

Trabajo Práctico Final - Caso de Uso

Pacheco, Martín

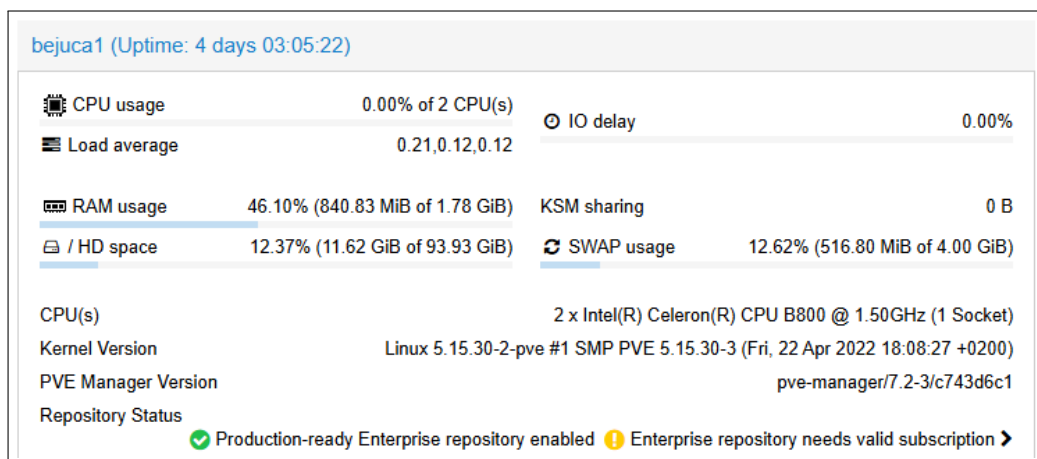
5K3 - 2024

1 INTRODUCCIÓN

En este trabajo se desarrolla el proceso de despliegue de un blog personal sobre el entorno virtual Proxmox, haciendo uso de dos contenedores LXC. El primero de ellos será utilizado para una base de datos relacional PostgreSQL, mientras que el segundo para el acceso a la aplicación misma, mediante un proxy inverso establecido con Nginx.

2 DESCRIPCIÓN DEL NODO

El nodo de Proxmox elegido para el desarrollo del trabajo es el denominado **bejuca1**, el cual tiene las siguientes características:



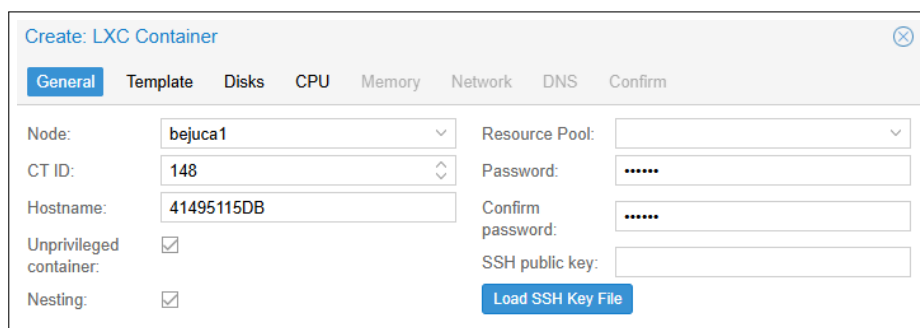
3 BASE DE DATOS

3.1 Creación del Contenedor

El contenedor LXC utilizado para la base de datos debe cumplir con las siguientes restricciones:

- Nombre: 41495115DB
- Características:
 - Procesamiento: 1 procesador
 - Memoria: 128 MB
 - Almacenamiento: 8 GB
 - Redes: Cliente DHCP


Para poder crear el contenedor, se debe seleccionar la opción **Create CT** de la plataforma Proxmox. De esta forma se presenta el wizard que guiará a lo largo del proceso de creación. El primer paso consiste en el ingreso de los datos generales del contenedor. Esto incluye la selección del nodo **bejuca1**, el ingreso del Hostname **41495115DB** y la contraseña para el usuario **root**.



