




ASIGNATURA:
Virtualización
Consolidación de Servidores

AÑO: 2023

PROFESOR: Carriles, Luis
María

Trabajo Practico
Final

Integrantes:
Busto, Kevin Jonás - 48114

	CATEDRA DE VIRTUALIZACIÓN DEPARTAMENTO INGENIERIA EN SISTEMAS DE INFORMACION UNIVERSIDAD TECNOLÓGICA NACIONAL - FACULTAD REGIONAL TUCUMAN	TRABAJO PRÁCTICO FINAL
		CICLO 2023

INTRODUCCIÓN

En este trabajo práctico final, se utilizará Proxmox, una plataforma de virtualización de código abierto, para crear dos contenedores con Apache2. El objetivo principal es configurar un entorno de alojamiento web completo para un blog personal. Para lograr esto, se utilizarán dos contenedores distintos: uno contendrá el backend, desarrollado con Node.js y utilizando una base de datos MariaDB o MySQL, mientras que el otro contenedor se encargue del frontend con JavaScript, HTML y CSS. Además, se utilizará la API Fetch para establecer la comunicación entre el backend y el frontend. Se abordará la implementación del backend del blog web, utilizando Node.js para desarrollar la lógica del servidor y la interacción con la base de datos MariaDB. Se detallarán los pasos necesarios para configurar correctamente Node.js y establecer la conexión con la base de datos, así como la instalación y configuración de los módulos y dependencias necesarios. En cuanto al frontend, se abordará la creación de un contenedor separado donde se alojarán los archivos de JavaScript, HTML y CSS que conformarán el diseño y la funcionalidad del blog web personal. Se explicará cómo instalar Apache2 en este contenedor y se detallará como es la conexión entre contenedores para realizar solicitudes a través de una API, comunicándose así con el backend y accediendo a los datos almacenados en la base de datos en el frontend.

SOBRE PROXMOX


Proxmox es una plataforma de virtualización de código abierto que permite la creación y gestión de contenedores y máquinas virtuales. Es una solución completa que combina la virtualización basada en contenedores (LXC) y la virtualización completa basada en hipervisor en una sola plataforma. Además, proporciona una interfaz de administración web intuitiva que facilita la creación, configuración y supervisión de los contenedores y máquinas virtuales. Permite a los usuarios gestionar los recursos del sistema, como la asignación de CPU, memoria RAM y almacenamiento, de manera eficiente.

CREACIÓN DE CONTENEDORES LXC DE PROXMOX

Para la creación de los contenedores, tanto para backend como frontend, deben cumplir los siguientes requisitos, establecidos por la catedra de Virtualización:

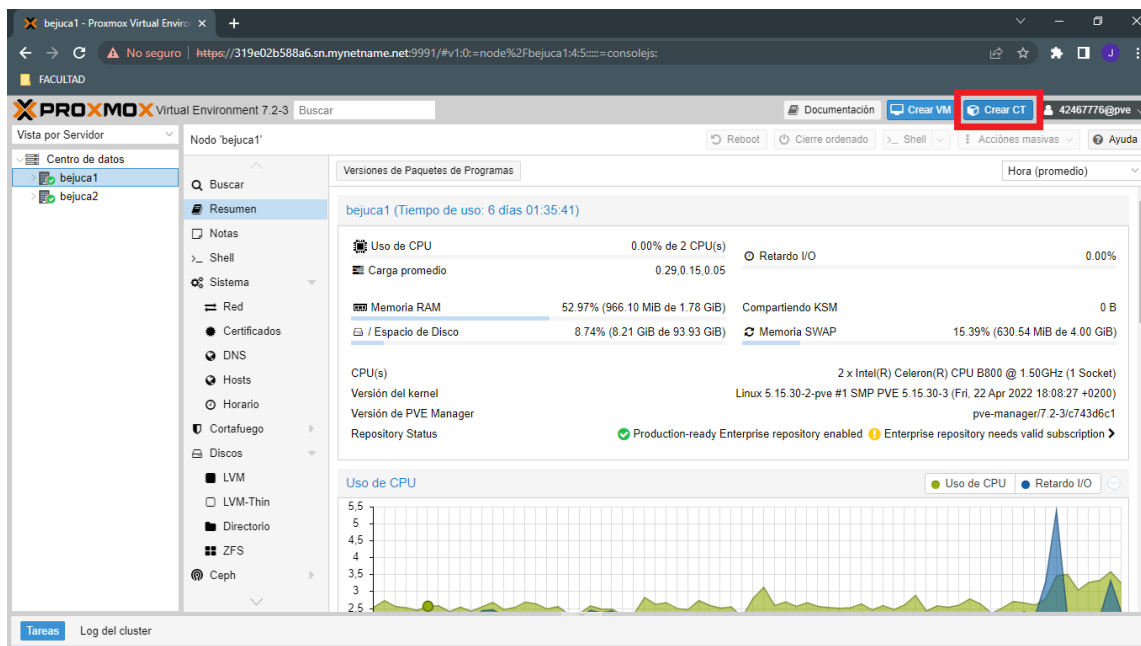
- Nombre del contenedor: "DNI + DB" (Para backend), y "DNI + A" (Para frontend).
- Memoria RAM: 128 MB.
- Almacenamiento: 8GB.
- Procesador: 1 núcleo.
- Red: DHCP.

Para ello debemos acceder a Proxmox y autenticarse en la plataforma, y seguir los siguientes pasos:

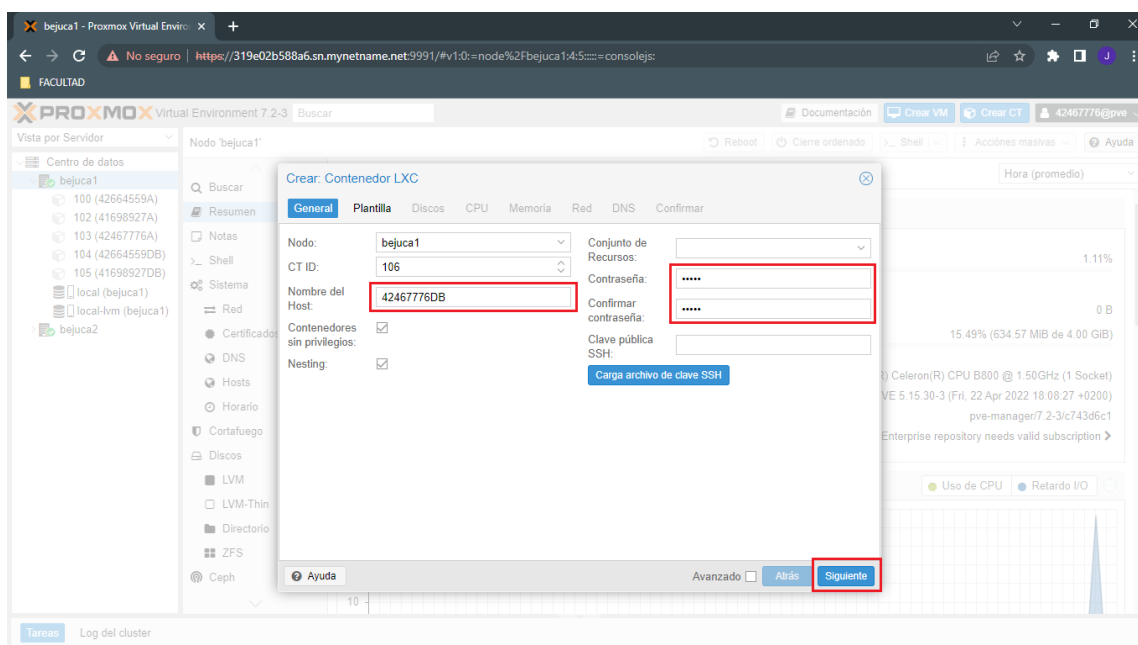
	<p align="center">CATEDRA DE VIRTUALIZACIÓN</p> <p align="center">DEPARTAMENTO INGENIERIA EN SISTEMAS DE INFORMACION</p> <p align="center">UNIVERSIDAD TECNOLÓGICA NACIONAL - FACULTAD REGIONAL TUCUMAN</p>	<p align="center">TRABAJO</p> <p align="center">PRÁCTICO FINAL</p>
		<p align="center">CICLO</p> <p align="center">2023</p>

PASOS PARA LA CREACIÓN DEL CONTENEDOR (CT):

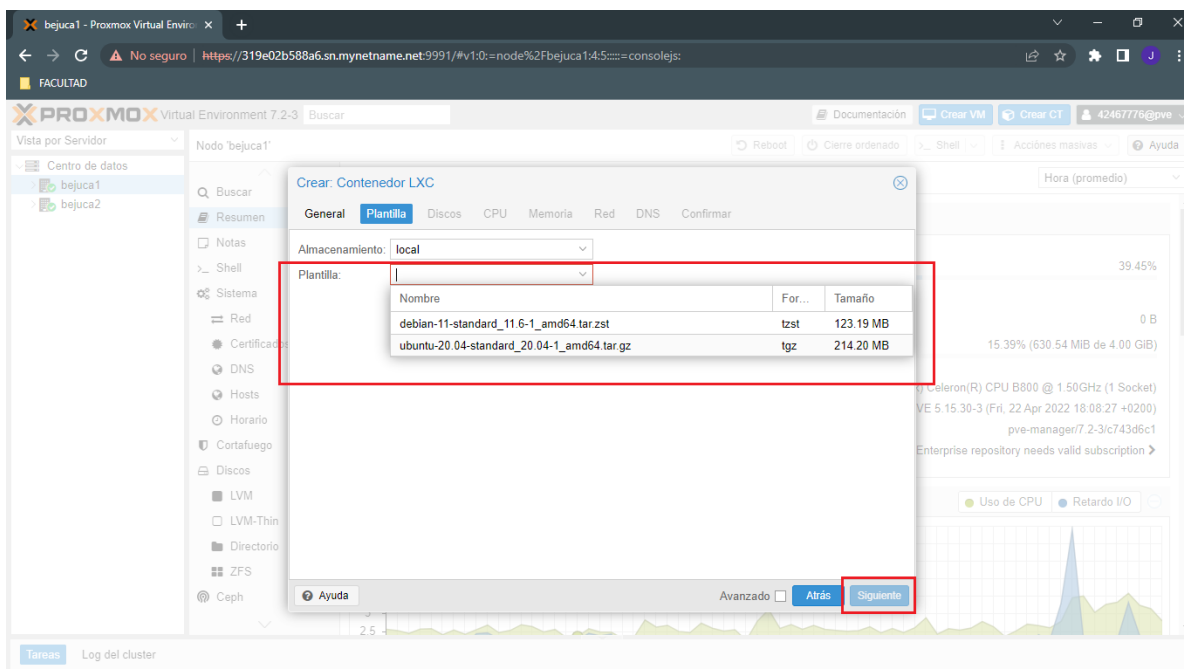
1) Paso 1: Presionamos en el botón '**Crear CT**'.



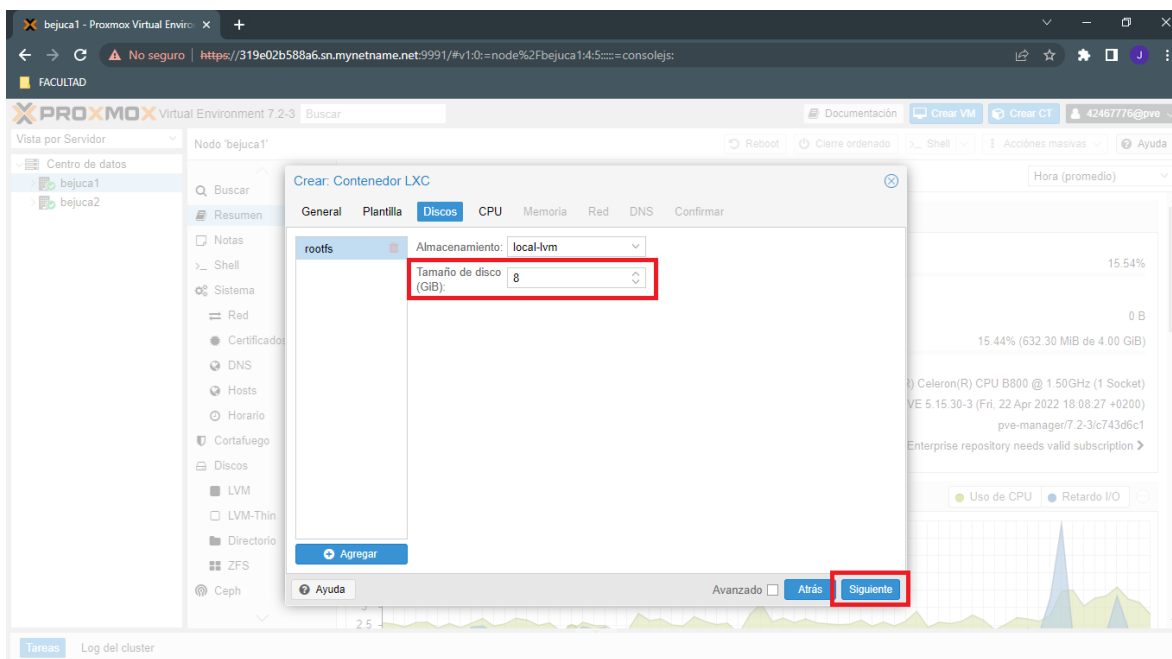
2) Paso 2: Cargamos el nombre del CT, y una contraseña para el mismo. Puede o no ser la misma que la del inicio de sesión. Luego presionar en '**Siguiente**'.




- 3) Paso 3: Elegir la plantilla. Es indistinto elegir una u otra para la futura configuración del CT para el blog web. Luego de la selección, presionar en '**Siguiente**'.

