso de actualización, especialmente al ejecutar `apt upgrade`, puede aparecer una pantalla que le pregunte si desea reemplazar el archivo de configuración de SSH (`sshd_config`).

Si esto ocurre, seleccione la opción que indica “Conservar la versión actual del archivo” (normalmente marcada como la predeterminada).

Esto garantiza que no se modifique la configuración actual del servicio SSH y evita el riesgo de perder el acceso remoto a la VPS.



## 4. Instalación de MySQL Server.

Un motor de base de datos relacional (RDBMS, por sus siglas en inglés) como MySQL o SQL Server es un componente fundamental en la arquitectura de la mayoría de las aplicaciones modernas. Estos sistemas permiten almacenar, organizar y acceder a grandes volúmenes de datos de manera estructurada, utilizando el lenguaje SQL (Structured Query Language) para realizar consultas, inserciones, actualizaciones y eliminación de datos.

### 4.1 Instalar MySQL Server

Ejecute el siguiente comando para instalar MySQL:

```
sudo apt install mysql-server -y
```

Esto descargará e instalará automáticamente la última versión estable del servidor MySQL.

#### 4.1.1 Asegurar la instalación

Una vez instalado, ejecute el siguiente comando para configurar y asegurar su instalación de MySQL:

```
sudo mysql_secure_installation
```

Durante este proceso, se le realizarán varias preguntas. A continuación se explica qué significan y qué opciones se recomienda seleccionar:

- **¿Desea configurar el plugin de validación de contraseñas?** Le ayudará a establecer políticas de seguridad para las contraseñas de los usuarios MySQL. Puede responder **No** si desea usar contraseñas simples. Si responde **Sí**, deberá escoger un nivel de validación:
  - o 0 Bajo
  - o 1 Medio

o 2 Alto

```
VALIDATE PASSWORD COMPONENT can be used to test passwords
and improve security. It checks the strength of password
and allows the users to set only those passwords which are
secure enough. Would you like to setup VALIDATE PASSWORD component?

Press y|Y for Yes, any other key for No: |
```

- “Remove anonymous users?” Se recomienda responder **y (Yes)** para eliminar usuarios anónimos y aumentar la seguridad.

```
By default, a MySQL installation has an anonymous user,
allowing anyone to log into MySQL without having to have
a user account created for them. This is intended only for
testing, and to make the installation go a bit smoother.
You should remove them before moving into a production
environment.

Remove anonymous users? (Press y|Y for Yes, any other key for No) : |
```

- “Disallow root login remotely?” Se recomienda responder **y (Yes)** para impedir el acceso remoto del usuario root. Posteriormente podrá crear usuarios con permisos específicos para conexiones remotas.

```
Normally, root should only be allowed to connect from
'localhost'. This ensures that someone cannot guess at
the root password from the network.

Disallow root login remotely? (Press y|Y for Yes, any other key for No) : |
```

- “Remove test database and access to it?” Se recomienda responder **y (Yes)**. La base de datos de prueba no es necesaria y puede representar un riesgo de seguridad.

```
Remove test database and access to it? (Press y|Y for Yes, any other key for No) : |
```

- “Reload privilege tables now?” Responda **y (Yes)** para que los cambios de configuración tengan efecto inmediato.

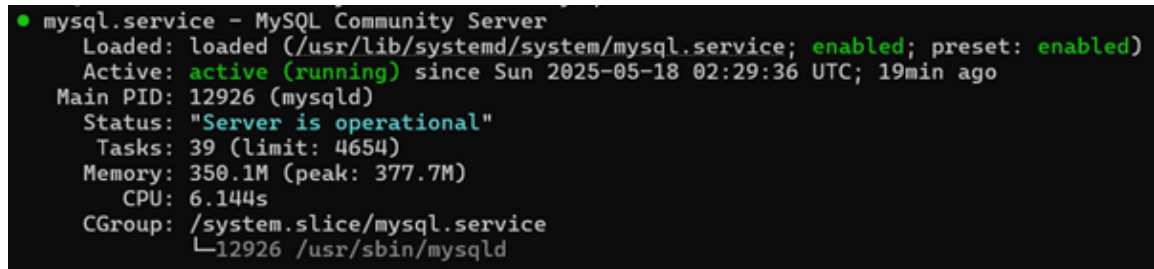
```
Reload privilege tables now? (Press y|Y for Yes, any other key for No) : |
```

#### 4.1.2 Verificar el estado del servicio

Para asegurarse de que el servidor MySQL se encuentra activo y funcionando, ejecute:

```
sudo systemctl status mysql
```

El estado debe mostrar active (running) como en la siguiente imagen:



```
● mysql.service - MySQL Community Server
   Loaded: loaded (/usr/lib/systemd/system/mysql.service; enabled; preset: enabled)
   Active: active (running) since Sun 2025-05-18 02:29:36 UTC; 19min ago
     Main PID: 12926 (mysqld)
    Status: "Server is operational"
      Tasks: 39 (limit: 4654)
     Memory: 350.1M (peak: 377.7M)
        CPU: 6.144s
    CGroup: /system.slice/mysql.service
            └─12926 /usr/sbin/mysqld
```

#### 4.2 Creación de usuarios MySQL

En muchos entornos de desarrollo y producción es necesario acceder a la base de datos de forma remota, ya sea mediante herramientas como MySQL Workbench o XAMPP. A continuación, se explica cómo crear un usuario MySQL con permisos de administración y habilitar el acceso remoto de forma segura.

- Acceder a la consola de MySQL

Ejecute el siguiente comando para entrar al cliente de MySQL como usuario root:

```
sudo mysql
```

- Crear el nuevo usuario con acceso remoto

Dentro de la consola de MySQL, ejecute lo siguiente (modifique adminuser y la contraseña por valores personalizados):

```
CREATE USER 'adminuser'@'%' IDENTIFIED BY 'TuClaveSegura123!';
```

Esto crea un usuario llamado adminuser con permisos para conectarse desde cualquier dirección IP (%).

- Otorgar privilegios administrativos

Otorgue permisos completos al usuario creado:

```
GRANT ALL PRIVILEGES ON . TO 'adminuser'@'%' WITH GRANT OPTION;
```

Esto le da acceso total a todas las bases de datos y permite delegar permisos a otros usuarios.

- Aplicar los cambios

```
FLUSH PRIVILEGES;  
EXIT;
```

- Permitir el puerto 3306 en el firewall (si usa UFW)  
MySQL usa el puerto 3306 por defecto. Si tiene activado el firewall UFW, ejecute:

```
sudo ufw allow 3306
```

- Permitir conexiones externas en MySQL

Por defecto, MySQL solo permite conexiones desde localhost (127.0.0.1). Para habilitar conexiones externas, edite el archivo de configuración:

```
sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf
```

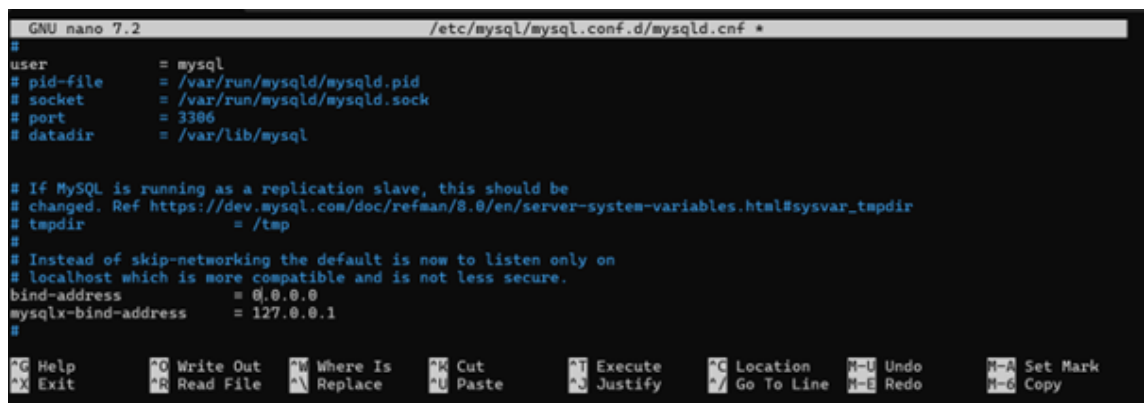
Busque la línea:

```
bind-address = 127.0.0.1
```

Y cámbiela por:

```
bind-address = 0.0.0.0
```

Esto indica que MySQL puede aceptar conexiones desde cualquier dirección IP.



```
GNU nano 7.2 /etc/mysql/mysql.conf.d/mysqld.cnf *  
#  
user                = mysql  
# pid-file           = /var/run/mysqld/mysqld.pid  
# socket             = /var/run/mysqld/mysqld.sock  
# port               = 3306  
# datadir            = /var/lib/mysql  
  
# If MySQL is running as a replication slave, this should be  
# changed. Ref https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_tmpdir  
# tmpdir             = /tmp  
#  
# Instead of skip-networking the default is now to listen only on  
# localhost which is more compatible and is not less secure.  
bind-address        = 0.0.0.0  
mysqlx-bind-address = 127.0.0.1  
#  
^G Help      ^O Write Out ^M Where Is  ^K Cut       ^J Execute   ^C Location  ^U Undo      ^_ Set Mark  
^X Exit      ^R Read File ^N Replace  ^P Paste     ^_ Justify   ^/ Go To Line ^E Redo      ^D Copy
```

Importante: asegúrese de no dejar accesos abiertos innecesarios en entornos de producción. Se recomienda limitar el acceso por IP y evitar el uso de usuarios root de forma remota.