The screenshot shows the 'Create: LXC Container' wizard in Proxmox, specifically the 'General' tab. The form includes the following fields and options:

- Node:** bejuca1 (dropdown)
- CT ID:** 148 (spin box)
- Hostname:** 41495115DB (text input)
- Resource Pool:** (empty dropdown)
- Password:** (masked text input)
- Confirm password:** (masked text input)
- SSH public key:** (empty text input)
- Unprivileged container:** ☒
- Nesting:** ☒
- Buttons:** 'Load SSH Key File' (blue)

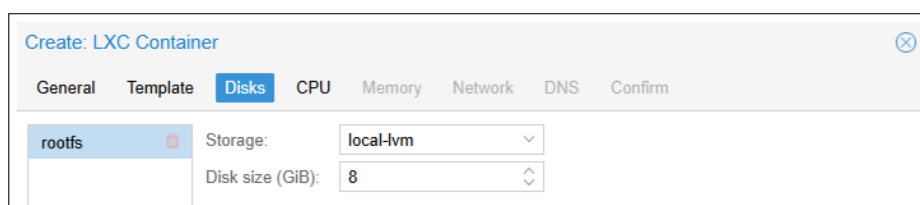
A continuación se elige la unidad de almacenamiento **local** y se elige el template usado para la creación del contenedor. En este caso se se hizo uso de **Ubuntu 20**.



The screenshot shows the 'Create: LXC Container' wizard in Proxmox, specifically the 'Template' tab. The form includes the following fields and options:

- Storage:** local (dropdown)
- Template:** ubuntu-20.04-standard_20.04-1_a (dropdown)

Se configura el espacio de almacenamiento a utilizar, administrado del LVM local. Como se expuso en las restricciones, será de **8 GB**. Seguido de esto indicamos el uso de **1 procesador**.



The screenshot shows the 'Create: LXC Container' wizard in Proxmox, specifically the 'Disks' tab. The form includes the following fields and options:

- rootfs** (selected disk entry)
- Storage:** local-lvm (dropdown)
- Disk size (GiB):** 8 (spin box)

Create: LXC Container

General Template Disks **CPU** Memory Network DNS Confirm

Cores: 1

La cantidad de memoria principal es de **128 MB**.

Create: LXC Container

General Template Disks CPU **Memory** Network DNS Confirm

Memory (MiB): 128

Swap (MiB): 128

Se indica el uso de **DHCP** para la obtención de la dirección IPv4 del contenedor.

Create: LXC Container

General Template Disks CPU Memory **Network** DNS Confirm

Name: eth0 IPv4: ☐ Static ☒ DHCP

MAC address: auto IPv4/CIDR:

Bridge: vmbro Gateway (IPv4):

VLAN Tag: no VLAN IPv6: ☒ Static ☐ DHCP ☐ SLAAC

Rate limit (MB/s): unlimited IPv6/CIDR: None

Firewall: ☒ Gateway (IPv6):

Create: LXC Container

General Template Disks CPU Memory Network **DNS** Confirm

DNS domain: use host settings

DNS servers: 1.1.1.1

Antes de finalizar se muestra de forma resumida sus datos. Al seleccionar **Finish** se inicia la creación del contenedor, la cual se indica como exitosa mediante el mensaje **"TASK OK"**.