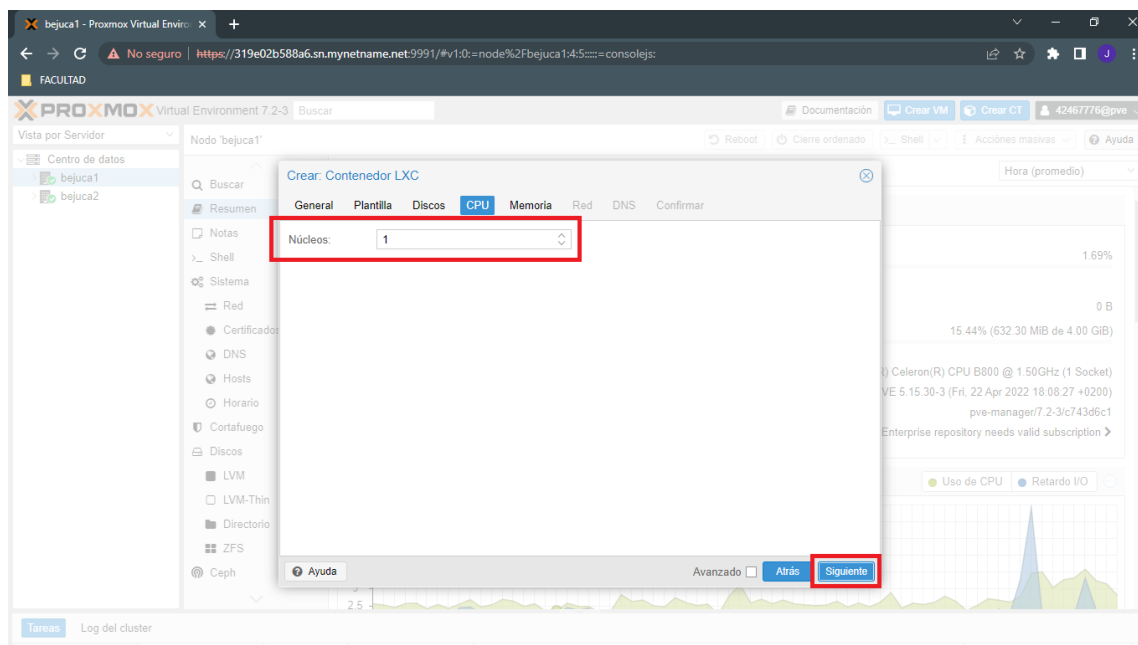


- 4) Paso 4: Elegir el tamaño del disco. Deben ser 8GB para cumplir con los requisitos de la catedra. Luego presionar en '**Siguiente**'.

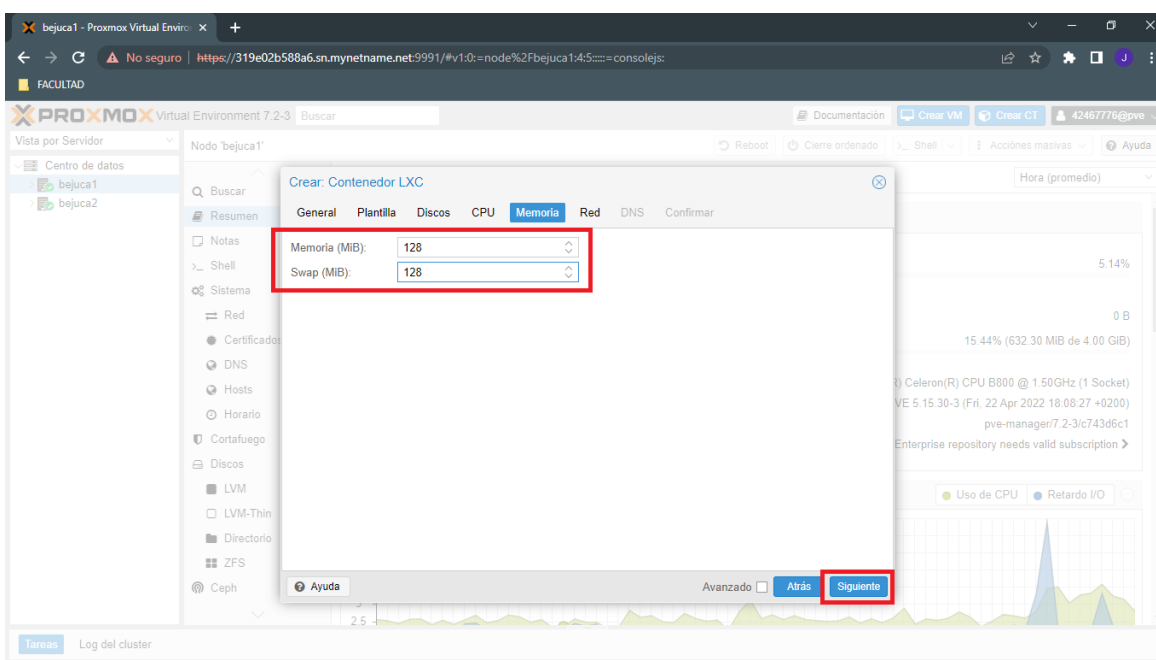



	<p align="center">CATEDRA DE VIRTUALIZACIÓN DEPARTAMENTO INGENIERIA EN SISTEMAS DE INFORMACION UNIVERSIDAD TECNOLOGICA NACIONAL - FACULTAD REGIONAL TUCUMAN</p>	<p align="center">TRABAJO PRÁCTICO FINAL</p>
		<p align="center">CICLO 2023</p>

- 5) Paso 5: Elegir la cantidad de núcleos del procesador. Debe elegirse 1 núcleo para cumplir con los requisitos de la catedra. Luego presionar en '**Siguiente**'.

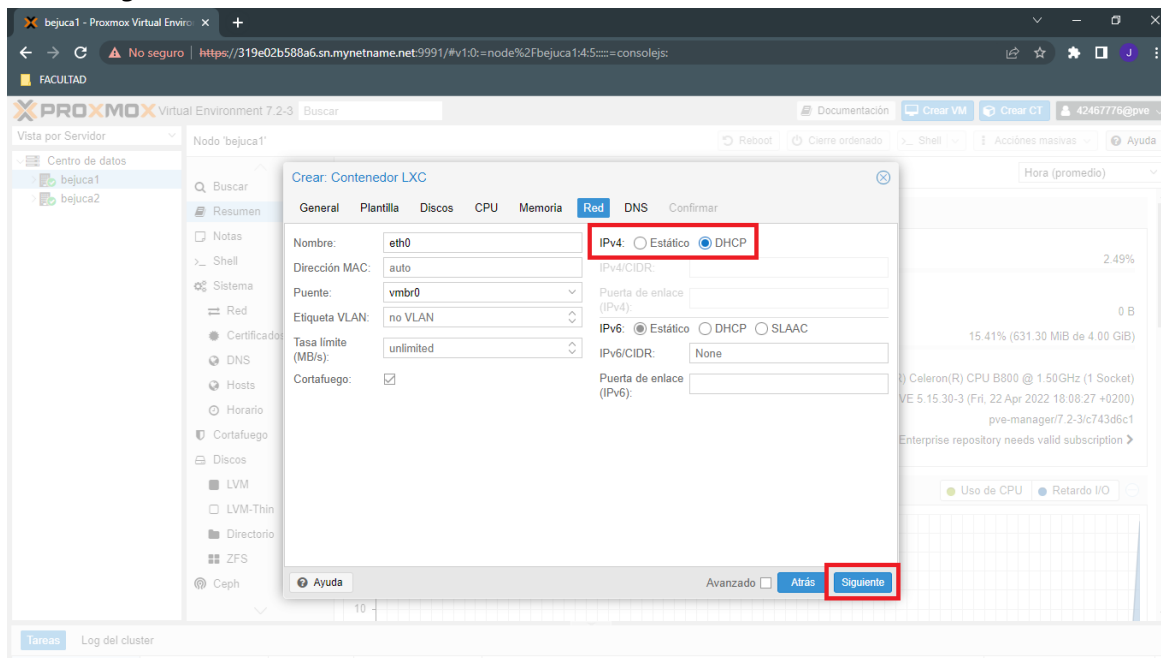


- 6) Paso 6: Elegir la cantidad de memoria RAM. Debe elegirse 128MB de RAM para cumplir con los requisitos de la catedra. Luego presionar en '**Siguiente**'.

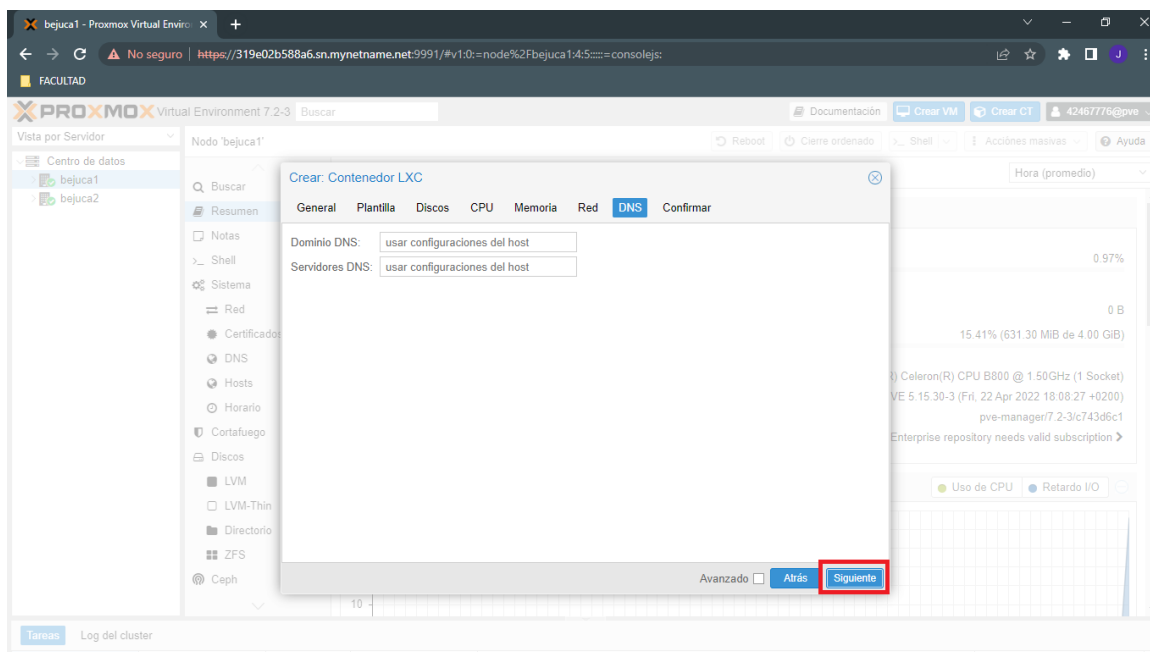


	<p align="center">CATEDRA DE VIRTUALIZACIÓN DEPARTAMENTO INGENIERIA EN SISTEMAS DE INFORMACION UNIVERSIDAD TECNOLOGICA NACIONAL - FACULTAD REGIONAL TUCUMAN</p>	<p align="center">TRABAJO PRÁCTICO FINAL</p>
		<p align="center">CICLO 2023</p>

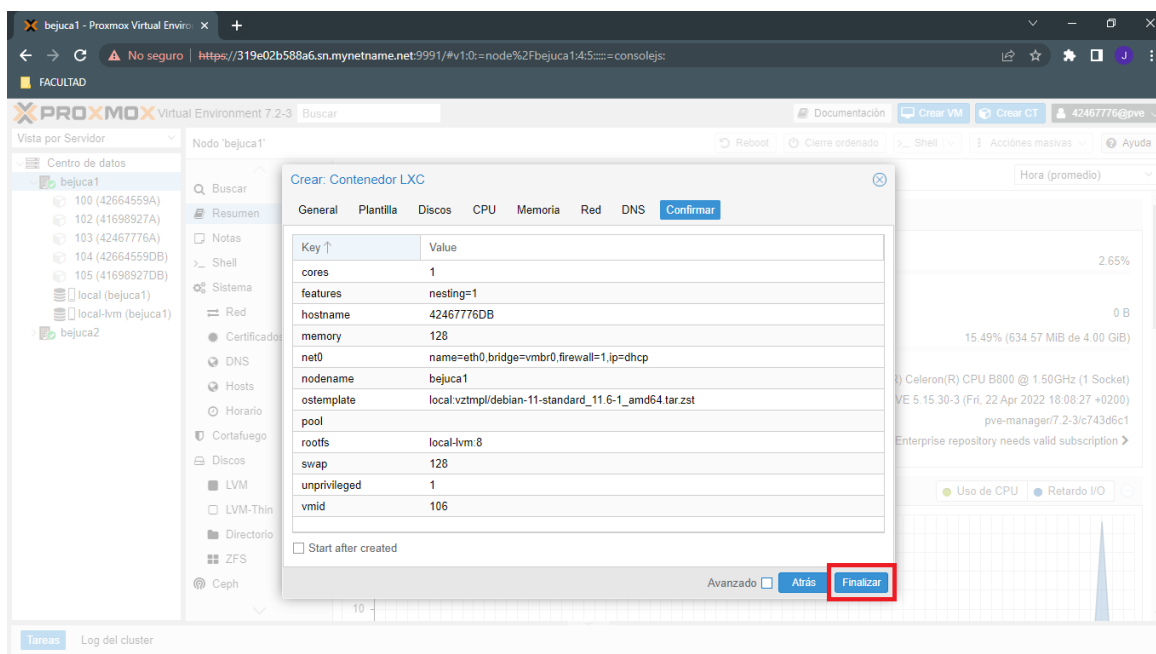
7) Paso 7: Configurar la red eligiendo DHCP por requisitos de la catedra. Luego presionar en '**Siguiente**'.



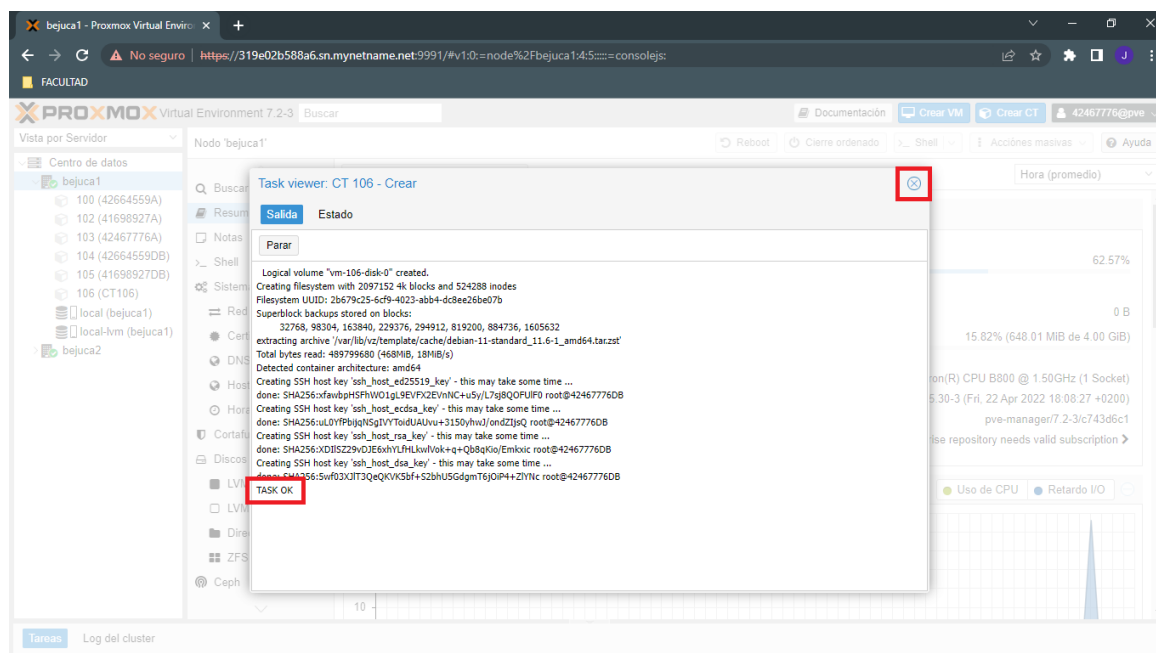
8) Paso 8: No hace falta configurar el DNS. Presionar en '**Siguiente**'.