Guarde los cambios presionando:

- Ctrl + O para guardar
  - Enter para confirmar
  - Ctrl + X para salir del editor
- 
- Reiniciar el servicio de MySQL

Finalmente, reinicie el servidor de MySQL para aplicar los cambios:

```
sudo systemctl restart mysql
```

### 4.3 ¿Cómo conectarse a MySQL desde XAMPP o MySQL Workbench?

Para conectarse a MySQL Server de forma remota utilizando una herramienta gráfica como XAMPP (con phpMyAdmin) o MySQL Workbench, debe completar los siguientes campos:

- Host: la dirección IP pública de su VPS
- Puerto: 3306 (puerto por defecto de MySQL)
- Usuario: el nombre del usuario MySQL con permisos remotos (por ejemplo, adminuser)
- Contraseña: la contraseña que definió para ese usuario

Una vez conectado, podrá crear bases de datos, usuarios, tablas y administrar MySQL Server de forma remota a través de una interfaz gráfica amigable.

### 4.4 Crear un usuario local para uso exclusivo de la Web API

Aunque se tenga acceso remoto, es buena práctica de seguridad crear un usuario con acceso restringido a localhost cuando la Web API .NET y MySQL Server están dentro de la misma VPS. De esta forma, el usuario solo puede conectarse desde el propio sistema, lo que reduce el riesgo de accesos no autorizados.

Esto aplica porque tanto MySQL como la API .NET estarán corriendo en el mismo servidor. Si la API se alojara en otro servidor o entorno, sería necesario crear un usuario con permiso de conexión desde la IP correspondiente.

#### 4.4.1 Crear base de datos y usuario local

Ejecute los siguientes comandos:

```
sudo mysql
```

Dentro del cliente de MySQL:

```
CREATE DATABASE dbbim;
```

```
CREATE USER 'userbim'@'localhost' IDENTIFIED BY 'ClaveFuerte123!';
```

```
GRANT ALL PRIVILEGES ON dbbim.* TO 'userbim'@'localhost';
```

```
FLUSH PRIVILEGES;
```

```
EXIT;
```

Esto crea:

- Una base de datos llamada dbbim
- Un usuario userbim que solo puede conectarse desde localhost
- Acceso completo de ese usuario a la base de datos dbbim

#### 4.4.2 Cadena de conexión (Connection String) en .NET

Para que su aplicación .NET se conecte correctamente a esta base de datos desde la misma VPS, utilice la siguiente cadena de conexión:

```
Server=localhost;Database=dbbim;User=userbim;Password=ClaveFuerte123!
```

Asegúrese de no exponer esta cadena de conexión en el código fuente público ni en repositorios sin protección.

## 5. Instalación de SQL Server

En esta sección se explica cómo instalar Microsoft SQL Server en una VPS con Ubuntu. Esto puede ser útil si tu Web API .NET está diseñada para trabajar con SQL Server en lugar de MySQL.

### 5.1 Importar la clave GPG y agregar el repositorio

Antes de instalar SQL Server, es necesario:

- Importar la clave GPG de Microsoft, que garantiza que los paquetes provienen de una fuente confiable y no han sido modificados.
- Agregar el repositorio oficial de Microsoft, ya que SQL Server no se encuentra disponible en los repositorios predeterminados de Ubuntu.



Este proceso asegura que el sistema pueda descargar, verificar e instalar correctamente SQL Server utilizando apt

Ejecute:

```
curl https://packages.microsoft.com/keys/microsoft.asc | sudo apt-key  
add -
```

```
curl  
https://packages.microsoft.com/config/ubuntu/22.04/mssql-server-2022.li  
st | sudo tee /etc/apt/sources.list.d/mssql-server.list
```

Reemplaza 22.04 por la versión de Ubuntu que esté usando.

Para verificar tu versión, ejecuta:

```
lsb_release -a
```

## 5.2 Instalar SQL Server

Ejecute:

```
sudo apt update
```

```
sudo apt install -y mssql-server
```

### 5.2.1 Configurar SQL Server

Una vez instalado, ejecuta el asistente de configuración:

```
sudo /opt/mssql/bin/mssql-conf setup
```

Durante este proceso:

- Acepta los términos de licencia
- Selecciona la edición.
  - o Para desarrollo y pruebas puedes usar Developer o Express.
  - o Para entornos profesionales se recomienda usar una de las ediciones comerciales.
- Configura una contraseña segura para el usuario sa

### 5.2.2 Verificar que el servicio esté corriendo

```
sudo systemctl status mssql-server
```

Debes ver active (running) en la salida.

Nota sobre compatibilidad:

En caso de encontrar un error, debe comprobar la compatibilidad del sistema operativo con las versiones de Sql Server.

A la fecha de este manual (junio 2025), Microsoft SQL Server aún no es compatible oficialmente con Ubuntu 24.04. Por este motivo, al intentar instalarlo desde los repositorios oficiales, el sistema mostrará errores como Not Found o fallos al iniciar el servicio (code=exited, status=127).

Para poder instalar SQL Server correctamente, se recomienda usar Ubuntu 22.04 LTS, la versión más reciente oficialmente soportada por Microsoft.

Si su VPS ya está en uso con Ubuntu 24.04 y no puede reinstalar, una alternativa válida es desplegar SQL Server dentro de un contenedor Docker, el cual es completamente funcional y no requiere repositorio nativo, dirijase a la sección 5.3 Alternativa: Ejecutar SQL Server con Docker en donde se explica este proceso.

### **5.2.3 (Opcional) Instalar herramientas de línea de comandos**

Aunque SQL Server ya puede ejecutarse como servicio en la VPS, la instalación de herramientas como sqlcmd y bcp brinda capacidades de administración directa desde la terminal, sin necesidad de interfaces gráficas o herramientas externas como SSMS o Azure Data Studio.

Estas herramientas permiten:

- Ejecutar consultas SQL directamente desde consola (ideal para pruebas rápidas o scripts automatizados).
- Administrar bases de datos, usuarios y permisos sin depender de conexión remota.
- Ejecutar scripts .sql o comandos automatizados en procesos CI/CD.
- Transferir datos en masa con bcp (Bulk Copy Program) entre archivos y bases de datos.

Además, son útiles en entornos donde no hay entorno gráfico (como una VPS en producción) o donde se desea ejecutar tareas administrativas desde scripts de mantenimiento o despliegue.

Ejecute:

```
sudo apt install -y curl gnupg2
```

```
curl https://packages.microsoft.com/keys/microsoft.asc | sudo apt-key  
add -
```

```
curl https://packages.microsoft.com/config/ubuntu/22.04/prod.list |  
sudo tee /etc/apt/sources.list.d/msprod.list
```

```
sudo apt update
```

```
sudo apt install -y mssql-tools unixodbc-dev
```

Agrega las herramientas al PATH:

```
echo 'export PATH="$PATH:/opt/mssql-tools/bin"' >> ~/.bashrc
```

```
source ~/.bashrc
```

#### 5.2.4 Probar conexión con sqlcmd

```
sqlcmd -S localhost -U sa -P 'TuContraseñaSegura123!'
```

Dentro del prompt puedes probar:

```
SELECT @@VERSION;
```

```
GO
```

#### 5.2.5 Permitir conexiones remotas (opcional)

Si deseas conectarte desde Azure Data Studio, SSMS o Visual Studio, abre el puerto predeterminado de SQL Server:

```
sudo ufw allow 1433
```

SQL Server utiliza el puerto TCP 1433 por defecto.

#### 5.2.6 Cadena de conexión en .NET

Usa esta cadena para conectarte desde tu Web API:

```
"Server=localhost;Database=TuBase;User  
Id=sa;Password=TuContraseñaSegura123!;"
```

### 5.3 Alternativa: Ejecutar SQL Server con Docker (recomendado para Ubuntu 24.04+)

En caso de que tu VPS use una versión de Ubuntu no soportada oficialmente por Microsoft (como Ubuntu 24.04), puedes optar por instalar SQL Server como contenedor Docker, lo cual es una solución funcional, ligera y portable.

#### 5.3.1 ¿Por qué usar Docker?

- Evita problemas de compatibilidad de sistema operativo
- Aísla SQL Server del resto de servicios
- Permite instalar, actualizar o reiniciar el servicio sin afectar otros procesos
- Requiere menos intervención del sistema operativo base
- Ideal para VPS en producción que ya tienen otros servicios configurados

#### 5.3.2 Instalar Docker

Si Docker no está instalado, ejecútalo en tu VPS:

```
sudo apt update
```

```
sudo apt install -y docker.io
```

Inicia el servicio y actívalo al arranque:

```
sudo systemctl start docker
```

```
sudo systemctl enable docker
```

#### 5.3.3 Ejecutar SQL Server en un contenedor

```
sudo docker run -e 'ACCEPT_EULA=Y'
```

```
-e 'MSSQL_SA_PASSWORD=TuClaveSegura123!'
```

```
-p 1433:1433 --name sqlserver
```

```
-d mcr.microsoft.com/mssql/server:2022-latest
```

Cambia TuClaveSegura123! por una contraseña segura que cumpla los requisitos de complejidad de SQL Server.