Create: LXC Container

General
Template
Disks
CPU
Memory
Network
DNS
Confirm

Key ↑	Value
cores	1
features	nesting=1
hostname	41495115DB
memory	128
nameserver	1.1.1.1
net0	name=eth0,bridge=vbr0,firewall=1,ip=dhcp
nodename	bejuca1
ostemplate	local:vztmpl/ubuntu-20.04-standard_20.04-1_amd64.tar.gz
pool	
rootfs	local-lvm:8
swap	128
unprivileged	1
vmid	148

☐ Start after created

Advanced ☐
Back
Finish

Task viewer: CT 148 - Create

Output
Status

Stop

```

Logical volume "vm-148-disk-0" created.
Creating filesystem with 2097152 4k blocks and 524288 inodes
Filesystem UUID: 7064ebec-9a91-45e2-939d-d51007110f74
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632
extracting archive '/var/lib/vz/template/cache/ubuntu-20.04-standard_20.04-1_amd64.tar.gz'
Total bytes read: 669050880 (639MiB, 17MiB/s)
Detected container architecture: amd64
Creating SSH host key 'ssh_host_rsa_key' - this may take some time ...
done: SHA256:tSIKyhhagC8TDKcjr9qYFwNlNqlsGkNTk98WrNPTUDM root@41495115DB
Creating SSH host key 'ssh_host_ed25519_key' - this may take some time ...
done: SHA256:zTZCxWE5a+P5C6vrixYekwqW17mMLicXkmm8kwKQ8E root@41495115DB
Creating SSH host key 'ssh_host_dsa_key' - this may take some time ...
done: SHA256:VqJ4ZE6Of1kl/PQrvjSnU4PIh31z/6CP5TwJASaa3k root@41495115DB
Creating SSH host key 'ssh_host_ecdsa_key' - this may take some time ...
done: SHA256:TR7wk7FJ9uZR4o0WrsWkzgGzmXR+XWY76XXDGTlBgn8 root@41495115DB
TASK OK

```

Una vez creado el contenedor, lo iniciamos y accedemos a él mediante la terminal de Proxmox. Obtenemos su dirección IP mediante el comando `ip a`. En este caso la dirección es **192.168.77.226/24**.

```

2: eth0@if13: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 1e:f7:82:06:5a:84 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.77.226/24 brd 192.168.77.255 scope global dynamic eth0
        valid_lft 458sec preferred_lft 458sec
    inet6 fe80::1cf7:82ff:fe06:5a84/64 scope link
        valid_lft forever preferred_lft forever

```

3.2 Instalación y Configuración de PostgreSQL

Antes de seguir con la instalación de la base de datos, se debe actualizar los repositorios del administrador de paquetes mediante el comando `apt-get update`. Una vez hecho esto, es

posible descargar PostgreSQL con `apt-get install postgresql`. En este caso la última versión disponible en los repositorios de Ubuntu es la 12. Los archivos de configuración de interés se encuentran en el directorio `/etc/postgresql/12/main/`.

```
root@41495115DB:~# ls -l /etc/postgresql/12/main/
total 56
drwxr-xr-x 2 postgres postgres 4096 Jun 16 23:04 conf.d
-rw-r--r-- 1 postgres postgres 315 Jun 16 23:04 environment
-rw-r--r-- 1 postgres postgres 143 Jun 16 23:04 pg_ctl.conf
-rw-r----- 1 postgres postgres 4933 Jun 16 23:04 pg_hba.conf
-rw-r----- 1 postgres postgres 1636 Jun 16 23:04 pg_ident.conf
-rw-r--r-- 1 postgres postgres 26874 Jun 16 23:04 postgresql.conf
-rw-r--r-- 1 postgres postgres 317 Jun 16 23:04 start.conf
```

El primer paso de la configuración de PostgreSQL consiste en la edición de `postgresql.conf`. Para permitir el acceso remoto a la base de datos se debe cambiar el valor de la variable `listen_addresses` a `'*'` en la sección **CONNECTIONS AND AUTHENTICATION**. Por otro lado, como se destinará el puerto **1433** para el acceso, se debe cambiar el valor de `port` para reflejar tal decisión.

```
#-----
# CONNECTIONS AND AUTHENTICATION
#-----

# - Connection Settings -

listen_addresses = '*'           # what IP address(es) to listen on;
                                # comma-separated list of addresses;
                                # defaults to 'localhost'; use '*' for all
                                # (change requires restart)
port = 1433                      # (change requires restart)
max_connections = 100            # (change requires restart)
#superuser_reserved_connections = 3 # (change requires restart)
unix_socket_directories = '/var/run/postgresql' # comma-separated list of directories
                                # (change requires restart)
```