- 9) Paso 9: Revisar la configuración definida para el CT. Se puede retornar a los pasos anteriores para reconfigurar lo necesario. Sino, presionar en '**Finalizar**'.

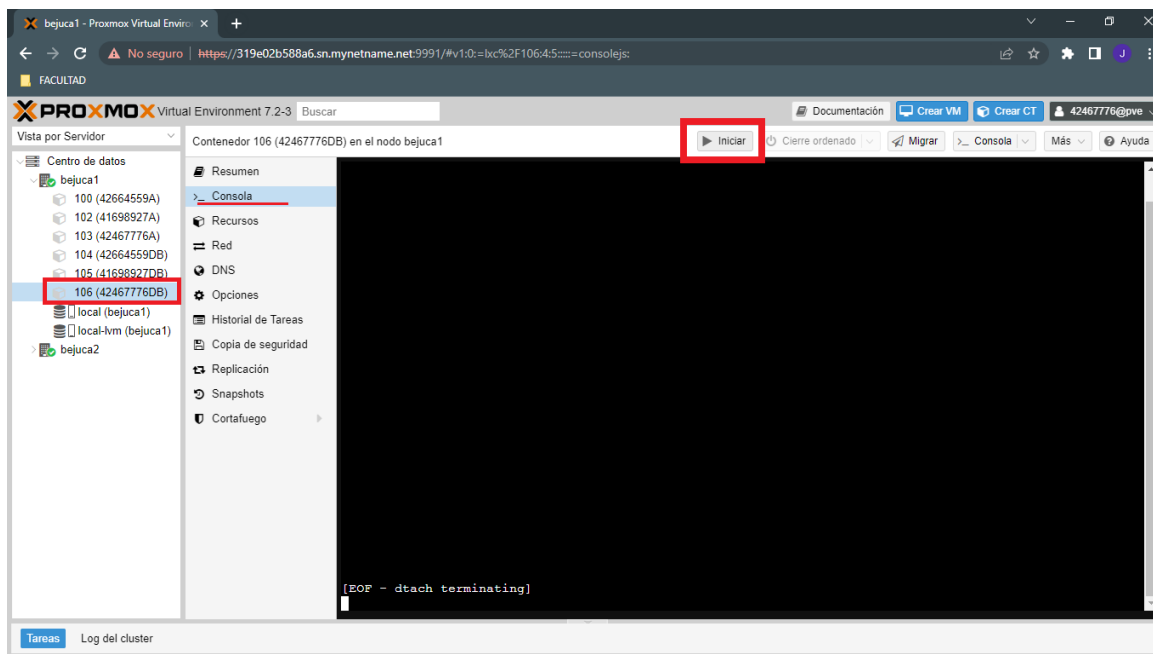


- 10) Paso 10: Esperar a la instalación. Luego chequear si aparece '**TASK OK**', lo cual significa la creación exitosa del CT. Cerrar ventana.

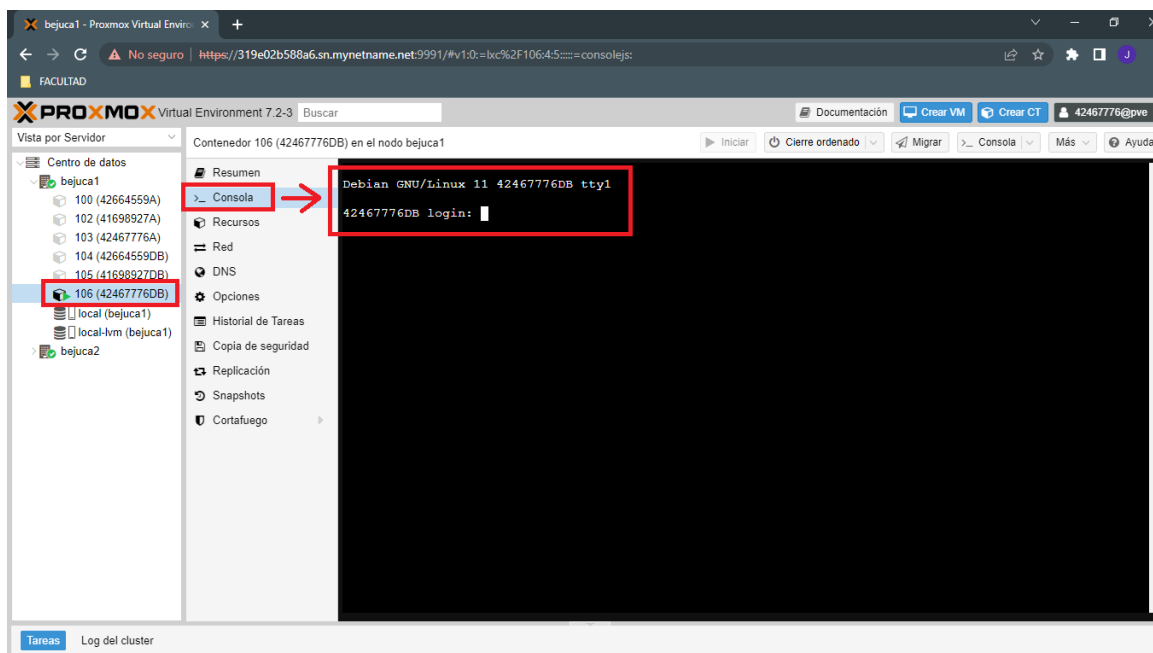



	<p align="center">CATEDRA DE VIRTUALIZACIÓN DEPARTAMENTO INGENIERIA EN SISTEMAS DE INFORMACION UNIVERSIDAD TECNOLOGICA NACIONAL - FACULTAD REGIONAL TUCUMAN</p>	TRABAJO PRÁCTICO FINAL
		CICLO 2023

11) Paso 11: Seleccionar el CT creado como se muestra en la imagen, y presionar en '*Iniciar*' para arrancar el CT.



12) Paso 12: Seleccionando la consola, tal como se muestra en la imagen, debe aparecer la opción de Login si el CT arranco.



	<p align="center">CATEDRA DE VIRTUALIZACIÓN DEPARTAMENTO INGENIERIA EN SISTEMAS DE INFORMACION UNIVERSIDAD TECNOLÓGICA NACIONAL - FACULTAD REGIONAL TUCUMAN</p>	TRABAJO PRÁCTICO FINAL
		CICLO 2023

CONFIGURAR LOS CONTENEDORES

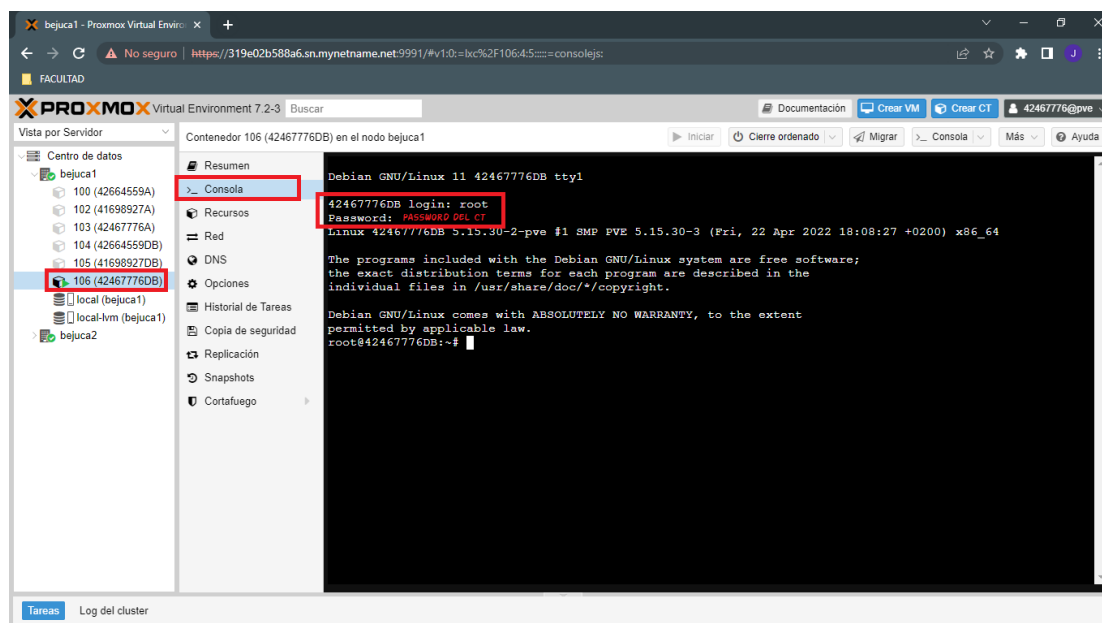
Como se mencionó previamente, se utilizarán dos contenedores en Proxmox: uno para el backend y otro para el frontend. El contenedor del backend contendrá Apache2, Git, Node.js y MariaDB, mientras que el contenedor del frontend contendrá Apache2 y Git. En el contenedor del backend se configurará Apache2 como el servidor web para alojar el backend del blog y se instalarán Git, Node.js y MariaDB. Apache2 será responsable de servir las páginas web y gestionar las solicitudes entrantes, mientras que Git se utilizará para gestionar las versiones del código. Node.js será utilizado para desarrollar la lógica del servidor y gestionar las solicitudes HTTP, y MariaDB se encargará del almacenamiento y gestión de la base de datos del blog. En el contenedor del frontend, se configurará Apache2 como el servidor web para alojar los archivos del blog y se instalará Git para el control de versiones. Apache2 será responsable de servir los archivos HTML, CSS y JavaScript del frontend.

Se abordarán los pasos necesarios para instalar y configurar cada componente de software. Al finalizar la configuración de los contenedores, tendremos un entorno listo para alojar el blog personal.

PASOS PARA LA CONFIGURACIÓN DE UN CONTENEDOR (CT):

Solo se mostrará la configuración del contenedor del backend, ya que es la más completa. La configuración del frontend será exactamente la misma, con la excepción de que únicamente se usará Apache2 y Git, por lo cual Node y MariaDB quedan excluidos para este último contenedor.

- 1) Paso 1: Iniciar sesión con el usuario **'root'** (Para disponer de todos los permisos) y la contraseña del CT previamente definida en la creación del mismo.





CATEDRA DE VIRTUALIZACIÓN
DEPARTAMENTO INGENIERIA EN SISTEMAS DE INFORMACION
UNIVERSIDAD TECNOLOGICA NACIONAL - FACULTAD REGIONAL TUCUMAN

TRABAJO
PRÁCTICO FINAL


CICLO
2023

- 2) Paso 2: Ejecutar el comando **'apt update'** para actualizar los paquetes disponibles del sistema. Esto evitara problemas de dependencia o incompatibilidades con otros paquetes.

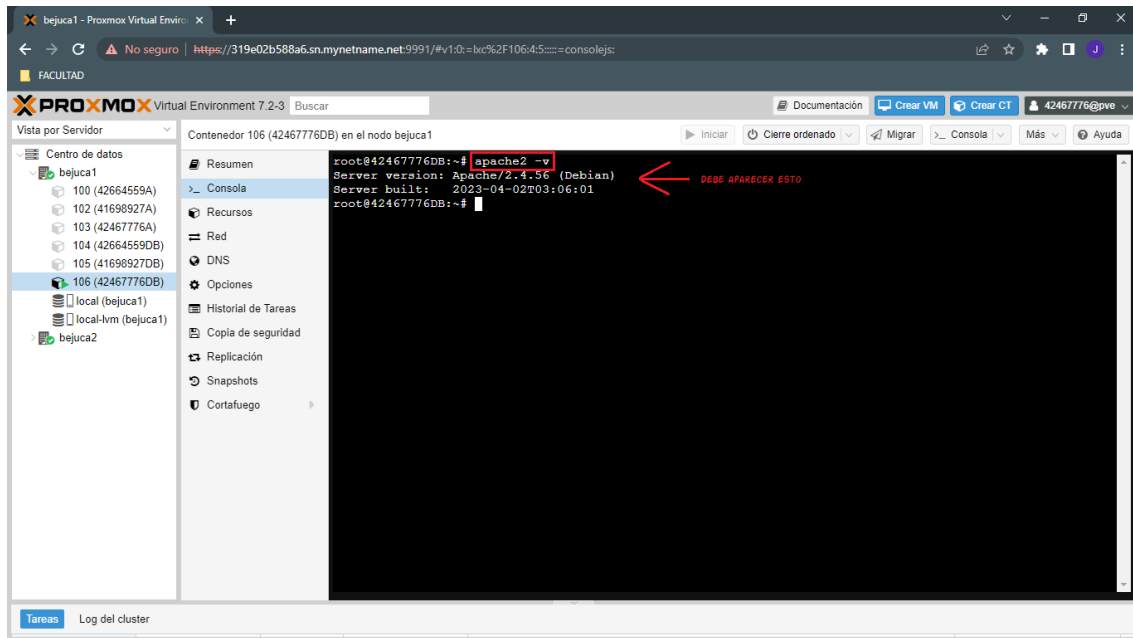
```
root@42467776DB:~# apt update
Get:1 http://deb.debian.org/debian bullseye InRelease [116 kB]
Get:2 http://deb.debian.org/debian bullseye-updates InRelease [44.1 kB]
Get:3 http://security.debian.org bullseye-security InRelease [48.4 kB]
Get:4 http://deb.debian.org/debian bullseye/main amd64 Packages [8183 kB]
Get:5 http://security.debian.org bullseye-security/main amd64 Packages [252 kB]
Get:6 http://security.debian.org bullseye-security/main Translation-en [164 kB]
Get:7 http://deb.debian.org/debian bullseye-updates/main amd64 Packages.diff/Index [18.5 kB]
Get:8 http://deb.debian.org/debian bullseye/main Translation-en [6240 kB]
Get:9 http://deb.debian.org/debian bullseye/contrib amd64 Packages [50.6 kB]
Get:10 http://deb.debian.org/debian bullseye/contrib Translation-en [46.9 kB]
Get:11 http://deb.debian.org/debian bullseye-updates/main Translation-en [8129 B]
Get:12 http://deb.debian.org/debian bullseye-updates/main amd64 Packages T-2023-05-24-2006.01-F-2023-02-20-2009.37.pdiff [1516 B]
Fetched 15.2 MB in 9s (1763 kB/s)
Reading package lists... Done
Building dependency tree... Done
34 packages can be upgraded. Run 'apt list --upgradable' to see them.
N: Repository 'http://deb.debian.org/debian bullseye InRelease' changed its 'Version' value from '11.6' to '11.7'
N: Repository 'http://deb.debian.org/debian bullseye InRelease' changed its 'Suite' value from 'stable' to 'oldstable'
N: Repository 'http://deb.debian.org/debian bullseye-updates InRelease' changed its 'Suite' value from 'stable-updates' to 'oldstable-updates'
N: Repository 'http://security.debian.org bullseye-security InRelease' changed its 'Suite' value from 'stable-security' to 'oldstable-security'
root@42467776DB:~#
```