Este comando:

- Descarga la imagen oficial de SQL Server 2022
- Expone el puerto 1433 (el puerto estándar)
- Crea un contenedor llamado `sqlserver`
- Lo ejecuta en segundo plano (-d)

#### 5.3.4 Verificaciones y conexión

- Verificar que el contenedor está corriendo  
`sudo docker ps`  
Debes ver una línea con `mssql/server` en ejecución y el puerto 1433 mapeado.
- Conectarse a SQL Server desde la misma VPS  
Instala `mssql-tools` si aún no lo tienes (ver sección 5.2.3) y luego conéctate:  
`sqlcmd -S localhost -U sa -P 'TuClaveSegura123!'`
- Habilitar el puerto para acceso remoto (opcional)  
`sudo ufw allow 1433`
- Cadena de conexión para contenedor Docker  
"Server=localhost;Database=TuBase;User Id=sa;Password=TuClaveSegura123!;"

Si accedes desde fuera (por IP pública o nombre de dominio):

"Server=mi-servidor.com,1433;Database=TuBase;User  
Id=sa;Password=TuClaveSegura123!;"

Esta es una solución oficial, soportada y funcional, recomendada especialmente si ya tienes servicios críticos en tu VPS y no puedes reinstalar el sistema operativo.

## 6. Instalación de .Net

Para ejecutar una Web API desarrollada en .NET dentro de su VPS, es necesario instalar el runtime de ASP.NET Core, que permite ejecutar aplicaciones web sin necesidad del entorno completo de desarrollo.

A continuación se describen los pasos para instalar el runtime de .NET 8 en Ubuntu. En caso de utilizar otra versión, puede ajustarse según sea necesario.

### 6.1 Descargar e instalar los paquetes oficiales de Microsoft

Ejecute los siguientes comandos:

```
wget  
https://packages.microsoft.com/config/ubuntu/22.04/packages-microsoft-prod.deb -O packages-microsoft-prod.deb  
  
sudo dpkg -i packages-microsoft-prod.deb  
  
sudo apt update
```

## 6.2 Instalar el runtime de ASP.NET Core

Para proyectos desarrollados en .NET 8:

```
sudo apt install -y aspnetcore-runtime-8.0
```

Si su proyecto fue desarrollado con .NET 6, cambie el paquete por `aspnetcore-runtime-6.0`:

```
sudo apt install -y aspnetcore-runtime-6.0
```

## 6.3 Verificar la instalación

Para verificar que el runtime se instaló correctamente, ejecute:

```
dotnet --list-runtimes
```

Debe obtener un resultado similar a:

```
root@srv830133:~# dotnet --list-runtimes  
Microsoft.AspNetCore.App 8.0.16 [/usr/share/dotnet/shared/Microsoft.AspNetCore.App]  
Microsoft.NETCore.App 8.0.16 [/usr/share/dotnet/shared/Microsoft.NETCore.App]  
root@srv830133:~# |
```

Esto indica que el entorno está listo para ejecutar aplicaciones web desarrolladas con .NET.

# 7. Configuración de inicio de sesión sin contraseña (opcional, recomendado para CI/CD)

Por defecto, cada vez que se establece una conexión SSH con la VPS se debe ingresar manualmente la contraseña del usuario. Sin embargo, es posible automatizar el inicio de sesión utilizando claves SSH, lo cual es especialmente útil para flujos de integración y despliegue continuo (CI/CD) o simplemente para mejorar la seguridad y comodidad.

## 7.1 Generar clave SSH desde el equipo local

Desde su equipo (Windows, Linux o macOS), ejecute el siguiente comando en la terminal:

```
ssh-keygen -t ed25519 -C "mi-clave-ssh"
```

Reemplace "mi-clave-ssh" por un nombre descriptivo para identificar la clave.

Durante el proceso:

- Presione Enter en cada pregunta (archivo, passphrase y confirmación) para aceptar los valores por defecto.
- Esto generará dos archivos en la carpeta C:\Users\TuUsuario\.ssh\ (en Windows):
  - o id\_ed25519: clave privada (no compartir)
  - o id\_ed25519.pub: clave pública (esta se copiará a la VPS)

## 7.2 Copiar la clave pública a la VPS

Desde su equipo, ejecute el siguiente comando (reemplazando la IP):

```
scp C:\Users\TuUsuario.ssh\id_ed25519.pub root@:/root/
```

Ingrese la contraseña del VPS cuando sea solicitada. Esto copiará el archivo id\_ed25519.pub al directorio raíz de la VPS.

## 7.3 Configurar la clave pública en la VPS

Ahora, en la consola de la VPS, ejecute los siguientes comandos:

```
mkdir -p ~/.ssh
```

```
cat ~/id_ed25519.pub >> ~/.ssh/authorized_keys
```

```
chmod 600 ~/.ssh/authorized_keys
```

```
chmod 700 ~/.ssh
```

```
rm ~/id_ed25519.pub
```

Estos pasos:

- Crean el directorio ~/.ssh si no existe
- Añaden la clave pública al archivo authorized\_keys
- Establecen los permisos correctos para seguridad
- Elimina el archivo temporal .pub copiado

## 7.4 Verificar acceso sin contraseña

Desde su equipo, pruebe la conexión especificando la clave privada:

```
ssh -i C:\Users\TuUsuario.ssh\id_ed25519 root@ <IP_VPS>
```

Si la configuración fue exitosa, accederá a la VPS sin que se le solicite la contraseña.

A partir de este punto, podrá conectarse directamente con:

```
ssh root@<IP_VPS>
```

Este método es más seguro que el uso de contraseñas, siempre que la clave privada esté bien protegida.

Nota:

Explicación de ssh-keygen -t ed25519 -C "mi-clave-ssh"

Parte del comando	Significado
ssh-keygen	Es la herramienta estándar en sistemas Unix/Linux (y disponible en Windows) para <b>generar claves SSH</b> .
-t ed25519	Indica el <b>tipo de clave</b> que se va a generar. En este caso, se elige el algoritmo ed25519, que es moderno, seguro y más eficiente que el tradicional rsa.
-C "mi-clave-ssh "	Es un <b>comentario</b> que se añade a la clave pública para identificarla fácilmente. No afecta a la funcionalidad, pero es útil si maneja varias claves.

Hasta este punto, la VPS ya cuenta con el sistema operativo configurado, .NET Runtime instalado y MySQL Server en funcionamiento. A continuación, se explicarán dos formas de desplegar la Web API .NET dentro de la VPS.

- Opción 1: Despliegue manual (modo básico)



Este método consiste en publicar el proyecto .NET localmente en una carpeta, luego subirlo manualmente a la VPS y configurarlo para que se ejecute.

Cada vez que se realicen cambios en el código fuente, será necesario repetir el proceso completo de publicación y subida manual.

Esta opción es útil para proyectos personales, entornos de prueba o cuando se está comenzando.

Los pasos detallados se explicarán en la siguiente sección.

- **Opción 2: Despliegue automatizado con GitHub Actions (CI/CD)**  
Esta es la forma profesional y escalable de publicar una aplicación. Se configura un flujo de trabajo usando GitHub Actions, de modo que cada vez que se haga un Push al repositorio, el proyecto:

Se compila automáticamente

Se publica

Se transfiere e inicia en la VPS sin intervención manual

Esta es la opción recomendada para entornos de producción, trabajo colaborativo y despliegue continuo.

A continuación, se explicarán ambas metodologías paso a paso, comenzando por el despliegue manual.

## **8. Despliegue manual de la Web API .NET en la VPS**

En esta sección se explica cómo publicar, subir y ejecutar manualmente una Web API .NET en una VPS con Linux. Este método es útil para pruebas o proyectos personales, aunque no es óptimo para equipos ni despliegue continuo.

### **8.1 Publicar el proyecto localmente**

Desde su equipo local (Windows), abra una terminal o consola y publique su proyecto .NET en modo Release:

```
dotnet publish -c Release -o "C:\Ruta-Proyecto-Publicado\NameProject"
```

Esto generará los archivos de salida en la carpeta especificada. NameProject será el nombre de la carpeta a transferir.

## 8.2 Crear la carpeta del proyecto en la VPS

Conéctese a su VPS mediante SSH y ejecute:

```
sudo mkdir -p /var/www/api  
  
sudo chown -R root:root /var/www/api
```

Esto crea el directorio donde se almacenará el proyecto.