Se debe establecer la contraseña del usuario **postgres** para el acceso remoto. Para ello se hace un query a la tabla **template1** mediante la interfaz de comando de PostgreSQL **psql**. El comando para acceder a ella es:

```
sudo -u postgres psql template1
```

Mientras que el query se muestra a continuación (se debe reemplazar `[contraseña]` con la contraseña deseada):

```
ALTER USER postgres WITH ENCRYPTED PASSWORD '[contraseña]';
```

```
root@41495115DB:/etc/postgresql/12/main# sudo -u postgres psql template1
sudo: setrlimit(RLIMIT_CORE): Operation not permitted
psql (12.18 (Ubuntu 12.18-0ubuntu0.20.04.1))
Type "help" for help.

template1=# ALTER USER postgres WITH ENCRYPTED PASSWORD '[redacted]';
ALTER ROLE
```

A continuación se debe editar el archivo `pg_hba.conf` para finalizar con la configuración del acceso remoto. La adición de las siguientes líneas la conexión remota (host) a todas las bases de datos (all) y bajo todos los usuarios (all), provenientes de cualquier dirección IP (0.0.0.0/24) con

el uso de contraseñas encriptadas (md5). Una vez hecho todo esto, el servicio de PostgreSQL debe reiniciarse mediante `systemctl restart postgresql`. El comando `netstat -nlt` indica que PostgreSQL efectivamente está haciendo uso del puerto 1433 para establecer conexiones remotas.

```
# Database administrative login by Unix domain socket
local    all             postgres              peer

# TYPE  DATABASE  USER  ADDRESS  METHOD

# "local" is for Unix domain socket connections only
local    all             all                  peer
# IPv4 local connections:
host     all             all                127.0.0.1/32      md5
# IPv6 local connections:
host     all             all                ::1/128           md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
local    replication     all                  peer
host     replication     all                127.0.0.1/32      md5
host     replication     all                ::1/128           md5

# Allow remote connections
host     all             all                0.0.0.0/0         md5
host     all             all                :::/0             md5

root@41495115DB:/etc/postgresql/12/main# systemctl restart postgresql
root@41495115DB:/etc/postgresql/12/main# netstat -nlt
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:25           0.0.0.0:*              LISTEN
tcp        0      0 0.0.0.0:1433           0.0.0.0:*              LISTEN
tcp        0      0 127.0.0.0.53:53        0.0.0.0:*              LISTEN
tcp6       0      0 :::22                  :::*                    LISTEN
tcp6       0      0 :::1433                 :::*                    LISTEN
tcp6       0      0 :::1:25                 :::*                    LISTEN
root@41495115DB:/etc/postgresql/12/main#
```

3.3 Población de la Base de Datos

Una opción para la creación y poblado de la base de datos es mediante una conexión remota usando herramientas como DBeaver o pgAdmin. Caso contrario, es posible ejecutar queries desde el contenedor mediante la consola de `psql` de PostgreSQL. Primero descargamos los scripts correspondientes con el uso del comando `wget` dentro de un directorio perteneciente al usuario `postgres`. Los enlaces de descarga son los siguientes:

- Creación:

<https://docs.google.com/uc?export=download&id=1g9KtkV8MvaTGj86Sp6ovdEk4UkPSDxVd>