- 3) Paso 3: Ejecutar el comando **'apt install apache2'** para instalar apache en el CT. Nos pedirá confirmación, y presionamos 'y'.

```
root@42467776DB:~# apt install apache2
Reading package lists... Done
Building dependency tree... Done
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap libcurl4 liblua5.3-0
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap libcurl4 liblua5.3-0
0 upgraded, 10 newly installed, 0 to remove and 34 not upgraded.
Need to get 2834 kB of archives.
After this operation, 9181 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://security.debian.org/debian-security bullseye/main amd64 libapr1 amd64 1.7.0-6+deb11u2 [106 kB]
Get:2 http://deb.debian.org/debian bullseye/main amd64 libaprutil1 amd64 1.6.1-5+deb11u1 [92.3 kB]
Get:3 http://deb.debian.org/debian bullseye/main amd64 libaprutil1-dbd-sqlite3 amd64 1.6.1-5+deb11u1 [19.0 kB]
Get:4 http://deb.debian.org/debian bullseye/main amd64 libcurl4 amd64 7.74.0-1.3+deb11u7 [346 kB]
Get:5 http://deb.debian.org/debian bullseye/main amd64 liblua5.3-0 amd64 5.3.3-1.1+b1 [120 kB]
Get:6 http://deb.debian.org/debian bullseye/main amd64 apache2-bin amd64 2.4.56-1-deb11u2 [1431 kB]
Get:7 http://deb.debian.org/debian bullseye/main amd64 apache2-data all 2.4.56-1-deb11u2 [160 kB]
Get:8 http://deb.debian.org/debian bullseye/main amd64 apache2-utils amd64 2.4.56-1-deb11u2 [265 kB]
Get:9 http://deb.debian.org/debian bullseye/main amd64 apache2 amd64 2.4.56-1-deb11u2 [278 kB]
Fetched 2834 kB in 1s (3213 kB/s)
Selecting previously unselected package libapr1:amd64.
(Reading database ... 19030 files and directories currently installed.)
Preparing to unpack .../0-libapr1 1.7.0-6+deb11u2_amd64.deb ...
Unpacking apache2-utils (2.4.56-1-deb11u2) ...-1-deb11u2_amd64.deb .....deb ...
```

	<p align="center">CATEDRA DE VIRTUALIZACIÓN DEPARTAMENTO INGENIERIA EN SISTEMAS DE INFORMACION UNIVERSIDAD TECNOLOGICA NACIONAL - FACULTAD REGIONAL TUCUMAN</p>	<p align="center">TRABAJO PRÁCTICO FINAL</p>
		<p align="center">CICLO 2023</p>

- 4) Paso 4: Ejecutar el comando '**apache2 -v**' para verificar la existencia y versión de apache en el CT. Si no aparece como en la imagen, volver al 'Paso 3'. Caso contrario, continuar al siguiente.

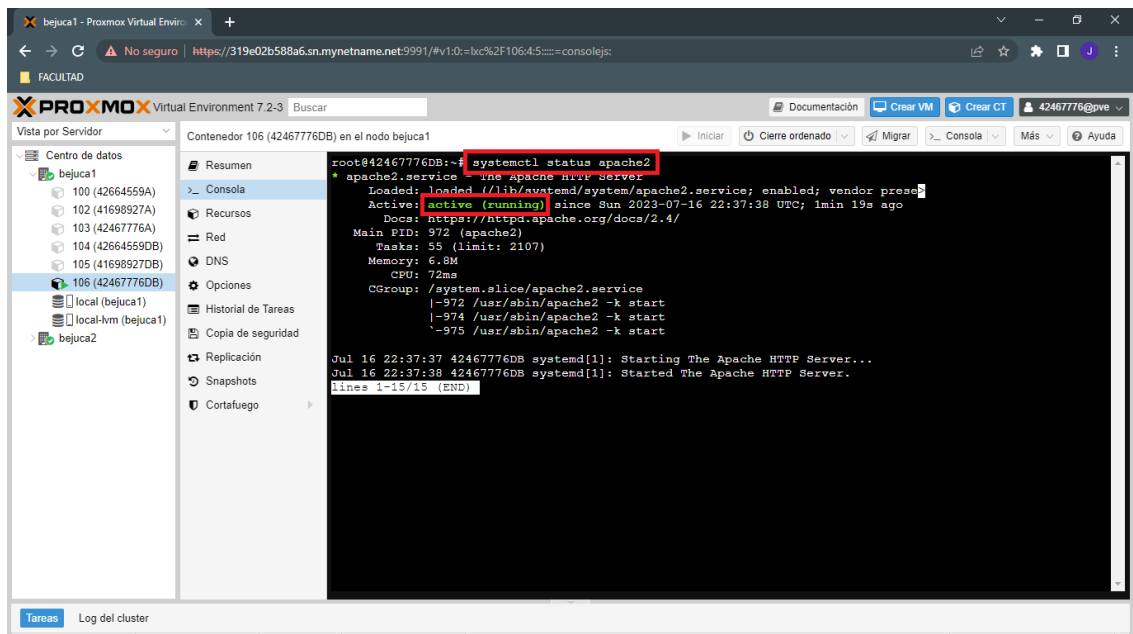


```

root@42467776DB:~# apache2 -v
Server version: Apache/2.4.56 (Debian)
Server built: 2023-04-02T03:06:01
root@42467776DB:~#

```

- 5) Paso 5: Ejecutar el comando '**systemctl status apache2**' para verificar el estado actual del servicio apache2. Debe aparecer '**Running**', sino ejecutar '**systemctl enable apache2**'. Y volver a verificar.



```

root@42467776DB:~# systemctl status apache2
* apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset:
   Active: active (running) since Sun 2023-07-16 22:37:38 UTC; 1min 19s ago
     Docs: https://httpd.apache.org/docs/2.4/
    Main PID: 972 (apache2)
      Tasks: 55 (limit: 2107)
     Memory: 6.8M
        CPU: 72ms
    CGroup: /system.slice/apache2.service
            └─972 /usr/sbin/apache2 -k start
              └─974 /usr/sbin/apache2 -k start
                └─975 /usr/sbin/apache2 -k start

Jul 16 22:37:37 42467776DB systemd[1]: Starting The Apache HTTP Server...
Jul 16 22:37:38 42467776DB systemd[1]: Started The Apache HTTP Server.
lines 1-15/15 (END)

```



CATEDRA DE VIRTUALIZACIÓN
DEPARTAMENTO INGENIERIA EN SISTEMAS DE INFORMACION
UNIVERSIDAD TECNOLOGICA NACIONAL - FACULTAD REGIONAL TUCUMAN

TRABAJO
PRÁCTICO FINAL

CICLO
2023

6) Paso 6: Ejecutar el comando **'apt install mariadb-server'** para instalar MariaDB en el CT.

```
root@42467776DB:~# apt install mariadb-server
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  galera-4 gawk libaio1 libcgf-fast-perl libcgf-pm-perl libclone-perl
  libconfig-inifiles-perl libdbd-mariadb-perl libdbi-perl
  libencode-locale-perl libfcgi-bin libfcgi-perl libfcgi0ldbl
  libhtml-parser-perl libhtml-tagset-perl libhtml-template-perl
  libhttp-date-perl libhttp-message-perl libio-html-perl
  liblwp-mediatypes-perl libmariadb3 libmpfr6 libsigsegv2 libsnappy1v5
  libterm-readkey-perl libtimedate-perl liburi-perl mariadb-client-10.5
  mariadb-client-core-10.5 mariadb-common mariadb-server-10.5
  mariadb-server-core-10.5 mysql-common psmisc rsync socat
Suggested packages:
  gawk-doc libdbd-perl libnet-daemon-perl libsql-statement-perl
  libdata-dump-perl libipc-sharedcache-perl libwww-perl mailx mariadb-test
  netcat-openbsd
The following NEW packages will be installed:
  galera-4 gawk libaio1 libcgf-fast-perl libcgf-pm-perl libclone-perl
  libconfig-inifiles-perl libdbd-mariadb-perl libdbi-perl
  libencode-locale-perl libfcgi-bin libfcgi-perl libfcgi0ldbl
  libhtml-parser-perl libhtml-tagset-perl libhtml-template-perl
  libhttp-date-perl libhttp-message-perl libio-html-perl
  liblwp-mediatypes-perl libmariadb3 libmpfr6 libsigsegv2 libsnappy1v5
  libterm-readkey-perl libtimedate-perl liburi-perl mariadb-client-10.5
  mariadb-client-core-10.5 mariadb-common mariadb-server mariadb-server-10.5
  mariadb-server-core-10.5 mysql-common psmisc rsync socat
0 upgraded, 37 newly installed, 0 to remove and 34 not upgraded.
Need to get 19.9 MB of archives.
After this operation, 163 MB of additional disk space will be used.
```


7) Paso 7: Ejecutar el comando **'mariadb -v'** para verificar si la instalación es correcta. Debe mostrarse como en la imagen, caso contrario repetir 'Paso 2' y 'Paso 7'.

```
root@42467776DB:~# mariadb -v
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 30
Server version: 10.5.19-MariaDB-0+deb11u2 Debian 11

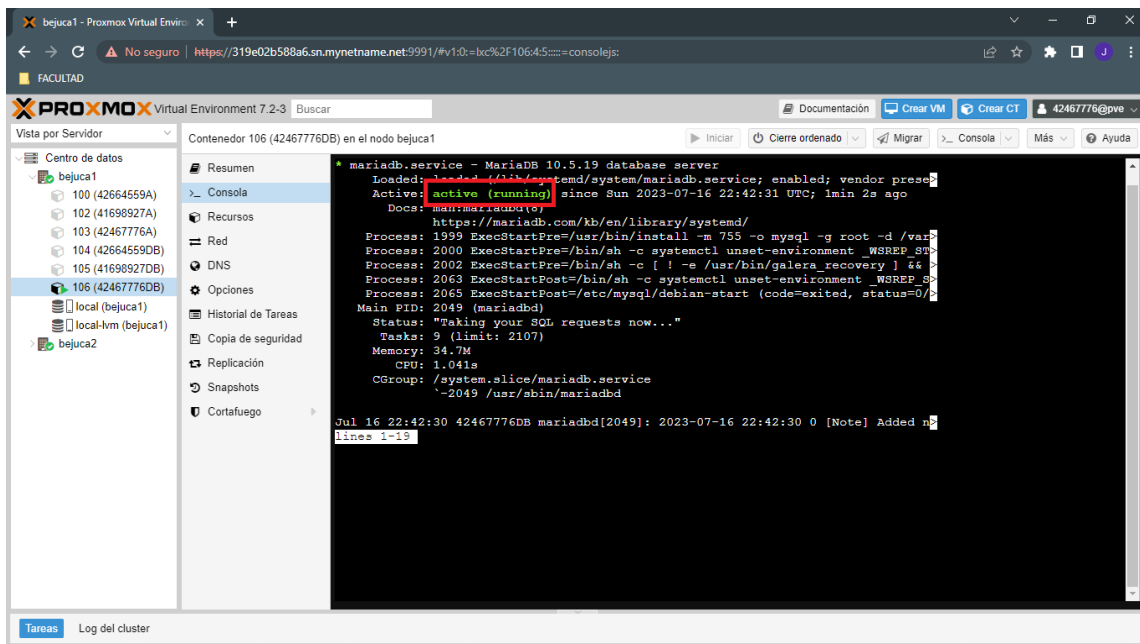
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Reading history file /root/.mysql_history
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

	<p align="center">CATEDRA DE VIRTUALIZACIÓN DEPARTAMENTO INGENIERIA EN SISTEMAS DE INFORMACION UNIVERSIDAD TECNOLÓGICA NACIONAL - FACULTAD REGIONAL TUCUMAN</p>	<p align="center">TRABAJO PRÁCTICO FINAL</p>
		<p align="center">CICLO 2023</p>

- 8) Paso 8: Ejecutar comando **'systemctl status mariadb'** para verificar si el servicio se está ejecutando. Caso contrario ejecutar **'systemctl enable mariadb'** y volver a verificar el estado de MariaDB.



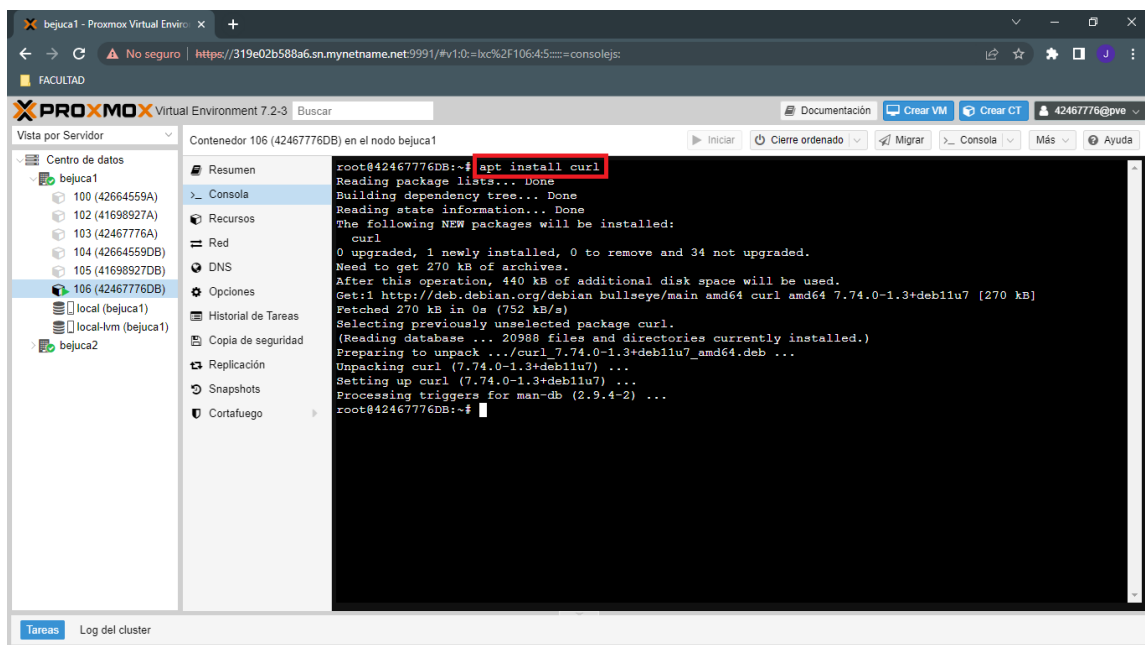
```