## 8.3 Subir los archivos a la VPS

Desde su equipo local, utilice scp para transferir la carpeta del proyecto a la VPS:

```
scp -r "C:\Ruta-Proyecto-Publicado\NameProject"  
root@<IP_VPS>:/var/www/api
```

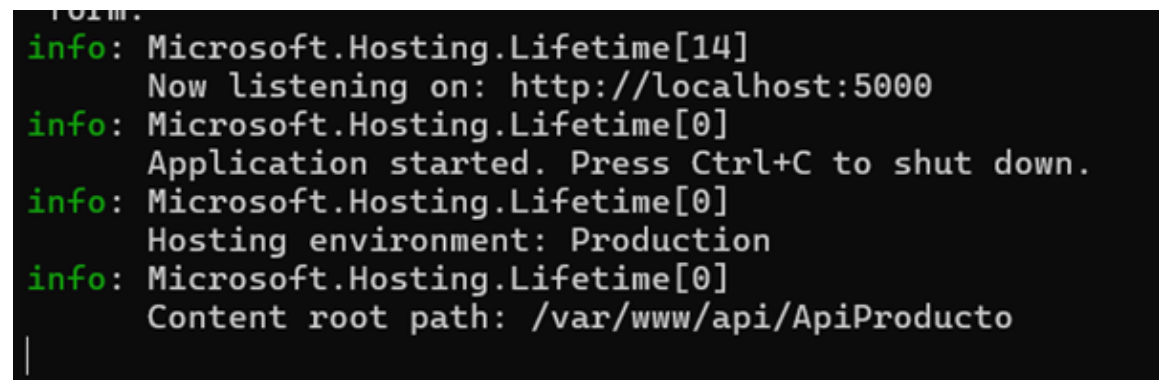
Asegúrese de reemplazar <IP\_VPS> por la dirección IP real de su VPS.

## 8.4 Ejecutar la API en la VPS

Ingresa a la VPS, diríjase a la carpeta del proyecto y ejecute:

```
cd /var/www/api/NameProject  
  
dotnet NameProject.dll
```

Si todo está correctamente configurado, la API comenzará a ejecutarse. El mensaje en consola debe mostrar que está escuchando en <http://localhost:5000>.



```
info: Microsoft.Hosting.Lifetime[14]  
Now listening on: http://localhost:5000  
info: Microsoft.Hosting.Lifetime[0]  
Application started. Press Ctrl+C to shut down.  
info: Microsoft.Hosting.Lifetime[0]  
Hosting environment: Production  
info: Microsoft.Hosting.Lifetime[0]  
Content root path: /var/www/api/ApiProducto  
|
```

## 8.5 Probar un endpoint desde la VPS

Si la API tiene un endpoint como /api/productos, puede probarlo directamente desde la consola usando curl:

```
curl http://localhost:5000/api/productos
```

## 8.6 Actualizar el proyecto

Cada vez que realice cambios en el código deberá repetir los siguientes pasos:

- **Detener el servicio actual:**

Buscar el proceso en ejecución:

```
ps aux | grep NameProject
```

```
root@srv830133:~# ps aux | grep ApiProducto
root      18101  0.1  3.1 273725164 125156 pts/1 Sl+  17:13   0:01 dotnet ApiProducto.dll
root      18326  0.0  0.0   7080   2176 pts/3  S+   17:31   0:00 grep --color=auto ApiProducto
root@srv830133:~#
```

Identifique la línea que contiene dotnet NameProject.dll y obtenga el **PID** (segunda columna).

Detenga el proceso con:

```
kill -9 <PID>
```

- **Volver a subir el proyecto actualizado:**

Desde su equipo local:

```
scp -r "C:\Ruta-Proyecto-Publicado\NameProject"
root@<IP_VPS>:/var/www/api
```

- **Reiniciar la API:**

Desde la VPS:

```
cd /var/www/api/NameProject
```

```
dotnet NameProject.dll
```

## 8.7 Limitaciones del despliegue manual

Aunque este método es funcional, presenta varias limitaciones:

- Se debe acceder manualmente a la VPS para subir y ejecutar el proyecto.
- No es escalable para equipos de trabajo, ya que habría que dar acceso SSH a cada integrante.
- No automatiza el proceso de despliegue tras realizar cambios.

## 8.8 Recomendación: Automatizar con CI/CD

Una alternativa profesional es utilizar GitHub Actions para implementar un flujo de CI/CD (Integración y Despliegue Continuo). Esto permite:

- Automatizar el despliegue al hacer push a la rama `main`.
- Mantener la VPS segura, sin acceso directo para el equipo.
- Visualizar el estado del despliegue desde GitHub.

En la siguiente sección se explicará cómo configurar GitHub Actions para desplegar automáticamente la Web API en la VPS.

## 9. Despliegue automatizado de Web API .NET en la VPS (CI/CD)

Para automatizar el despliegue de su Web API en la VPS, se utilizará **GitHub Actions** como solución de **CI/CD**. Este enfoque permite que, cada vez que se haga un push a la rama que escoja, el proyecto se publique, transfiera e inicie automáticamente en la VPS, sin necesidad de intervención manual.

Requisitos previos

Antes de comenzar, asegúrese de haber completado el paso:

- **Paso 7: Configuración de inicio de sesión sin contraseña**, en donde se generó la clave SSH para el acceso sin contraseña a la VPS.

### 9.1 Crear carpeta del proyecto en la VPS

En su VPS, cree la carpeta donde se almacenará el proyecto publicado:

```
sudo mkdir -p /var/www/api/NameProject
```

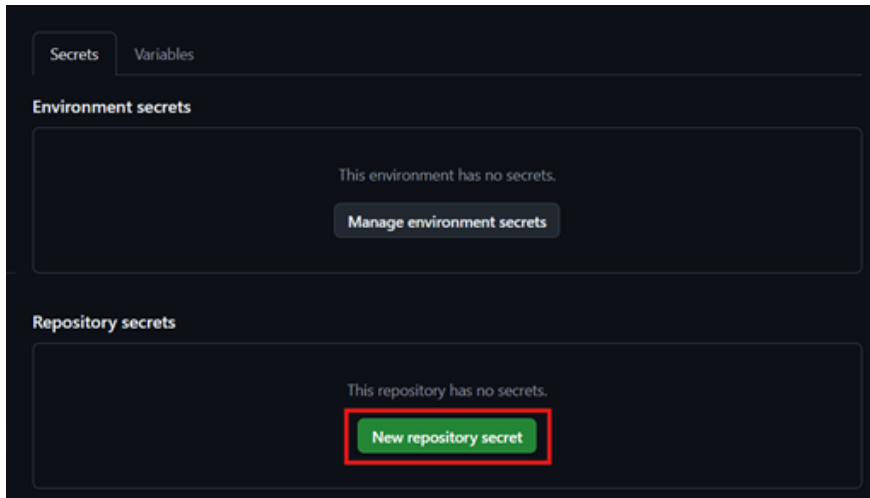
```
sudo chown -R root:root /var/www/api/NameProject
```

Reemplace `NameProject` por el nombre real de su Web API

### 9.2 Configurar los Repository Secrets en GitHub

Ingresa al repositorio del proyecto en GitHub y diríjase a:

Settings → Secrets and variables → Actions → Repository secrets



Actions secrets / New secret

Name \*

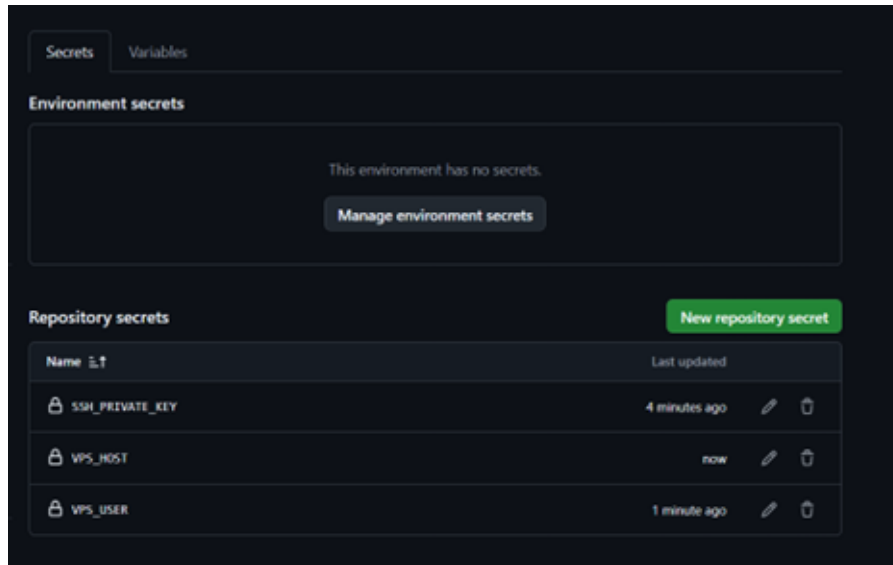
YOUR\_SECRET\_NAME

Secret \*

Add secret

Cree los siguientes secrets:

Name	Contenido
SSH_PRIVATE_KEY	El contenido completo del archivo id_ed25519 (clave privada, sin extensión) generado en el paso 5
VPS_USER	El nombre del usuario (por defecto: root)
VPS_HOST	La IP pública de su VPS



### 9.3 Crear el archivo deploy.yml

Dentro de su repositorio, cree un archivo en la ruta:

`.github/workflows/deploy.yml`

Agregue el siguiente contenido, adaptándolo según su proyecto:

```
name: Deploy .NET API to VPS
```

```
on: push: branches: [main] # Cambie si usa otra rama
```

```
jobs: deploy: runs-on: ubuntu-latest
```

```
steps:
```

```
- name: Checkout repository
  uses: actions/checkout@v3
```

```
- name: Setup .NET
  uses: actions/setup-dotnet@v3
  with:
    dotnet-version: '8.0.x' # Cambie por la versión que usa su
    proyecto
```

```
- name: Publish API
  run: dotnet publish NameProject/NameProject.csproj -c Release -o
```

publish

# Reemplace la ruta por la ruta real de su archivo .csproj en el repositorio

- name: Copy files to VPS via SSH  
uses: appleboy/scp-action@v0.1.7  
with:  
  host: \${ secrets.VPS\_HOST }  
  username: \${ secrets.VPS\_USER }  
  key: \${ secrets.SSH\_PRIVATE\_KEY }  
  source: publish/  
  target: /var/www/api/NameProject
- name: Restart service on VPS  
uses: appleboy/ssh-action@v1.0.0  
with:  
  host: \${ secrets.VPS\_HOST }  
  username: \${ secrets.VPS\_USER }  
  key: \${ secrets.SSH\_PRIVATE\_KEY }  
  script: |  
    systemctl restart nameproject.service  
    # Cambie 'nameproject' por el nombre real del servicio creado

En su primer push, es normal que el último paso falle, ya que el servicio aún no ha sido creado en la VPS. Esto se resolverá al configurar el servicio en la siguiente sección.