- Poblado:

https://docs.google.com/uc?export=download&id=11Hx4IUoq-cnU7IH6-06ksDdx3-cb2T-_

- Creación y poblado:

https://docs.google.com/uc?export=download&id=1_dFzZNgXIOQ90mxY1ktu00W2zZZPdQm1

```
root@41495115DB:~# cd /home/
root@41495115DB:/home# mkdir scripts
root@41495115DB:/home# chown postgres:postgres scripts/
root@41495115DB:/home# ls -l
total 4
drwxr-xr-x 2 postgres postgres 4096 Jun 18 03:47 scripts
```

```
create_full.sql 100%[=====>] 3.37K --.-KB/s in 0s
2024-06-18 03:51:34 (22.2 MB/s) - 'create_full.sql' saved [3447/3447]
root@41495115DB:/home/scripts# ls
create_full.sql
```

Para crear la base de datos `blog` desde la consola `psql` :

```
CREATE DATABASE blog;
```

```
root@41495115DB:/home/scripts# sudo -u postgres psql
sudo: setrlimit(RLIMIT_CORE): Operation not permitted
psql (12.18 (Ubuntu 12.18-0ubuntu0.20.04.1))
Type "help" for help.

postgres=# CREATE DATABASE blog;
CREATE DATABASE
```

La ejecución de los scripts se realiza mediante la directiva `\i` , indicando el path a cada archivo. Es posible que en la descarga se hayan introducido espacios vacíos en la primera línea de cada script, lo cual resulta en un error de ejecución. Es posible editar esto usando nano.

```
\i /home/scripts/create_full.sql
```

```

postgres=# \c blog
You are now connected to database "blog" as user "postgres".
blog=# \i /home/scripts/create_full.sql
CREATE TABLE
START TRANSACTION
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE INDEX
CREATE INDEX
CREATE INDEX
CREATE INDEX
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
COMMIT

```

4 APLICACIÓN

4.1 Descripción

4.2 Creación del Contenedor

El contenedor LXC utilizado para la aplicación del blog debe cumplir con las siguientes restricciones:

- Nombre: 41495115A
- Características:
 - Procesamiento: 1 procesador
 - Memoria: 128 MB
 - Almacenamiento: 8 GB
 - Redes: Cliente DHCP

El proceso de creación es idéntico al del contenedor de la base de datos, difiriendo en el Hostname, contraseña y template usados. En este caso se eligió **Debian 11** debido a su estabilidad y mejor uso de recursos, cuestiones importantes para el despliegue de la aplicación del blog.

Create: LXC Container

General

Template

Disks

CPU

Memory

Network

DNS

Confirm

Node:

bejuca1

Resource Pool:

CT ID:

145

Password:

Hostname:

41495115A

Confirm password:

Unprivileged container:

☒

SSH public key:

Nesting:

☒

Load SSH Key File

Create: LXC Container

General
Template
Disks
CPU
Memory
Network
DNS
Confirm

Storage:
local

Template:
debian-11-standard_11.6-1_amd64

Obtenemos la dirección IP del contenedor con el comando `ip a`. Esta resulta ser **192.168.77.227/24**.

```
2: eth0@if9: <BROADCAST,MULTICAST,UP,LOWER UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether ae:a0:bf:79:47:52 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.77.227/24 brd 192.168.77.255 scope global dynamic eth0
        valid_lft 560sec preferred_lft 560sec
    inet6 fe80::aca0:bfff:fe79:4752/64 scope link
        valid_lft forever preferred_lft forever
```