mariadb.service - MariaDB 10.5.19 database server
Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; vendor prese
Active: active (running) since Sun 2023-07-16 22:42:31 UTC; 1min 2s ago
Docs: man:mysqld(8)
      https://mariadb.com/kb/en/library/systemd/
Process: 1999 ExecStartPre=/usr/bin/install -m 755 -o mysql -g root -d /var
Process: 2000 ExecStartPre=/bin/sh -c systemctl unset-environment _WSREP_S
Process: 2002 ExecStartPre=/bin/sh -c [ ! -e /usr/bin/galera_recovery ] &&
Process: 2063 ExecStartPost=/bin/sh -c systemctl unset-environment _WSREP_S
Process: 2065 ExecStartPost=/etc/mysql/debian-start (code=exited, status=0/S
Main PID: 2049 (mariabdb)
Status: "Taking your SQL requests now..."
Tasks: 9 (limit: 2107)
Memory: 34.7M
CPU: 1.041s
CGroup: /system.slice/mariadb.service
        └─2049 /usr/sbin/mariabdb

Jul 16 22:42:30 42467776DB mariabdb[2049]: 2023-07-16 22:42:30 0 [Note] Added n
lines 1-13

```


- 9) Paso 9: Ejecutar el comando **'apt install curl'**, el cual nos permitirá realizar transferencias y descargas de datos/archivos a través desde una URL dada.



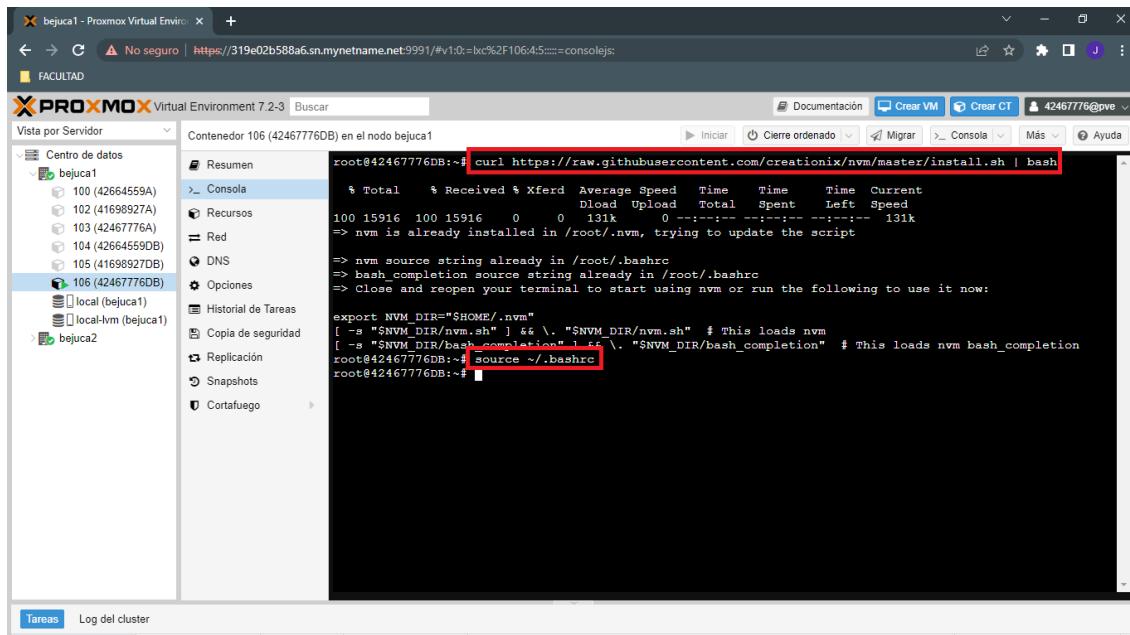
```

root@42467776DB:~# apt install curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  curl
0 upgraded, 1 newly installed, 0 to remove and 34 not upgraded.
Need to get 270 kB of archives.
After this operation, 440 kB of additional disk space will be used.
Get:1 http://deb.debian.org/debian bullseye/main amd64 curl amd64 7.74.0-1.3+deb11u7 [270 kB]
Fetched 270 kB in 0s (752 kB/s)
Selecting previously unselected package curl.
(Reading database ... 20988 files and directories currently installed.)
Preparing to unpack .../curl_7.74.0-1.3+deb11u7_amd64.deb ...
Unpacking curl (7.74.0-1.3+deb11u7) ...
Setting up curl (7.74.0-1.3+deb11u7) ...
Processing triggers for man-db (2.9.4-2) ...
root@42467776DB:~#

```

	<p align="center">CATEDRA DE VIRTUALIZACIÓN DEPARTAMENTO INGENIERIA EN SISTEMAS DE INFORMACION UNIVERSIDAD TECNOLOGICA NACIONAL - FACULTAD REGIONAL TUCUMAN</p>	TRABAJO PRÁCTICO FINAL
		CICLO 2023

- 10) **Paso 10:** Ejecutar el comando `'curl https://raw.githubusercontent.com/creationix/nvm/master/install.sh | bash'` y luego el comando `'source ~/.bashrc'`, el cual nos servirá para descargar el archivo `'install.sh'` usando `'curl'` e instalar nvm en el CT.



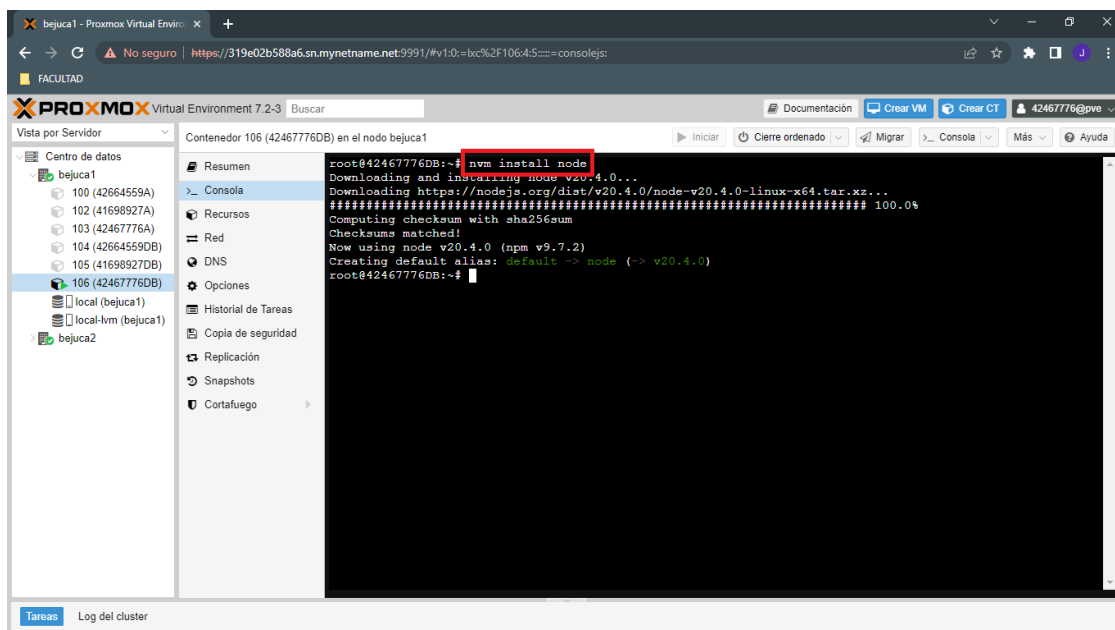
```

root@42467776DB:~# curl https://raw.githubusercontent.com/creationix/nvm/master/install.sh | bash
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
=> nvm is already installed in /root/.nvm, trying to update the script
=> nvm source string already in /root/.bashrc
=> bash_completion source string already in /root/.bashrc
=> Close and reopen your terminal to start using nvm or run the following to use it now:

export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh" # This loads nvm
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion" # This loads nvm bash_completion
root@42467776DB:~# source ~/.bashrc
root@42467776DB:~#

```


- 11) **Paso 11:** Con NVM (Node Version Manager) instalado, ejecutamos el comando `'nvm install node'`, el cual nos descargara e instalara la última versión disponible de NodeJS en el CT.



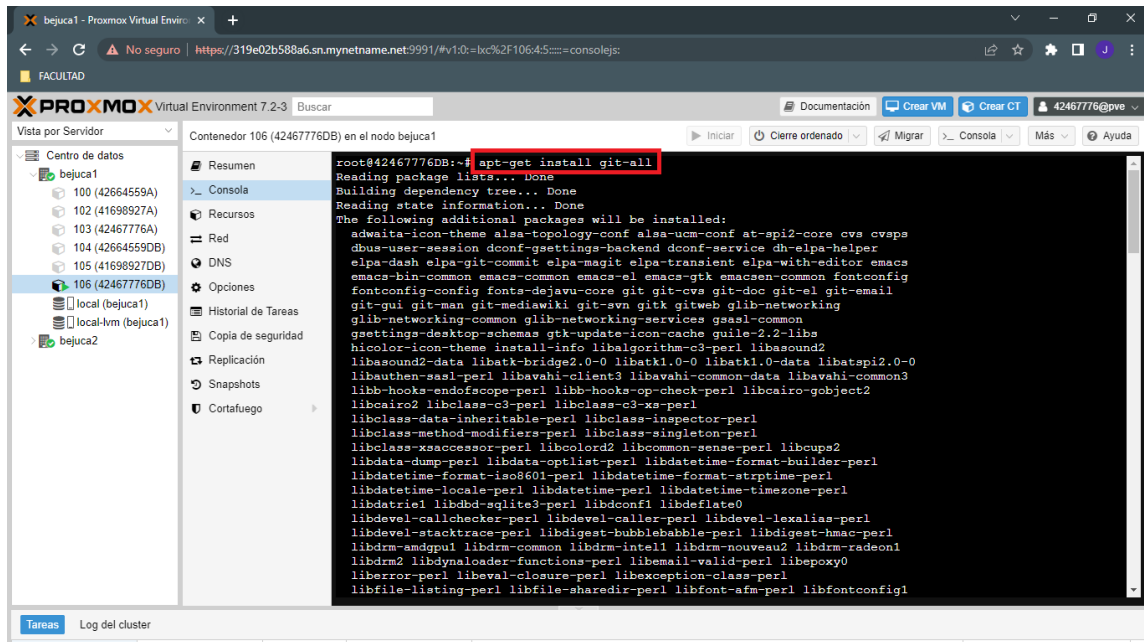
```

root@42467776DB:~# nvm install node
Downloading and installing node v20.4.0...
Downloading https://nodejs.org/dist/v20.4.0/node-v20.4.0-linux-x64.tar.xz...
##### 100.0%
Computing checksum with sha256sum
Checksums matched!
Now using node v20.4.0 (npm v9.7.2)
Creating default alias: default -> node (-> v20.4.0)
root@42467776DB:~#