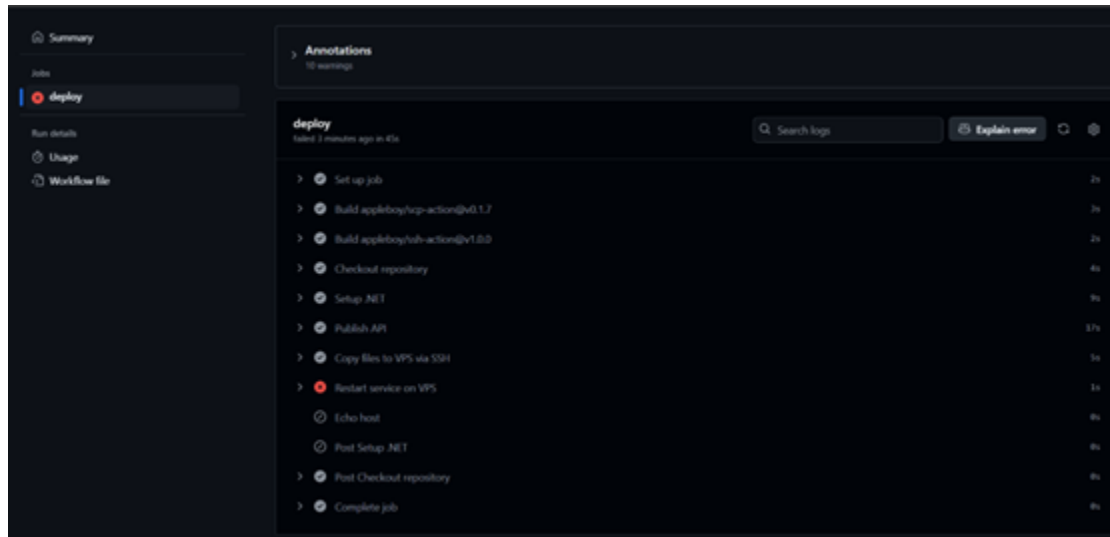
## 9.4 Probar el despliegue automático

Una vez configurado el archivo `deploy.yml`, realice un commit y push al repositorio. Esto activará el workflow.

Puede revisar el estado en:

**GitHub → Repositorio → Actions → [Nombre del commit] → Deploy**

Si el último paso (reinicio del servicio) falla, verá un error indicando que el servicio aún no existe, lo cual es esperado en este punto.



- Siguiendo paso

Ahora debe proceder con la creación del servicio en la VPS para que la Web API se ejecute como un demonio (servicio del sistema). Esto permitirá que la API:

Se ejecute en segundo plano

Se reinicie automáticamente si falla

Inicie automáticamente al encender la VPS

Diríjase a la sección 10: Configuración del servicio para inicio automático para completar el proceso.

## 10. Configuración del servicio para inicio automático (Systemd)

Para garantizar que su Web API .NET se ejecute en segundo plano, se reinicie automáticamente si falla y se inicie cada vez que se encienda la VPS, se configurará un servicio systemd personalizado.

### 10.1 Crear el archivo del servicio

En su VPS, ejecute:

```
sudo nano /etc/systemd/system/nameproject.service
```



Se abrirá un archivo vacío. Ingrese el siguiente contenido, adaptado a su proyecto:

```
[Unit] Description=API .NET de Productos After=network.target

[Service] WorkingDirectory=/var/www/api/NameProject/publish/
ExecStart=/usr/bin/dotnet
/var/www/api/NameProject/publish/NameProject.dll Restart=always
RestartSec=5 KillSignal=SIGINT SyslogIdentifier=nameproject User=root
Environment=ASPNETCORE_ENVIRONMENT=Production
Environment=DOTNET_PRINT_TELEMETRY_MESSAGE=false

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

¿Qué es ExecStart?

La directiva ExecStart indica qué comando se ejecutará para iniciar el servicio.

Se divide en dos partes:

`/usr/bin/dotnet`: ruta fija al ejecutable de .NET

`/var/www/api/NameProject/publish/NameProject.dll`: ruta al proyecto en la VPS

- Guarde y salga  
Presione Ctrl + O y luego Enter para guardar  
Presione Ctrl + X para salir del editor

## 10.2 Activar el servicio

Si tiene el proyecto corriendo desde el paso 6 de forma manual, deténgalo primero siguiendo la sección “Actualizar el proyecto” en ese mismo paso.

Ahora, ejecute los siguientes comandos:

```
sudo systemctl daemon-reexec
```

```
sudo systemctl daemon-reload
```

```
sudo systemctl enable nameproject.service
```

```
sudo systemctl start nameproject.service
```

## 10.3 Verificar que el servicio esté en ejecución

Use el siguiente comando:

```
sudo systemctl status nameproject.service
```

Debe ver un estado active (running) indicando que el servicio está funcionando correctamente.

```
root@srv830133:/# systemctl status apiproducto.service
● apiproducto.service - API .NET de Productos
   Loaded: loaded (/etc/systemd/system/apiproducto.service; enabled; preset: enabled)
   Active: active (running) since Sun 2025-05-18 20:17:02 UTC; 2min 56s ago
     Main PID: 21020 (dotnet)
        Tasks: 12 (limit: 4654)
       Memory: 26.3M (peak: 26.5M)
          CPU: 387ms
     CGroup: /system.slice/apiproducto.service
            └─21020 /usr/bin/dotnet /var/www/api/ApiProducto/publish/ApiProducto.dll

May 18 20:17:02 srv830133 apiproducto[21020]: info: Microsoft.AspNetCore.DataProtection.KeyManagement.XmlKeyManager[62]
May 18 20:17:02 srv830133 apiproducto[21020]: info: User profile is available. Using '/root/.aspnet/DataProtection-Keys'
May 18 20:17:02 srv830133 apiproducto[21020]: info: Microsoft.Hosting.Lifetime[10]
May 18 20:17:02 srv830133 apiproducto[21020]: info: Now listening on: http://localhost:5000
May 18 20:17:02 srv830133 apiproducto[21020]: info: Microsoft.Hosting.Lifetime[0]
May 18 20:17:02 srv830133 apiproducto[21020]: info: Application started. Press Ctrl+C to shut down.
May 18 20:17:02 srv830133 apiproducto[21020]: info: Microsoft.Hosting.Lifetime[0]
May 18 20:17:02 srv830133 apiproducto[21020]: info: Hosting environment: Production
May 18 20:17:02 srv830133 apiproducto[21020]: info: Microsoft.Hosting.Lifetime[0]
May 18 20:17:02 srv830133 apiproducto[21020]: info: Content root path: /var/www/api
lines 1-20/20 (END)
```

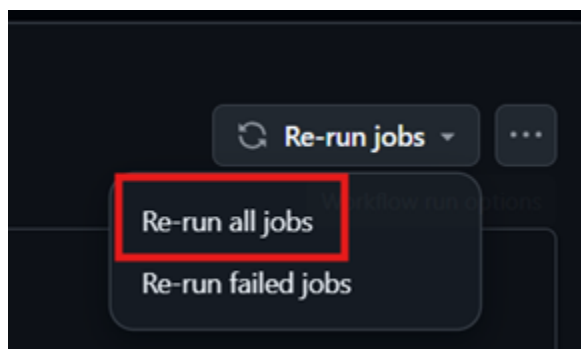
Por defecto, si no configuró explícitamente un puerto en Program.cs, su API escuchará en el puerto 5000.

## 10.4 Finalizar el Deploy si está usando GitHub Actions

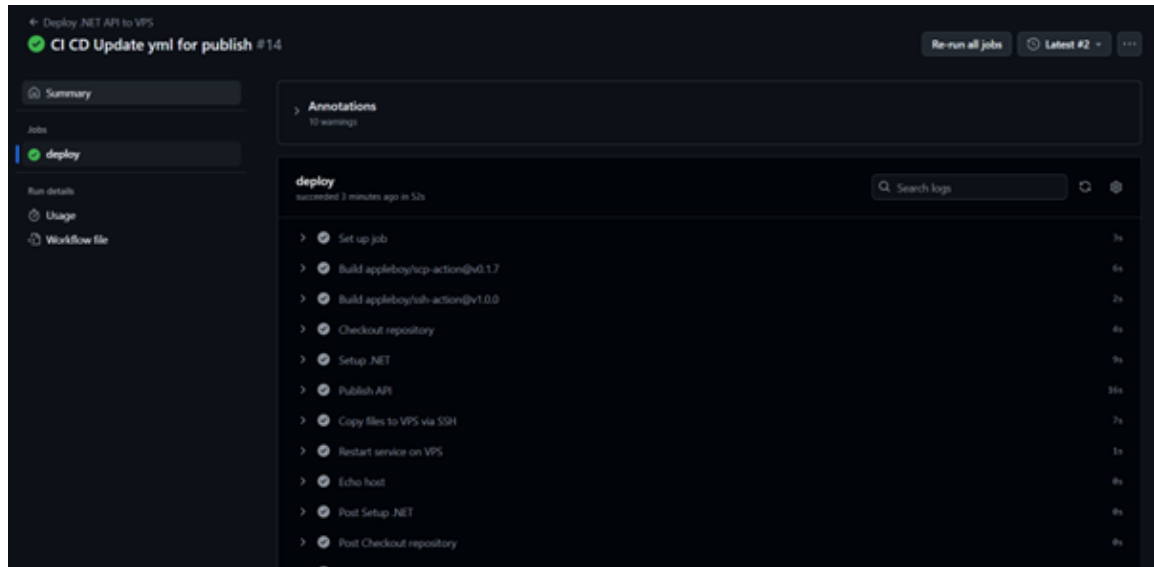
Si configuró CI/CD (paso 7), ahora que el servicio ya existe, puede volver a ejecutar el workflow para completar el proceso automático de despliegue.

Diríjase a:

GitHub → Repositorio → Actions → Último workflow → Deploy → Re-run jobs → Re-run all jobs



Al finalizar, la API se actualizará y reiniciará automáticamente gracias al servicio configurado.



## 10.5 Verificar el endpoint desde la VPS

Pruebe un endpoint de su API, por ejemplo:

```
curl http://localhost:5000/api/productos
```

Si la respuesta es correcta, su API está funcionando como servicio del sistema correctamente.

## 11. Configuración de NGINX como Proxy para solicitudes externas

Hasta este punto, su Web API .NET se ha ejecutado y probado de forma interna dentro de la VPS. Sin embargo, para permitir que clientes externos (como Postman, navegadores u otras aplicaciones) accedan a la API, es necesario exponerla mediante un proxy inverso.

Para esto se utilizará NGINX, un servidor HTTP ligero y potente que redirige las solicitudes externas a la Web API que corre de manera local en la VPS.

### 11.1 Instalar NGINX

Ejecute los siguientes comandos en su VPS:

```
sudo apt update
```

```
sudo apt install nginx -y
```

## 11.2 Crear el archivo de configuración para su proyecto

Cree un archivo de configuración para su Web API:

```
sudo nano /etc/nginx/sites-available/nameproject
```

Ingresa el siguiente contenido (ajuste el puerto si su API no usa el 5000):

```
server { listen 80; server_name _;

location / {
    proxy_pass          http://localhost:5000;
    proxy_http_version 1.1;
    proxy_set_header    Upgrade $http_upgrade;
    proxy_set_header    Connection keep-alive;
    proxy_set_header    Host $host;
    proxy_cache_bypass  $http_upgrade;
    proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header    X-Forwarded-Proto $scheme;
}

}
```

Reemplace el puerto 5000 por el puerto real en el que su API esté corriendo.

Este archivo permitirá a NGINX redirigir solicitudes externas entrantes al servidor local donde corre la Web API.

## 11.3 Activar el sitio y reiniciar NGINX

A continuación, active la configuración del sitio y reinicie NGINX:

```
sudo rm /etc/nginx/sites-enabled/default
```

```
sudo ln -s /etc/nginx/sites-available/nameproject
/etc/nginx/sites-enabled/
```

```
sudo nginx -t # Verificar que no haya errores de sintaxis
```

```
sudo systemctl restart nginx
```

## 11.4 Probar el acceso desde el exterior

Ahora puede realizar solicitudes externas a la API desde Postman, curl u otro cliente HTTP:

GET `http://<IP_VPS>/api/endpoint`

Reemplace <IP\_VPS> por la IP pública de su servidor y /api/endpoint por la ruta real de su Web API.

## 11.4 Sobre el uso de HTTPS

Con esta configuración, su Web API solo acepta solicitudes HTTP. Para habilitar HTTPS, necesita tener un dominio vinculado a su VPS. Esto se explicará en la siguiente sección del manual.

Nota técnica:

NGINX actúa como puerta de enlace entre internet y la Web API, por lo tanto:

- Solo NGINX necesita el certificado HTTPS, ya que es el que se comunica con el exterior.
- La Web API .NET no necesita escuchar en HTTPS, ya que su comunicación con NGINX es completamente local dentro del sistema.
- De hecho, configurar la Web API para escuchar en HTTPS (por ejemplo con `app.Urls.Add("https://0.0.0.0:7136")`) generará error en la VPS, ya que no existe un certificado HTTPS interno disponible.

## 11.5 ¿Y si tengo múltiples Web APIs?

Si planea desplegar más de una Web API en la misma VPS, deberá asegurarse de que cada API escuche en un puerto diferente. Para esto, modifique su Program.cs y agregue una línea como la siguiente:

```
app.Urls.Add("http://0.0.0.0:5136");
```

El puerto 5000 es el predeterminado, pero Linux no permite que múltiples servicios escuchen en el mismo puerto, por lo que debe asignar uno distinto para cada API.

## 12. Conexión entre VPS y Dominio

Antes de asociar un dominio a su VPS, es importante comprender la diferencia entre los conceptos clave: Dominio, Hosting y VPS.

## **12.1 Diferencias entre Dominio, Hosting, y VPS.**

### **12.1.1 ¿Qué es un Dominio?**

Un dominio es la dirección web que se utiliza para acceder a un sitio en internet, por ejemplo: [www.midominio.com](http://www.midominio.com).

Está compuesto por:

- Un nombre (por ejemplo: midominio)
- Una extensión o TLD (Top-Level Domain), como .com, .net, .org, .edu, .io, entre muchas otras.

Hasta este punto del manual se ha trabajado directamente con la dirección IP pública de la VPS. Sin embargo, para acceder a servicios web de forma amigable, profesional y recordable, es necesario utilizar un nombre de dominio que apunte a esa IP.

### **12.1.2 ¿Qué es un Hosting?**

El hosting es un servicio que proporciona espacio de almacenamiento para sitios web. En él se alojan los archivos HTML, imágenes, bases de datos y código de la página web.

Un hosting compartido no es un computador completo, sino un espacio limitado dentro de un servidor compartido con otros usuarios.

Sus recursos como CPU, RAM y disco son repartidos entre múltiples sitios, lo que limita su capacidad de personalización.

Además:

- No permite instalar software personalizado o servicios de backend como una API .NET.
- Generalmente está enfocado a sitios en PHP o WordPress.

### **12.1.3 ¿Qué es una VPS?**

Una VPS (Virtual Private Server) es un servidor virtual con un sistema operativo completo, como Ubuntu o Windows Server.

Funciona como un computador conectado a internet, sobre el cual se puede:




- Instalar cualquier software (como .NET, MySQL, NGINX, etc.)
- Configurar servicios personalizados
- Usarlo como hosting, pero con control total




En este contexto, la VPS actúa tanto como servidor de backend como alojamiento del sitio web.

#### 12.1.4 Costos y consideraciones

- Los dominios se compran de forma anual.  
El precio varía según el nombre y la extensión.  
Por ejemplo, un dominio que incluya palabras populares como jobs, cloud, tech puede ser más costoso.
- Un hosting compartido suele ser más económico que una VPS, pero ofrece menos control y capacidad de ejecución de servicios personalizados como .NET.
- Una VPS tiene un costo mayor, pero permite el despliegue profesional de aplicaciones, especialmente aquellas que requieren control total del entorno.

A continuación se explica a detalle todas las diferencias y características a detalle:

Característica	 Dominio	 Hosting (compartido)	 VPS (Servidor Privado Virtual)
¿Qué es?	Nombre o dirección del sitio web	Espacio en un servidor para alojar tu sitio	Servidor virtual con recursos dedicados
Función principal	Redirigir usuarios al sitio (resuelve a una IP)	Almacenar los archivos del sitio web	Control completo del entorno del servidor
Accesible por humanos	✓ Sí (nombre amigable)	✗ No directamente	✗ No directamente
Requiere conocimientos técnicos	✗ No	✗ No (en la mayoría de casos)	✓ Sí (manejo de sistema operativo, terminal)
Recursos del sistema	No aplica	Compartidos con otros usuarios	Dedicados (RAM, CPU, almacenamiento virtual)
Acceso root / administrador	✗ No	✗ No	✓ Sí

Personalización del entorno	 No	 Muy limitada	 Total
Seguridad	No aplica	Básica (depende del proveedor)	Alta (más aislamiento y control)
Ideal para	Darle nombre a un sitio	Sitios pequeños o medianos	Sitios con alto tráfico, apps personalizadas
Ejemplo	<a href="http://www.miempresa.com">www.miempresa.com</a>	Donde vives los archivos del sitio	Tu propio mini-servidor virtual
Costo aproximado	Bajo (pago anual)	Bajo (pago mensual/anual)	Medio a alto (según recursos contratados)

En la siguiente sección se explicará cómo conectar un dominio adquirido con la IP pública de la VPS, para permitir solicitudes externas a su API o sitio web de forma profesional, y cómo asegurar la conexión mediante HTTPS con Certbot.

## 12.2 Conexión de dominio a la VPS y configuración HTTPS con Certbot

En esta sección se explicará cómo asociar un dominio adquirido (en este caso desde Hostinger) a la dirección IP pública de su VPS, configurar NGINX para recibir solicitudes externas desde ese dominio, y asegurar la comunicación mediante HTTPS con Certbot.

### 12.2.1 ¿Por qué se necesita un registro DNS?

Los registros DNS permiten que cuando un usuario escribe un dominio (como <https://api.midominio.com>), el navegador sepa a qué dirección IP conectarse (por ejemplo, la IP pública de su VPS).

No es una redirección. El DNS actúa como un traductor de nombres a direcciones IP, y el usuario solo ve el dominio, nunca la IP.

### 12.2.2 Crear registro DNS en Hostinger

Ingresa al panel de administración DNS de su dominio en Hostinger y agregue un nuevo registro.

El formulario se verá similar a este:



**Administrar registros DNS**

Estos registros definen cómo se comporta tu dominio. Los usos comunes incluyen apuntar tu dominio a servidores web o configurar la entrega de email para tu dominio.

Type: A | Nombre: prueba | Apunta a: 0.0.0.0 | TTL: 300 | **Agregar registro**

Explicación de las casillas:

- Tipo: Esto define el tipo de registro DNS. El más común es el tipo A, pero hay otros también.

Tipo	Significado	Ejemplo de uso
A	Apunta un nombre de dominio a una dirección IP IPv4.	api.midominio.com → 31.97.12.69
AAAA	Igual que A, pero para direcciones IPv6.	No es necesario si se usa IPv4
CNAME	Alias de otro nombre de dominio.	<u>www.midominio.com</u> → midominio.com
MX	Dirección del servidor de correo.	(No se usa para API o web)
TXT	Texto arbitrario (verificaciones, SPF, etc).	(No aplica para apuntar al VPS)

- Host / Nombre: Este campo indica qué parte del dominio se está configurando. Se suele llamar "Host" o "Nombre".

Nombre	Resultado final	Qué significa
@	<u>davo.com</u>	El dominio raíz (principal)
api	<u>api.davo.com</u>	Subdominio usado típicamente para APIs
www	<u>www.davo.com</u>	Subdominio común para sitios web
backend	<u>backend.davo.com</u>	Otro subdominio personalizado

prueba	<u>prueba.davo.com</u>	Subdominio temporal o de pruebas
--------	------------------------	----------------------------------

Lo que escriba en este campo se convierte en un subdominio.

- Apunta: Aquí debe ingresar la IP pública de su VPS, que es donde está alojada la Web API .NET, Ejemplo: 54.54.54.54
- TTL (Time To Live) Define cuántos segundos deben los servidores DNS recordar (cachear) el valor antes de consultarlo nuevamente.  
Para pruebas: 300 (5 minutos)  
Para producción: 3600 (1 hora) o 14400 (4 horas)
- ¿Cómo funciona todo junto?  
Cuando un usuario escribe en el navegador (o se haga una solicitud HTTPS desde PostMan o un FrontEnd): <https://api.midominio.com>  
Ocurre lo siguiente:
  - o El navegador pregunta: ¿Cuál es la IP de `api.midominio.com`?
  - o El servidor DNS (Hostinger) revisa sus registros.
  - o Encuentra un registro tipo A con nombre `api` que apunta a `54.54.54.54` (su VPS).
  - o El navegador se conecta a esa IP.
  - o NGINX en su VPS recibe la solicitud y la redirige al puerto donde está corriendo la Web API .NET.
- Configuración recomendada
  - o Tipo: A
  - o Nombre: `api` (*puede usar cualquier nombre identificable*)
  - o Apunta a: IP pública de su VPS
  - o TTL: **300** para pruebas, **3600** o **14400** para producción

Una vez completado, haga clic en Agregar registro.

### 12.2.3 ¿Y si tengo múltiples Web APIs?

Si desea desplegar más de una Web API .NET en la misma VPS, debe tener en cuenta lo siguiente:

- Crear un registro DNS para cada Web API

Por cada API, debe agregar un nuevo registro tipo A en el panel DNS de su dominio. Lo único que cambia entre ellos es el campo “Nombre” (host), que se traduce en subdominios diferentes.

Ejemplos:

API	Subdominio sugerido	Resultado final
API de productos	apiproduktos	apiproduktos.midominio.com
API de usuarios	apiusuarios	apiusuarios.midominio.com
API de pedidos	apipedidos	apipedidos.midominio.com

Aunque podría nombrarlas `api1`, `api2`, etc., se recomienda usar nombres descriptivos para facilitar la identificación a futuro.

- Configurar un archivo NGINX por cada Web API

Por cada Web API desplegada, debe crear un archivo de configuración individual en NGINX, ubicado en:

```
/etc/nginx/sites-available/nameproject
```

Use el siguiente comando para crear el archivo:

```
sudo nano /etc/nginx/sites-available/apinameproject
```

Y escriba en él algo como esto (modifique el nombre del subdominio y el puerto):

```
server { listen 80; server_name apinameproject.midominio.com;

location / {
    proxy_pass          http://localhost:5100;
    proxy_http_version 1.1;
    proxy_set_header    Upgrade $http_upgrade;
    proxy_set_header    Connection keep-alive;
    proxy_set_header    Host $host;
    proxy_cache_bypass  $http_upgrade;
    proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header    X-Forwarded-Proto $scheme;
}

}
```

Cambie:

- o `apinameproject.midominio.com` por el subdominio real.
- o `5100` por el puerto donde esté corriendo esa Web API específica.
- Asegúrese de que cada Web API use un puerto distinto  
Como ya se explicó anteriormente, debe asignar un puerto único por API en su archivo `Program.cs` de .NET, por ejemplo:

```
app.Urls.Add("http://0.0.0.0:5100");
```

Si dos APIs usan el mismo puerto, una de ellas no podrá ejecutarse debido a conflicto de puerto.

- Habilitar cada archivo en NGINX  
Luego de crear cada archivo en `sites-available`, actívelo con un enlace simbólico:

```
sudo ln -s /etc/nginx/sites-available/apinameproject  
/etc/nginx/sites-enabled/
```

```
sudo nginx -t    # Verifica que la configuración esté correcta  
sudo systemctl reload nginx
```

Con esta estructura, podrá tener múltiples APIs independientes, cada una con su propio subdominio y configuración.

## 12.3 Configurar NGINX para responder a solicitudes desde el dominio + Certificado HTTPS

Una vez tengas tu registro DNS correctamente configurado apuntando a la IP de tu VPS, el siguiente paso es configurar NGINX en la VPS para que escuche solicitudes dirigidas al dominio (por ejemplo, `api.midominio.com`) y las redirija internamente hacia tu Web API.

### 12.3.1 Editar el archivo de configuración de NGINX

Si ya habías creado el archivo de configuración del proyecto, ábrelo nuevamente con:

```
sudo nano /etc/nginx/sites-available/nameproject
```

Modifica el parámetro `server_name` para que coincida con el dominio configurado (por ejemplo, `api.midominio.com`). El contenido del archivo debería quedar así:

```
server { listen 80; server_name api.midominio.com;
```

```
location / {
    proxy_pass          http://localhost:5000; # Cambie al puerto donde
corre su API
    proxy_http_version 1.1;
    proxy_set_header    Upgrade $http_upgrade;
    proxy_set_header    Connection keep-alive;
    proxy_set_header    Host $host;
    proxy_cache_bypass  $http_upgrade;
    proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header    X-Forwarded-Proto $scheme;
}

}
```

Asegúrate de usar el puerto correcto si tu API no corre en 5000.

Guarde con Ctrl + O, luego presione Enter, y cierre con Ctrl + X.

### **12.3.2 Activar la configuración en NGINX**

Cree el enlace simbólico en sites-enabled para habilitar la configuración:

```
sudo ln -s /etc/nginx/sites-available/nameproject
/etc/nginx/sites-enabled/
```

Verifique que no haya errores de sintaxis:

```
sudo nginx -t
```

Recargue NGINX:

```
sudo systemctl reload nginx
```

A partir de este momento, las solicitudes al dominio api.midominio.com se redirigirán al backend que corre en localhost:5000

Si cometiste un error en el nombre del archivo o necesitas eliminarlo:

```
sudo rm /etc/nginx/sites-enabled/NombreIncorrecto
```

Luego corrige y vuelve a ejecutar nginx -t y systemctl reload nginx.

### **12.3.3 Habilitar HTTPS con Certbot (Let's Encrypt)**

Para asegurar las solicitudes y habilitar el protocolo HTTPS, instala Certbot:

```
sudo apt update
```

```
sudo apt install certbot python3-certbot-nginx -y
```

Ejecuta Certbot para emitir y configurar el certificado:

```
sudo certbot --nginx -d api.midominio.com
```

Te pedirá:

- Un correo electrónico
- Confirmar condiciones de uso
- Si deseas redirigir automáticamente HTTP a HTTPS (recomendado: **Sí**)

#### **12.3.4 Verificar renovación automática**

Certbot intenta renovar los certificados antes de que expiren. Puedes probar la renovación con:

```
sudo certbot renew --dry-run
```

Una vez completado el proceso, ya puedes probar tu Web API de forma segura:

Desde navegador: <https://api.midominio.com/api/productos>

Desde Postman: GET <https://api.midominio.com/api/endpoint>

## **13. Comandos más usados.**

A continuación, se resumen los comandos clave utilizados durante la configuración de la VPS para ejecutar una Web API .NET. Esta sección sirve como referencia rápida para tareas frecuentes de despliegue, configuración y mantenimiento.

### **13.1 Creación de un servicio systemd para Web API .NET**

```
sudo nano /etc/systemd/system/nameproject.service
```

Contenido del archivo (modificar rutas y nombre del proyecto):

```
[Unit] Description=API .NET de Productos After=network.target
```

```
[Service] WorkingDirectory=/var/www/api/NameProject/publish/
```

```
ExecStart=/usr/bin/dotnet
```

```
/var/www/api/NameProject/publish/NameProject.dll Restart=always
```

```
RestartSec=5 KillSignal=SIGINT SyslogIdentifier=nameproject User=root
```

```
Environment=ASPNETCORE_ENVIRONMENT=Production
Environment=DOTNET_PRINT_TELEMETRY_MESSAGE=false
```

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

Activar el servicio:

```
sudo systemctl daemon-reexec

sudo systemctl daemon-reload

sudo systemctl enable nameproject.service
```

## 13.2 NGINX como Proxy Inverso

Crear archivo de configuración para el dominio:

```
sudo nano /etc/nginx/sites-available/nameproject
```

Contenido del archivo:

```
server { listen 80; server_name api.midominio.com;

    location / {
        proxy_pass          http://localhost:5000;
        proxy_http_version  1.1;
        proxy_set_header    Upgrade $http_upgrade;
        proxy_set_header    Connection keep-alive;
        proxy_set_header    Host $host;
        proxy_cache_bypass  $http_upgrade;
        proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header    X-Forwarded-Proto $scheme;
    }

}
```

## 13.3 Activar y recargar configuración NGINX

```
sudo ln -s /etc/nginx/sites-available/nameproject
/etc/nginx/sites-enabled/

sudo nginx -t
```

```
sudo systemctl reload nginx
```

### 13.4 Eliminar sitio NGINX

```
sudo rm /etc/nginx/sites-enabled/NameToDelete
```

### 13.5 Generación de certificados HTTPS con Certbot

```
sudo apt update
```

```
sudo apt install certbot python3-certbot-nginx -y
```

```
sudo certbot --nginx -d api.midominio.com
```

### 13.6 Subir carpeta publicada desde el repositorio

Desde su máquina local (Windows):

```
scp -r C:/Ruta/Del/Proyecto/* root@IP_VPS:/var/www/api
```

Reemplace C:/Ruta/Del/Proyecto/ por la ruta local al proyecto publicado

Reemplace root@IP\_VPS por sus credenciales e IP real

## 14. Conclusión

Con esto ha finalizado la **configuración completa de su VPS para desplegar y ejecutar una Web API .NET de forma profesional**.

Este manual ha cubierto desde la adquisición e instalación del sistema operativo hasta la exposición pública y segura de su aplicación mediante **NGINX, dominio personalizado y certificado HTTPS**.

### 14.1. ¿Qué logró al finalizar este proceso?

- Instalar y asegurar un sistema Ubuntu en una VPS
- Instalar y configurar .NET y MySQL Server
- Ejecutar una Web API .NET como servicio en segundo plano
- Implementar despliegue manual y automatizado (CI/CD con GitHub Actions)
- Exponer la API mediante NGINX con nombre de dominio propio
- Asegurar la comunicación mediante HTTPS con Certbot



Si encontró algún error, tiene sugerencias o desea resolver dudas adicionales, **no dude en ponerse en contacto conmigo** por medio de mi correo electrónico: [joebaena@gmail.com](mailto:joebaena@gmail.com), también puede dirigirse a mi sitio web [www.joelflow.com](http://www.joelflow.com) y ver más proyectos

Gracias por seguir este manual, y ¡mucho éxito desarrollando y desplegando sus proyectos .NET!

## 15. Referencias

Para las referencias hice el trabajo de reunir sitios oficiales, artículos científicos y videos que he visto acerca del tema, fue un trabajo extenso reunir todo realmente, separe la documentación en dos: Para los pragmáticos tanto documentación oficial, como de páginas web y artículos científicos, y para los que les gusta más los videos, les comparto los videos de YouTube, igual en algún punto tendrán que ir a la documentación oficial, no se acostumbren mucho a los videos.

### 15.1. Documentación y artículos técnicos

- Host ASP.NET Core on Linux with Nginx. Microsoft.  
<https://learn.microsoft.com/en-us/aspnet/core/host-and-deploy/linux-nginx?view=aspnetcore-9.0&tabs=linux-ubuntu>
- Deploying with GitHub Actions. GitHub.  
<https://docs.github.com/en/actions/use-cases-and-examples/deploying/deploying-with-github-actions>
- Using secrets in GitHub Actions. GitHub.  
<https://docs.github.com/es/actions/security-for-github-actions/security-guides/using-secrets-in-github-actions>
- Securing CI/CD with secrets and variables. GitHub.  
<https://resources.github.com/learn/pathways/automation/advanced/securing-ci-cd-pipelines-with-secrets-and-variables/>
- Install and configure a MySQL server - Ubuntu. Ubuntu.  
<https://documentation.ubuntu.com/server/how-to/databases/install-mysql/index.html>
- Installing MySQL on Linux. MySQL  
<https://dev.mysql.com/doc/refman/8.4/en/linux-installation.html>
- How to point a domain to Hostinger. Hostinger.  
<https://support.hostinger.com/en/articles/1863967-how-to-point-a-domain-to-hostinger>
- Quickstart: Install SQL Server and create a database on Ubuntu. Microsoft.  
<https://learn.microsoft.com/en-us/sql/linux/quickstart-install-connect-ubuntu?view=sql-server-ver17&tabs=ubuntu2004>
- Generating a new SSH key and adding it to the ssh-agent. GitHub.

<https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent>

- Inicio rápido: Ejecución de imágenes de contenedor de SQL Server para Linux con Docker. Microsoft  
<https://learn.microsoft.com/es-es/sql/linux/quickstart-install-connect-docker?view=sql-server-ver17&tabs=cli&pivots=cs1-bash>
- Microsoft SQL Server - Ubuntu based images. DockerHub.  
<https://hub.docker.com/r/microsoft/mssql-server>
- How to Install MySQL on Ubuntu – Step-by-Step Guide. DigitalOcean.  
<https://www.digitalocean.com/community/tutorials/how-to-install-mysql-on-ubuntu-20-04>
- How to Create an SSH Key in Linux: Easy Step-by-Step Guide. DigitalOcean.  
<https://www.digitalocean.com/community/tutorials/how-to-configure-ssh-key-based-authentication-on-a-linux-server>
- How To Configure Nginx as a Reverse Proxy on Ubuntu 22.04. DigitalOcean.  
<https://www.digitalocean.com/community/tutorials/how-to-configure-nginx-as-a-reverse-proxy-on-ubuntu-22-04>
- NGINX. Darin.web.id.  
<https://darin.web.id/linux/nginx-basic/NGINX A Practical Guide Preview Edition.pdf>

## 15.2 Videos recomendados

- SSL, TLS, HTTPS Explained – ByteByteGo.  
<https://www.youtube.com/watch?v=j9QmMEWmcfo&t>
- ¿Qué es HTTPS y cómo funciona? – Soy Dalto.  
<https://www.youtube.com/watch?v=JKyVWiD0U6o>
- CI/CD with GitHub Actions [8 of 8] | .NET on Azure for Beginners – Dotnet.  
<https://www.youtube.com/watch?v=7LkRipTlTzc>
- CI/CD Tutorial using GitHub Actions - Automated Testing & Deployments – Tom Shaw.  
<https://www.youtube.com/watch?v=YLtlz88zrLg>
- Configure NGINX as a Reverse Proxy – NGINX  
<https://www.youtube.com/watch?v=lZVAI3PqgHc>
- Setting up an Nginx Reverse Proxy – Tech Blog  
<https://www.youtube.com/watch?v=yiO7dm4cIMw>
- Deploying DotNet Core to Linux | Blazor Deploy Linux – Coding Droplets  
<https://www.youtube.com/watch?v=bXK-F-uL7Qo>
- **Cómo subir Web C# .NET a VPS y HOSTING – Maxi Programa.**  
<https://www.youtube.com/watch?v=jqHWOcO7FyM>

- ¿Qué es un VPS y cuál elegir en 2025? – Victor Robles WEB.  
<https://www.youtube.com/watch?v=sBPbWdMU1FM>
- Microsoft SQL Server desde Docker Tutorial – Fazt Code.  
<https://www.youtube.com/watch?v=uHz9xOiaBbw>