4.3 Instalación y Configuración de la Aplicación

Para poder desplegar la aplicación, primero es necesario instalar el runtime de ASP.NET Core 6.0. Para ello se debe hacer uso de los repositorios oficiales de Microsoft y el administrador de paquetes:

```
root@41495115A:~# wget https://packages.microsoft.com/config/debian/11/packages-microsoft-prod.deb -O packages-microsoft-prod.deb
--2024-06-16 23:27:11-- https://packages.microsoft.com/config/debian/11/packages-microsoft-prod.deb
Resolving packages.microsoft.com (packages.microsoft.com)... 13.107.246.33, 2620:1ec:bdf::33
Connecting to packages.microsoft.com (packages.microsoft.com)|13.107.246.33|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3134 (3.1K) [application/octet-stream]
Saving to: 'packages-microsoft-prod.deb'

packages-microsoft-prod.deb      100%[=====] 3.06K  --.-KB/s  in 0s
2024-06-16 23:27:12 (33.8 MB/s) - 'packages-microsoft-prod.deb' saved [3134/3134]
```

```
root@41495115A:~# dpkg -i packages-microsoft-prod.deb
Selecting previously unselected package packages-microsoft-prod.
(Reading database ... 19030 files and directories currently installed.)
Preparing to unpack packages-microsoft-prod.deb ...
Unpacking packages-microsoft-prod (1.0-debian10.1) ...
Setting up packages-microsoft-prod (1.0-debian10.1) ...
root@41495115A:~# rm packages-microsoft-prod.deb
```

```
root@41495115A:~# apt-get update
Hit:1 http://security.debian.org bullseye-security InRelease
Hit:2 http://deb.debian.org/debian bullseye InRelease
Hit:3 http://deb.debian.org/debian bullseye-updates InRelease
Get:4 https://packages.microsoft.com/debian/10/prod buster InRelease [6538 B]
Get:5 https://packages.microsoft.com/debian/10/prod buster/main arm64 Packages [29.1 kB]
Get:6 https://packages.microsoft.com/debian/10/prod buster/main amd64 Packages [227 kB]
Get:7 https://packages.microsoft.com/debian/10/prod buster/main armhf Packages [23.8 kB]
Get:8 https://packages.microsoft.com/debian/10/prod buster/main all Packages [2393 B]
Fetched 289 kB in 1s (215 kB/s)
Reading package lists... Done
```

La instalación del runtime es posible mediante el comando:

```
apt-get install aspnetcore-runtime-6.0
```

Para comprobar si fue exitosa es posible usar `dotnet --info` y asegurar que el runtime 6.0 se encuentra disponible para su uso.


```
GNU nano 5.4 appsettings.json *
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "PostgreSQL": "Host=192.168.1.226;Port=1433;Username=postgres;Password=;Database=blog;"
  },
  "Legajo": 48311
}
```

Comprobar el funcionamiento de la aplicación requiere iniciar el runtime usando el archivo .dll del blog y ejecutar `curl` contra la dirección y puerto indicados:

```
dotnet Blog.Presentation.dll
curl localhost:5000
```

Antes de continuar con el trabajo, la aplicación puede ser apagada o ser movida a una ejecución de fondo (job). La primera opción requiere presionar `Ctrl+C`, mientras que la segunda requiere presionar `Ctrl+Z` seguida del comando `bg`. Traerla de vuelta a primer plano se hace mediante el comando `fg`.

```
root@41495115A:/var/Blog# dotnet Blog.Presentation.dll
warn: Microsoft.AspNetCore.DataProtection.KeyManagement.XmlKeyManager[35]
      No XML encryptor configured. Key {aa18b6cb-30db-443d-8d4a-9bb1823404f9} may be persisted to storage in unencrypted form.
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/Blog/
^Z
[1]+  Stopped                  dotnet Blog.Presentation.dll
root@41495115A:/var/Blog# bg
[1]+ dotnet Blog.Presentation.dll &
```