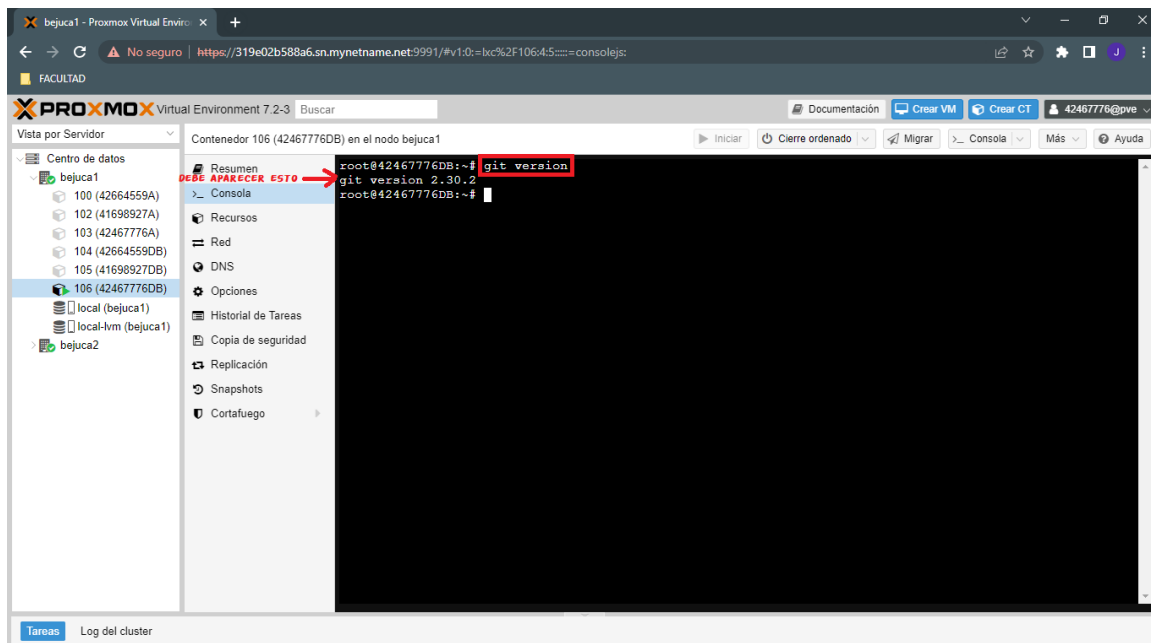
```


	CATEDRA DE VIRTUALIZACIÓN DEPARTAMENTO INGENIERIA EN SISTEMAS DE INFORMACION UNIVERSIDAD TECNOLOGICA NACIONAL - FACULTAD REGIONAL TUCUMAN		TRABAJO PRÁCTICO FINAL
			CICLO 2023

12) **Paso 12:** Ejecutar el comando **'apt-get install git-all'** para instalar Git en el CT y así poder gestionar repositorios, realizar commits, clonar repos, etc. Lo utilizaremos para los repositorios del backend y frontend.



13) **Paso 13:** Ejecutar el comando **'git version'** para verificar si Git se instaló correctamente, como muestra la imagen. Sino repetir 'Paso 2' y 'Paso 11'.



	CATEDRA DE VIRTUALIZACIÓN DEPARTAMENTO INGENIERIA EN SISTEMAS DE INFORMACION UNIVERSIDAD TECNOLÓGICA NACIONAL - FACULTAD REGIONAL TUCUMAN	TRABAJO PRÁCTICO FINAL
		CICLO 2023

FRONTEND DEL BLOG


El frontend se encarga de la presentación visual del blog persona. Su desarrollo es llevado a cabo por fuera de Proxmox, para que luego mediante GitHub ya instalado en el CT se pueda clonar el repositorio del frontend y así formar parte de este. Además, cuenta con los siguientes aspectos:

- Maquetación HTML: Se emplea HTML para crear la estructura y los elementos del blog. Organiza el contenido utilizando etiquetas, como encabezados, párrafos, listas y enlaces.
- Estilos CSS: Se emplea CSS para dar estilo al blog y mejorar su apariencia visual.
- Interactividad con JavaScript: Se utiliza JavaScript para agregar interactividad y funcionalidad dinámica al blog, como la inyección de información desde el frontend, a partir del backend dado.
- Google Firebase: Permite la funcionalidad de autenticar a un usuario en el blog mediante el uso de Google. Esto hace posible que el usuario pueda dejar comentarios en el blog, mientras que, por otro, el administrador del blog, gestionar la información del mismo.

VISTA DEL FRONTEND:

A continuación, se muestra una parte del blog:



	CATEDRA DE VIRTUALIZACIÓN DEPARTAMENTO INGENIERIA EN SISTEMAS DE INFORMACION UNIVERSIDAD TECNOLOGICA NACIONAL - FACULTAD REGIONAL TUCUMAN	TRABAJO PRÁCTICO FINAL
		CICLO 2023

BACKEND DEL BLOG:

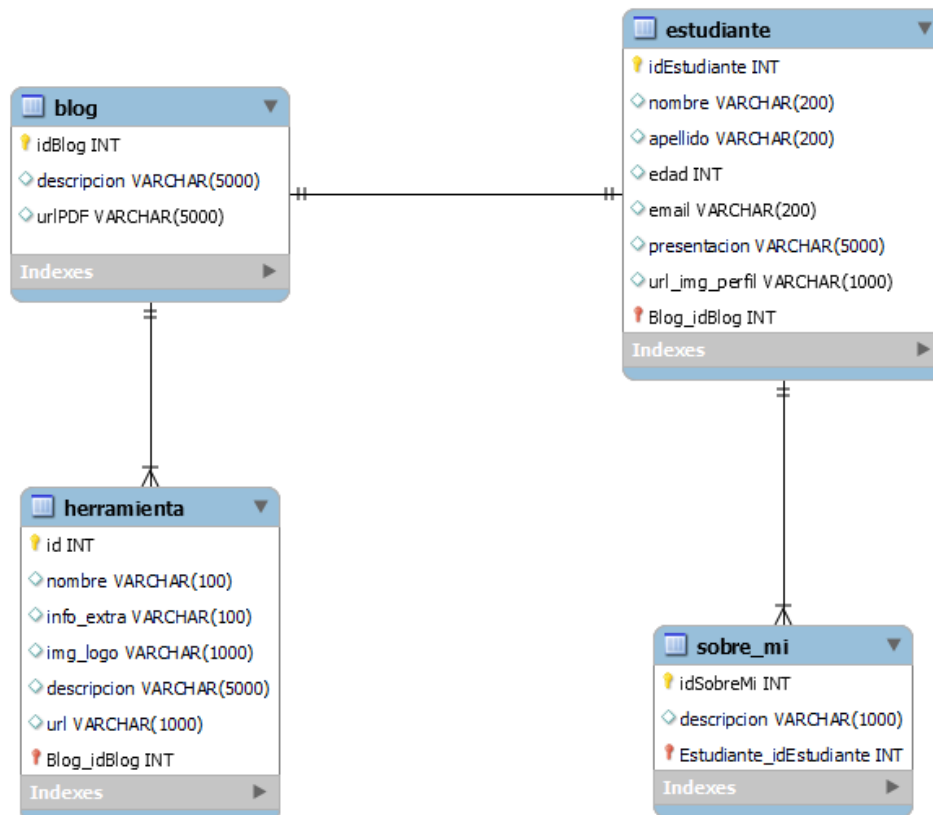
El backend del blog es el responsable de procesar las solicitudes del usuario y realizar las operaciones de almacenamiento y recuperación de datos en la base de datos.


Por un lado, se desarrolló la lógica del servidor utilizando Node.js. Se definieron las rutas y los controladores para manejar las diferentes solicitudes HTTP, como la obtención de los elementos del blog, como ser la información personal del estudiante, el envío de comentarios o la autenticación de usuarios. Además, se utilizó una base de datos MySQL / MariaDB para almacenar y recuperar los datos del blog. Se crearon las tablas y se definieron los esquemas necesarios para almacenar la información del blog. Al utilizar SQL, se pueden utilizar las consultas para realizar operaciones como la consulta de datos.

No obstante, se creó una API para permitir la comunicación entre el frontend y el backend desde el blog.

CREACION DE LA BASE DE DATOS Y TABLAS EN EL CT

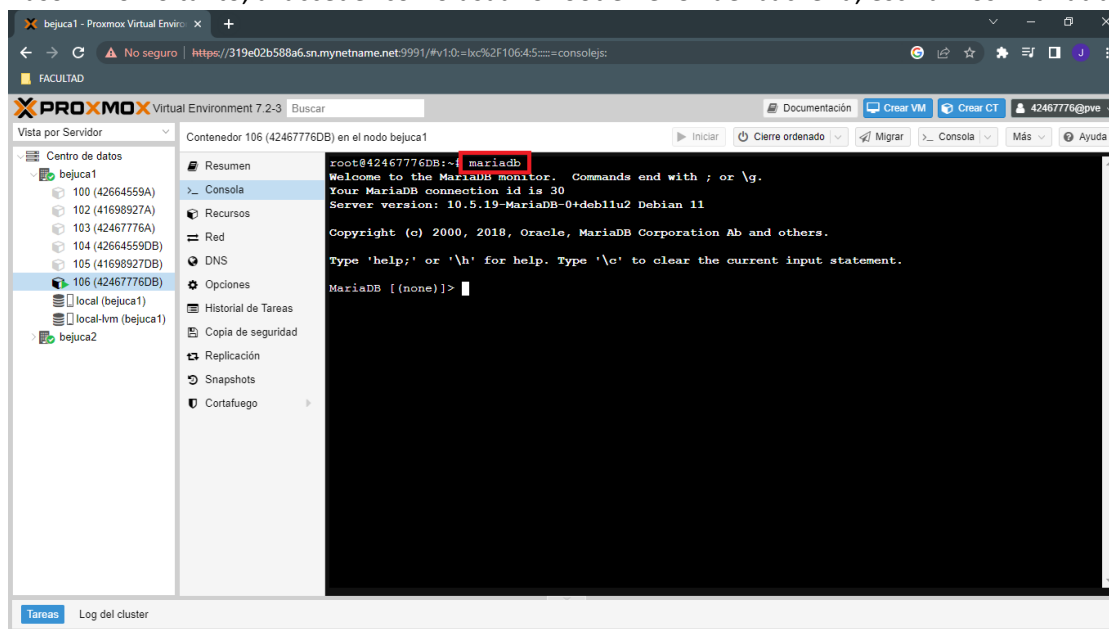
Primero se empleó MySQL Workbench para definir la estructura y tablas de la base de datos que se cargaran en el contenedor del backend. La estructura es la siguiente:



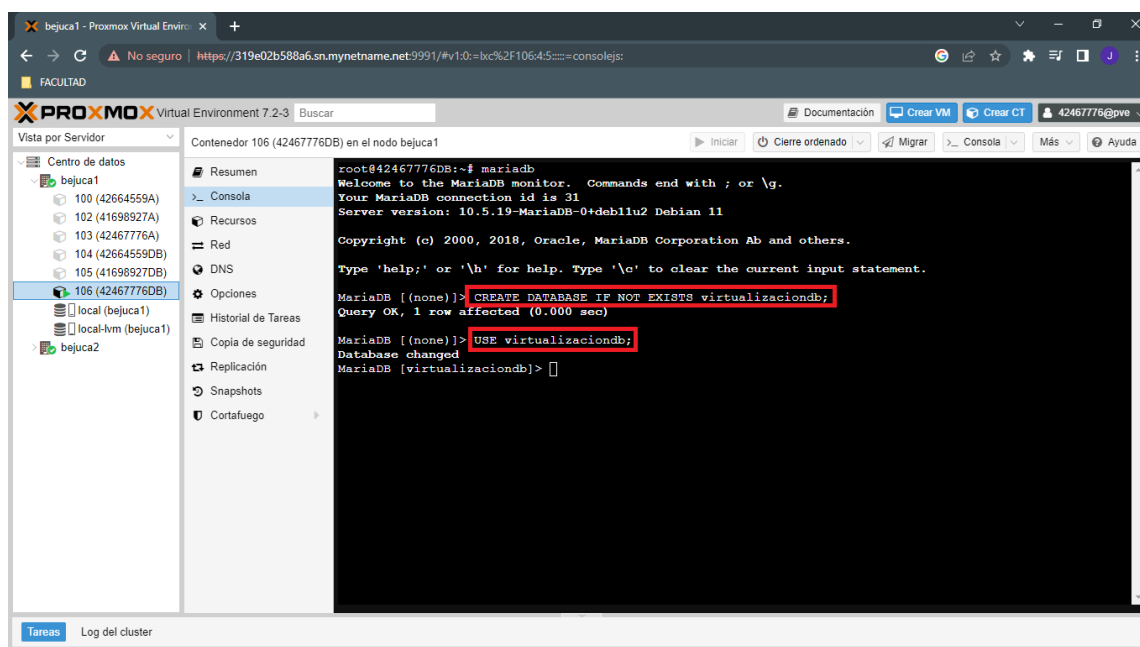
	<p align="center">CATEDRA DE VIRTUALIZACIÓN DEPARTAMENTO INGENIERIA EN SISTEMAS DE INFORMACION UNIVERSIDAD TECNOLOGICA NACIONAL - FACULTAD REGIONAL TUCUMAN</p>	<p align="center">TRABAJO PRÁCTICO FINAL</p>
		<p align="center">CICLO 2023</p>


De esta forma, definida la base de datos, se puede realizar un 'Foward' para obtener las consultas SQL que se insertaran en el contenedor del backend para implementar en el CT.

Paso 1: Por lo tanto, al acceder como usuario root en el CT del backend, escribimos 'mariadb'.



Paso 2: Se crea la base de datos con la consulta '**CREATE DATABASE IF NOT EXISTS virtualizaciondb;**' y accedemos a ella con '**USE virtualizaciondb;**'.



	<p align="center">CATEDRA DE VIRTUALIZACIÓN DEPARTAMENTO INGENIERIA EN SISTEMAS DE INFORMACION UNIVERSIDAD TECNOLOGICA NACIONAL - FACULTAD REGIONAL TUCUMAN</p>	TRABAJO PRÁCTICO FINAL
		CICLO 2023

Paso 3: Insertamos una de las tablas, que en este caso es la de 'estudiante'. Lo hacemos con '

```
CREATE TABLE IF NOT EXISTS `virtualizaciondb`.`estudiante` (  

  `idEstudiante` INT NOT NULL AUTO_INCREMENT,  

  `nombre` VARCHAR(200) NULL,  

  `apellido` VARCHAR(200) NULL,  

  `edad` INT NULL,  

  `email` VARCHAR(200) NULL,  

  `presentacion` VARCHAR(5000) NULL,  

  `url_img_perfil` VARCHAR(1000) NULL,  

  `Blog_idBlog` INT NOT NULL DEFAULT 1,  

  PRIMARY KEY (`idEstudiante`, `Blog_idBlog`),  

  INDEX `fk_Estudiante_Blog1_idx` (`Blog_idBlog` ASC) VISIBLE,  

  CONSTRAINT `fk_Estudiante_Blog1`  

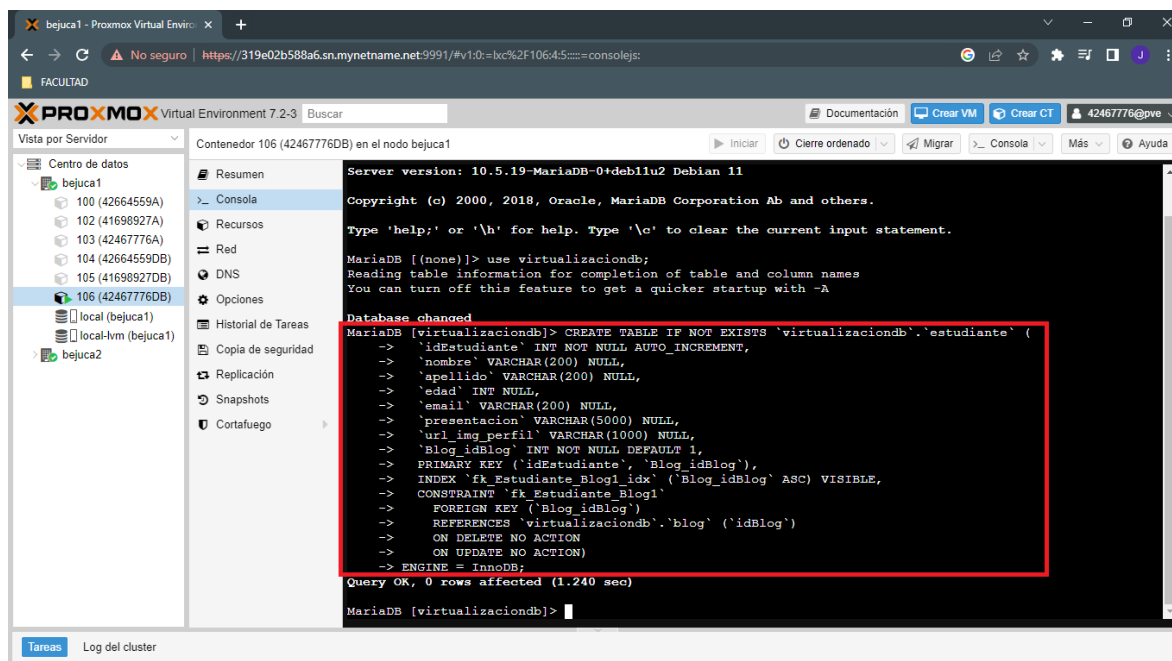
    FOREIGN KEY (`Blog_idBlog`)  

      REFERENCES `virtualizaciondb`.`blog` (`idBlog`)  

    ON DELETE NO ACTION  

    ON UPDATE NO ACTION)  

ENGINE = InnoDB;'
```



```
Server version: 10.5.19-MariaDB-0+deb11u2 Debian 11
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use virtualizaciondb;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [virtualizaciondb]> CREATE TABLE IF NOT EXISTS `virtualizaciondb`.`estudiante` (
->  `idEstudiante` INT NOT NULL AUTO_INCREMENT,
->  `nombre` VARCHAR(200) NULL,
->  `apellido` VARCHAR(200) NULL,
->  `edad` INT NULL,
->  `email` VARCHAR(200) NULL,
->  `presentacion` VARCHAR(5000) NULL,
->  `url_img_perfil` VARCHAR(1000) NULL,
->  `Blog_idBlog` INT NOT NULL DEFAULT 1,
->  PRIMARY KEY (`idEstudiante`, `Blog_idBlog`),
->  INDEX `fk_Estudiante_Blog1_idx` (`Blog_idBlog` ASC) VISIBLE,
->  CONSTRAINT `fk_Estudiante_Blog1`
->    FOREIGN KEY (`Blog_idBlog`)
->      REFERENCES `virtualizaciondb`.`blog` (`idBlog`)
->    ON DELETE NO ACTION
->    ON UPDATE NO ACTION)
-> ENGINE = InnoDB;

Query OK, 0 rows affected (1.240 sec)

MariaDB [virtualizaciondb]>
```



CATEDRA DE VIRTUALIZACIÓN
DEPARTAMENTO INGENIERIA EN SISTEMAS DE INFORMACION
UNIVERSIDAD TECNOLOGICA NACIONAL - FACULTAD REGIONAL TUCUMAN

TRABAJO
PRÁCTICO FINAL

CICLO
2023

Paso 4: Verificamos si la tabla estudiante se cargó en la DB, ingresando **'SHOW TABLES;'**

```
root@42467776DB:~# mariadb
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 35
Server version: 10.5.19-MariaDB-0+deb11u2 Debian 11

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use virtualizaciondb;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [virtualizaciondb]> show tables;
+-----+
| Tables_in_virtualizaciondb |
+-----+
| blog                        |
| estudiante                  |
+-----+
2 rows in set (0.000 sec)

MariaDB [virtualizaciondb]>
```

Paso 5: Insertamos un elemento en la tabla **'estudiante'**. Por un lado, podemos ver la estructura de la tabla con **'DESCRIBE estudiante'** para guiarnos en la carga de datos, e insertarlos correctamente. También usar **'SELECT * FROM estudiante'** para verificar los datos insertados previamente.

```
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use virtualizaciondb;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [virtualizaciondb]> INSERT INTO estudiante VALUES (1, 'Kevin Jons', 'Busto', 23, 'Kevin.Busto@al
.frit.utn.edu.ar', 'Tengo 23 a s. Estudio Ing. Sistemas...', 'https://drive.google.com/uc?export=views&id=
Zemc9V23wiT_tadoMqbpq08BoUX7vKRO', 1);
Query OK, 1 row affected (0.022 sec)

MariaDB [virtualizaciondb]> describe estudiante;
+-----+
| Field          | Type          | Null | Key | Default | Extra          |
+-----+
| idEstudiante   | int(11)       | NO   | PRI | NULL    | auto_increment |
| nombre         | varchar(200)  | YES  |     | NULL    |                |
| apellido       | varchar(200)  | YES  |     | NULL    |                |
| edad           | int(11)       | YES  |     | NULL    |                |
| email          | varchar(200)  | YES  |     | NULL    |                |
| presentacion   | varchar(5000) | YES  |     | NULL    |                |
| url_img_perfil | varchar(1000) | YES  |     | NULL    |                |
| Blog_idBlog    | int(11)       | NO   |     | 1       |                |
+-----+
8 rows in set (0.053 sec)

MariaDB [virtualizaciondb]>
```



CATEDRA DE VIRTUALIZACIÓN
DEPARTAMENTO INGENIERIA EN SISTEMAS DE INFORMACION
UNIVERSIDAD TECNOLOGICA NACIONAL - FACULTAD REGIONAL TUCUMAN

TRABAJO
PRÁCTICO FINAL

CICLO
2023

```
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use virtualizaciondb;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [virtualizaciondb]> select * from estudiante;
+-----+-----+-----+-----+-----+-----+-----+-----+
| idEstudiante | nombre | apellido | edad | email | presentacion | url_img_perfil | Blog_id |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | Kevin | Busto | 23 | Kevin.Busto@alu.frt.utn.edu.ar | Tengo 23 aos. Estudio Ing. Sistemas... | url_image.com | 1 |
+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.000 sec)

MariaDB [virtualizaciondb]>
```

Y finalmente, luego de cargar las tablas, al realizar la consulta **‘SHOW TABLES’** sobre la base de datos **‘virtualizaciondb’**, deberán aparecer las mismas que el diagrama relacional.

```
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 38
Server version: 10.5.19-MariaDB-0+deb11u2 Debian 11


Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use virtualizaciondb;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [virtualizaciondb]> show tables;
+-----+
| Tables_in_virtualizaciondb |
+-----+
| blog |
| estudiante |
| herramienta |
| sobre_mi |
+-----+
4 rows in set (0.000 sec)

MariaDB [virtualizaciondb]>
```

	CATEDRA DE VIRTUALIZACIÓN DEPARTAMENTO INGENIERIA EN SISTEMAS DE INFORMACION UNIVERSIDAD TECNOLOGICA NACIONAL - FACULTAD REGIONAL TUCUMAN	TRABAJO PRÁCTICO FINAL
		CICLO 2023

DEPENDENCIAS DEL BACKEND:

Las dependencias del backend del blog son los módulos, bibliotecas y paquetes de software adicionales que se utilizan para construir y ejecutar el backend de la aplicación. Estas dependencias proporcionan funcionalidades específicas y facilitan el desarrollo del servidor web y la interacción con la base de datos. Se instalan en la carpeta del backend empleando la consola o terminal. Las utilizadas para el blog son:

- Express.js: Express.js es un framework de aplicación web para Node.js que simplifica el manejo de rutas, middleware y solicitudes HTTP. Se instala con el comando ***'npm install express'***.
- MariaDB (o MySQL): MariaDB es un sistema de gestión de bases de datos relacional compatible con MySQL. Utiliza un controlador o librería específica para Node.js para interactuar con la base de datos y realizar operaciones como consultas, inserciones, actualizaciones y eliminaciones de datos. Se instala con el comando ***'npm install --save mysql12'***.
- Morgan: Morgan es un middleware de registro de solicitudes HTTP para Express.js. Proporciona información detallada sobre las solicitudes entrantes, como las URL. Se instala con el comando ***'npm install morgan'***.
- CORS (Cross-Origin Resource Sharing): CORS es una dependencia utilizada para habilitar el intercambio de recursos entre diferentes dominios en una aplicación web. Se instala con el comando ***'npm install cors'***.
- Nodemon: Nodemon es una herramienta de desarrollo que ayuda a reiniciar automáticamente la aplicación Node.js cuando se detectan cambios en los archivos. Se instala con el comando ***'npm install --save-dev nodemon'***.


GITHUB Y CTs DE PROXMOX

Para incorporar tanto el backend como frontend en los contenedores correspondientes de Proxmox, la práctica más fácil es subirlos a GitHub en repositorios separados, y luego, en los contenedores aplicar los comandos de Git para clonar dichos repositorios.

SUBIR BACKEND Y FRONTEND A GITHUB

Por lo tanto, los pasos para subir el backend, los cuales también aplican exactamente para el frontend, son los siguientes:

1. Crear un repositorio vacío en Github para backend.
2. Copiar la URL del repositorio remoto de GitHub creado.

	CATEDRA DE VIRTUALIZACIÓN DEPARTAMENTO INGENIERIA EN SISTEMAS DE INFORMACION UNIVERSIDAD TECNOLÓGICA NACIONAL - FACULTAD REGIONAL TUCUMAN	TRABAJO PRÁCTICO FINAL
		CICLO 2023

3. Localmente crear una carpeta y acceder a ella. Luego, abrir la consola de comandos y ejecutar el comando **'git init'**. Luego renombrar la rama 'master' a 'main' con el comando **'git branch -m main'**.
4. Agregar el repositorio remoto de GitHub como origen al repositorio local, es decir a la carpeta creada. Para ello ejecutar **'git remote add origin URL_REPOSITORIO'**.
5. Luego, para cargar el contenido del repositorio local al remoto, ejecutar la siguiente secuencia de comandos. El primero, **'git add .'**, luego **'git commit -m "primer commit"'**, y por ultimo **'git push origin main'** o **'git push -u origin main'**.

De esta forma, estará subido el backend a un repositorio de GitHub. Repetir los pasos para el frontend.

CLONAR REPOSITORIOS EN LOS CT DE PROXMOX

Para clonar un repositorio en un contenedor, primero se debe nuevamente acceder como usuario 'root'. Luego, debido a la instalación previa de Apache2 en el CT, este nos generara una nueva ubicación en el sistema, la cual nos sirve para almacenar los archivos del sitio web. Tal ruta se accede mediante el comando **'cd /var/www/html'**. Al verificar los datos dentro de dicha ruta mediante el comando **'ls'**, veremos un **'index.html'**. Tal archivo es el predeterminado que nos brinda Apache2 luego de su instalación, y para acceder al mismo basta con escribir el siguiente comando **'ip a'**, para así obtener la IP del CT y pegarla como URL para ver la página como tal. Además, el archivo **'index.html'** puede ser modificado, dentro de la ruta establecida, mediante el comando **'nano index.html'**.

En este caso, como el backend seguramente contendrá mucha información, al igual que el frontend, sin limitarse a un único **'index.html'**, es por eso que debemos clonar los repositorios y agilizar la carga de los archivos al CT usando GitHub.

Por lo tanto, la ruta que nos interesa, y que únicamente leerá Apache, es **'/html'**. Entonces, nuevamente accedemos a la ruta mediante **'cd /var/www'** o, si estas en la ruta **'/html'**, ejecutar el comando **'cd ..'** para volver atrás. De esta forma, eliminaremos la carpeta **'html'** para darle lugar a la carpeta clonada y así renombrarla a **'html'** para que Apache la reconozca sin problemas. Por ello, ejecutar **'rm -r html'**.

La siguiente imagen describe lo previamente mencionado, y en rojo se marcan los comandos importantes. Los comandos restantes son una muestra para navegar y analizar el contenido en las carpetas.



CATEDRA DE VIRTUALIZACIÓN
DEPARTAMENTO INGENIERIA EN SISTEMAS DE INFORMACION
UNIVERSIDAD TECNOLOGICA NACIONAL - FACULTAD REGIONAL TUCUMAN

TRABAJO
PRÁCTICO FINAL


CICLO
2023

```
root@42467776DB:~# cd /var/www
root@42467776DB:/var/www# ls
html
root@42467776DB:/var/www# cd html
root@42467776DB:/var/www/html# ls
index.html
root@42467776DB:/var/www/html# cd ..
root@42467776DB:/var/www# rm -r html
root@42467776DB:/var/www# ls
```

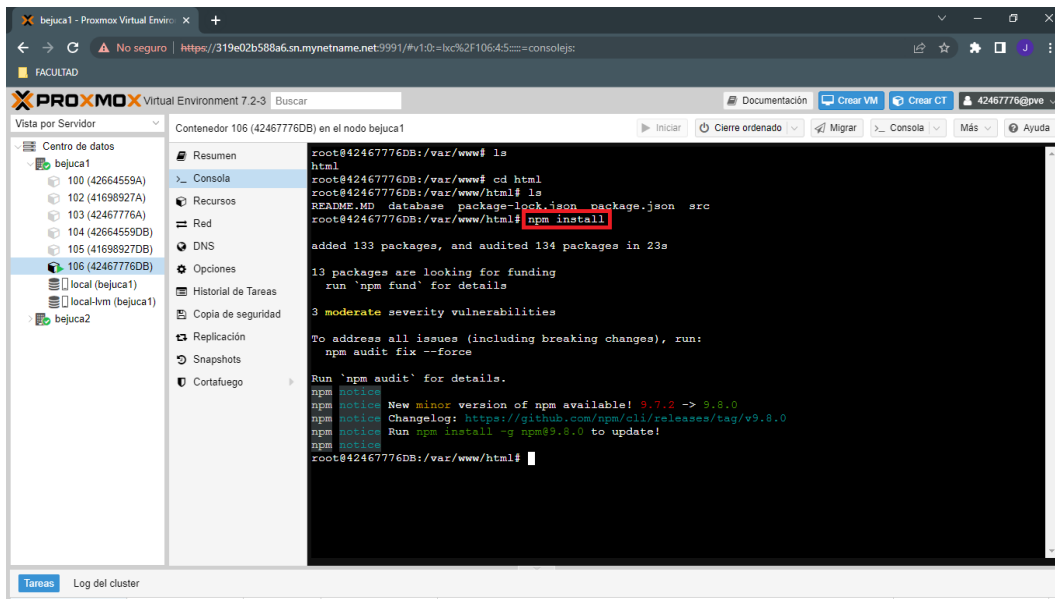
Luego de eliminar la carpeta 'html', y permaneciendo en la ruta '/var/www', clonar el repositorio de GitHub con el comando '**git clone URL_REPOSITORIO**'. GitHub pedirá antes iniciar sesión con la cuenta dentro del CT. En este caso, se clonará el repositorio del backend. Es lo mismo para el frontend, con una leve diferencia.

Luego de clonar el repositorio de GitHub, cuya carpeta tenga por nombre '**NOMBRE_REPOSITORIO**', renómbrela a 'html' para que la reconozca Apache2, mediante el comando '**mv NOMBRE_REPOSITORIO html**'.