4.4 Instalación y Configuración de la Nginx

Se hará uso de Nginx como proxy inverso para redireccionar las peticiones al servidor Kestrel de la aplicación ASP.NET. Para ello lo instalamos con el comando `apt-get install nginx` y comprobamos su funcionamiento con `systemctl`.

```
root@41495115A:/var/Blog# systemctl status nginx
* nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2024-06-16 23:37:28 UTC; 45s ago
     Docs: man:nginx(8)
  Process: 1894 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
  Process: 1906 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
 Main PID: 2107 (nginx)
   Tasks: 2 (limit: 2107)
  Memory: 5.6M
     CPU: 74ms
  CGroup: /system.slice/nginx.service
          └─2107 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
             └─2110 nginx: worker process

Jun 16 23:37:28 41495115A systemd[1]: Starting A high performance web server and a reverse proxy server...
Jun 16 23:37:28 41495115A systemd[1]: nginx.service: Failed to parse PID from file /run/nginx.pid: Invalid argument
Jun 16 23:37:28 41495115A systemd[1]: Started A high performance web server and a reverse proxy server.
```

Modificamos la configuración de bloque **location** del **default server** desde el archivo `default`, ubicado en el directorio `/etc/nginx/sites-available/` para establecer el proxy inverso. Esto requiere asignar la dirección IP y puerto de la aplicación en Kestrel a la variable `proxy_pass`, así como modificar ciertos parámetros del header (identificación del cliente, host, entre otros).

```
location / {
    proxy_pass http://127.0.0.1:5000;
    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;
}
```

Es posible verificar que el proxy inverso fue configurado correctamente mediante el comando `nginx -t`. Si la respuesta es positiva, luego de reiniciar Nginx y ejecutar la aplicación del blog nuevamente, correr `curl localhost:80` (el proxy inverso) debería ser equivalente a `curl localhost:5000` (la aplicación).

```
root@41495115A:/var/Blog# curl localhost:80
<!DOCTYPE html>
<html lang="en">
<head>
  <link rel="icon" href="/resources/images/logo/ulogo_w.png" type="image/x-icon">
  <title>Gallery - UNETERNAL</title>
  <div b-396btt4ex9 id="meta-tags">
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="language" content="en">
    <meta name="description" content="Uneternal Website">
    <meta name="revised" content="Martin Pacheco, 10/25/2023">
    <meta name="author" content="Martin Pacheco">
    <meta name="viewport" content="width=device-width, initial-scale=1">
  </div>
  <!--CSS-->
  <link rel="stylesheet" href="/css/layout.css">
  <link rel="stylesheet" href="/css/common.css">
  <link rel="stylesheet" href="/css/animations.css">
  <!--<link rel="stylesheet" href="/css/mobile.css" media="(max-width: 600px)">
```

4.5 Creación del Servicio

Es posible crear un servicio para la aplicación, el cual puede ser administrado por medio de `systemctl` de manera similar a PostgreSQL y Nginx. Para ello creamos un archivo `blog.service` dentro del directorio `/etc/systemd/system/` con los siguientes contenidos:

```
GNU nano 5.4
[Unit]
Description=Blog App

[Service]
WorkingDirectory=/var/Blog/
ExecStart=/usr/bin/dotnet /var/Blog/Blog.Presentation.dll
Restart=always
RestartSec=10
KillSignal=SIGINT
SyslogIdentifier=dotnet-blog
User=root
Environment=ASPNETCORE_ENVIRONMENT=Production
Environment=DOTNET_PRINT_TELEMETRY_MESSAGE=false

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

La creación del servicio requiere ejecutar el comando `systemctl enable blog.service`. Ahora es posible iniciar la aplicación mediante `systemctl start blog.service`, así como

también reiniciarla o detenerla.

```
root@41495115A:~# systemctl start blog.service
root@41495115A:~# systemctl status blog.service
* blog.service - Blog App
   Loaded: loaded (/etc/systemd/system/blog.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2024-06-17 00:38:39 UTC; 7s ago
     Main PID: 429 (dotnet)
        Tasks: 11 (limit: 2107)
      Memory: 52.7M
         CPU: 725ms
       CGroup: /system.slice/blog.service
              └─429 /usr/bin/dotnet /var/Blog/Blog.Presentation.dll

Jun 17 00:38:39 41495115A systemd[1]: Started Blog App.
root@41495115A:~# systemctl stop blog.service
```