```
root@42467776DB:/var/www# git clone https://github.com/JonasBusto/virtualizacion-back
Cloning into 'virtualizacion-back'...
remote: Enumerating objects: 67, done.
remote: Counting objects: 100% (67/67), done.
remote: Compressing objects: 100% (48/48), done.
remote: Total 67 (delta 25), reused 57 (delta 15), pack-reused 0
Receiving objects: 100% (67/67), 47.95 KiB | 1.84 MiB/s, done.
Resolving deltas: 100% (25/25), done.
root@42467776DB:/var/www# ls
virtualizacion-back
root@42467776DB:/var/www# mv virtualizacion-back html
root@42467776DB:/var/www# ls
html
```


	<p align="center">CATEDRA DE VIRTUALIZACIÓN DEPARTAMENTO INGENIERIA EN SISTEMAS DE INFORMACION UNIVERSIDAD TECNOLOGICA NACIONAL - FACULTAD REGIONAL TUCUMAN</p>	<p align="center">TRABAJO PRÁCTICO FINAL</p>
		<p align="center">CICLO 2023</p>

Luego acceder a la carpeta clonada y renombra mediante el comando **'cd html'**. Dentro de ella, ejecutamos el comando **'npm install'** para instalar las dependencias. Esta es la leve diferencia mencionada anteriormente respecto al frontend, ya que el mismo fue desarrollado con Vanilla JS y no fue necesario dependencias.



```

root@42467776DB:/var/www# ls
html
root@42467776DB:/var/www# cd html
root@42467776DB:/var/www/html# ls
README.md database package-lock.json package.json src
root@42467776DB:/var/www/html# npm install

added 133 packages, and audited 134 packages in 23s

13 packages are looking for funding
  run 'npm fund' for details

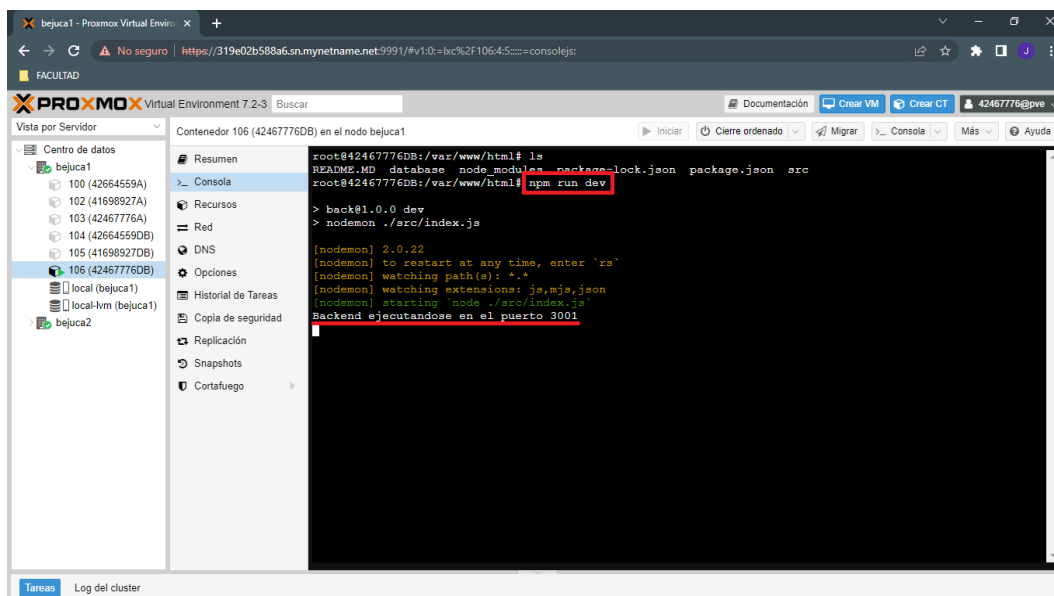
3 moderate severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run 'npm audit' for details.
npm notice New minor version of npm available! 9.7.2 -> 9.8.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v9.8.0
npm notice Run npm install -g npm@9.8.0 to update!
npm notice
root@42467776DB:/var/www/html#

```

De esta forma, el backend estará listo. Luego arranco el mismo con el comando **'npm run dev'** que definí al momento de la codificación. Y para acceder al mismo y generar peticiones, la URL es la IP del CT, que se obtiene mediante el comando **'ip a'**, y el puerto empleado, que en este caso es el 3001. Por lo tanto, la URL se compone de **'IP:3001/'**.




```

root@42467776DB:/var/www/html# ls
README.md database node_modules package-lock.json package.json src
root@42467776DB:/var/www/html# npm run dev

> back@1.0.0 dev
> nodemon ./src/index.js

[nodemon] 2.0.22
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting node ./src/index.js
Backend ejecutandose en el puerto 3001

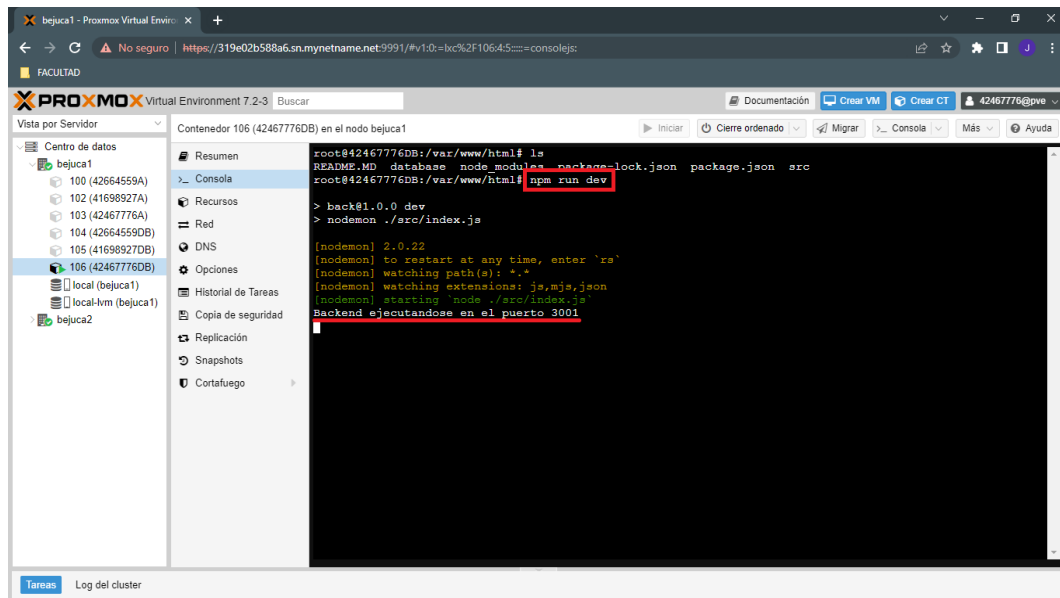
```

	<p align="center">CATEDRA DE VIRTUALIZACIÓN DEPARTAMENTO INGENIERIA EN SISTEMAS DE INFORMACION UNIVERSIDAD TECNOLÓGICA NACIONAL - FACULTAD REGIONAL TUCUMAN</p>	TRABAJO PRÁCTICO FINAL
		CICLO 2023

CONEXIÓN ENTRE CONTENEDORES:

Una vez que cada contenedor tenga almacenado el código del backend y frontend, y creada la base de datos mediante MariaDB en el contenedor del backend, verificamos la conexión entre ellos realizando los siguientes pasos:

- 1- Iniciar ambos contenedores y autenticarse en los mismos.
- 2- En el contenedor del backend, acceder a la carpeta donde tengamos cargado el repositorio del backend, en mi caso ejecute **'cd /var/www/html'**, y una vez allí arrancar el backend, en mi caso mediante el comando **'npm run dev'**.



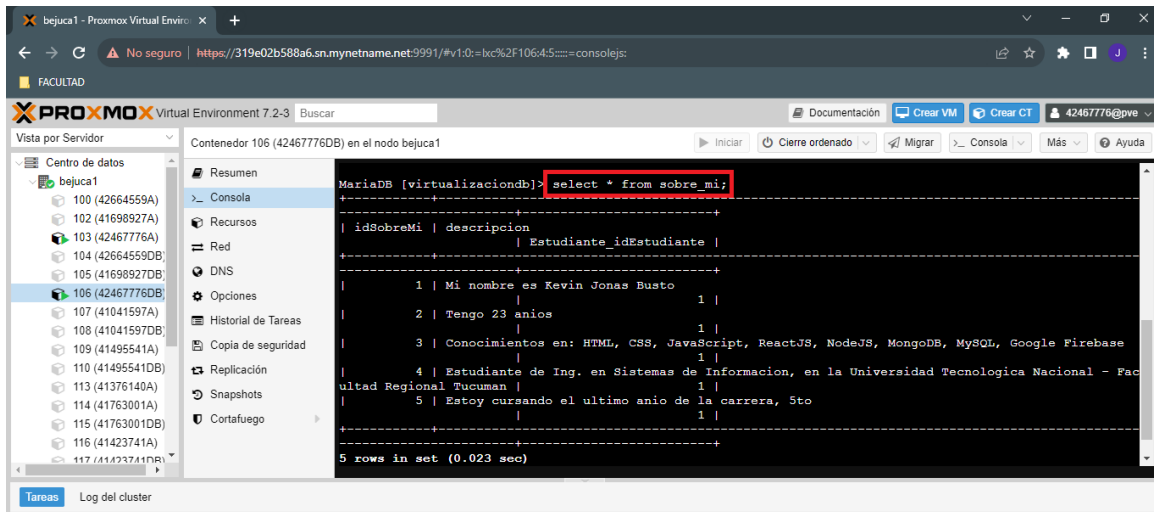
- 3- Desde el contenedor del frontend, ejecutar el comando **'curl RUTA-BACKEND'**. En este caso, la ruta del backend del contenedor es la **'IP:Puerto/rutaPersonalizada'** del mismo. En mi caso, por ejemplo, ejecute la siguiente petición **'curl 192.168.77.228:3001/sobreMi'**. La información referida a la tabla **'SobreMi'** se puede leer accediendo a MariaDB desde el CT del backend, como aparece a continuación:



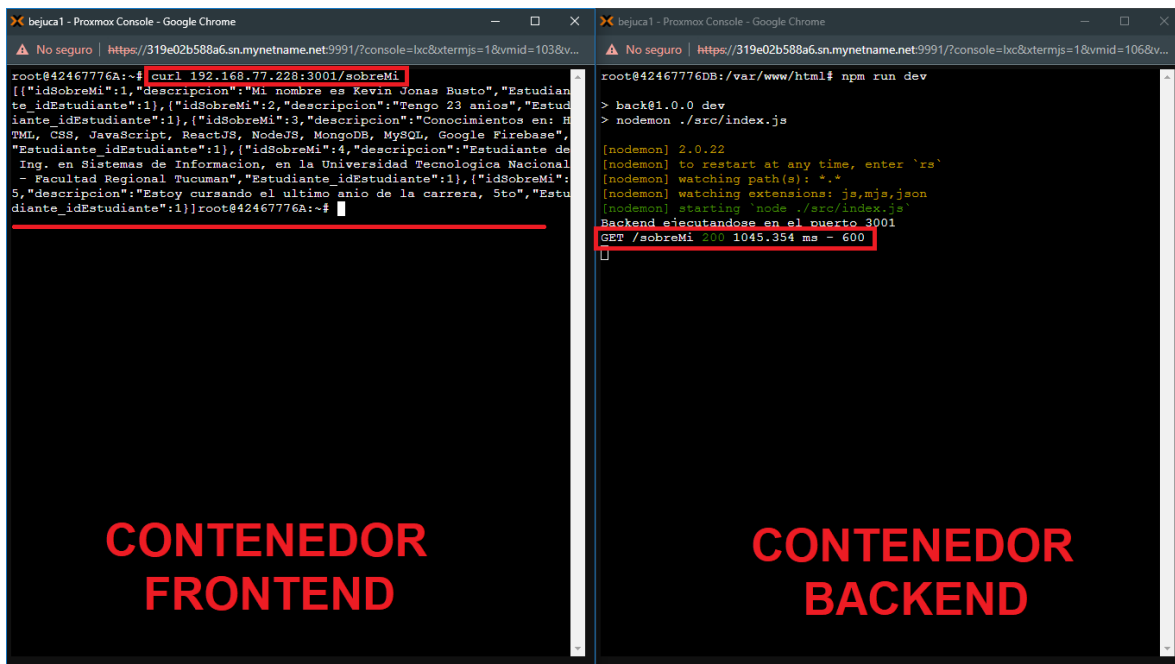
CATEDRA DE VIRTUALIZACIÓN
DEPARTAMENTO INGENIERIA EN SISTEMAS DE INFORMACION
UNIVERSIDAD TECNOLOGICA NACIONAL - FACULTAD REGIONAL TUCUMAN

TRABAJO
PRÁCTICO FINAL


CICLO
2023



Una vez comprobado la existencia de tabla 'SobreMi' con sus respectivos datos, ejecuto el comando '`curl 192.168.77.228:3001/sobreMi`' desde el CT Frontend:



Desde el CT Frontend realizo la petición al CT Backend. A la izquierda se ve la petición, acompañada del resultado de la misma, y a la derecha la información sobre la petición que fue realizada exitosamente mediante un mensaje 'GET 200' como muestra la imagen, hacia la ruta correspondiente. De esta forma, se verifica la conexión entre contenedores.

	<p align="center">CATEDRA DE VIRTUALIZACIÓN DEPARTAMENTO INGENIERIA EN SISTEMAS DE INFORMACION UNIVERSIDAD TECNOLÓGICA NACIONAL - FACULTAD REGIONAL TUCUMAN</p>	TRABAJO PRÁCTICO FINAL
		CICLO 2023

4- Luego de la redirección de puerto del contenedor del Frontend, y este estando encendido, se puede acceder de forma externa al contenedor, a la página web. La URL de mi página web es: <http://319e02b588a6.sn.mynetname.net:8002/>

5- Proyecto finalizado:

