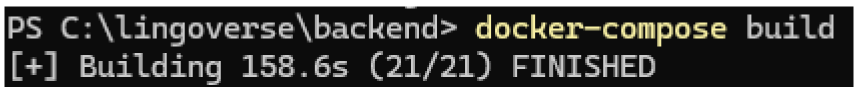
DESPLIEGUE DE LA APLICACIÓN

Una vez tenemos el proyecto descargado de GitHub, tenemos que abrir la terminal y nos colocamos en la carpeta backend, es ahí donde tenemos el composer y todos los archivos del Docker.

1. Construimos los contenedores:

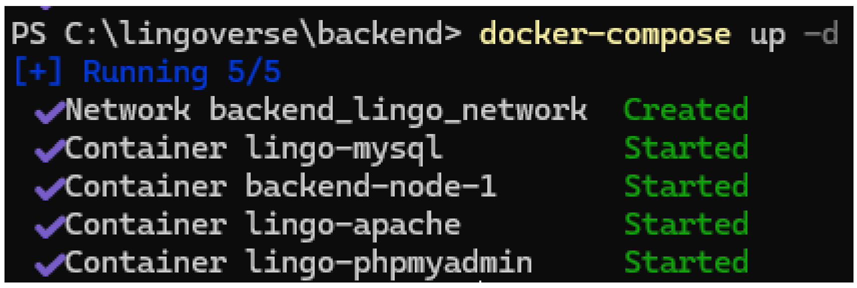
docker-compose build



```
PS C:\lingoverse\backend> docker-compose build
[+] Building 158.6s (21/21) FINISHED
```

2. Levantamos los contenedores:

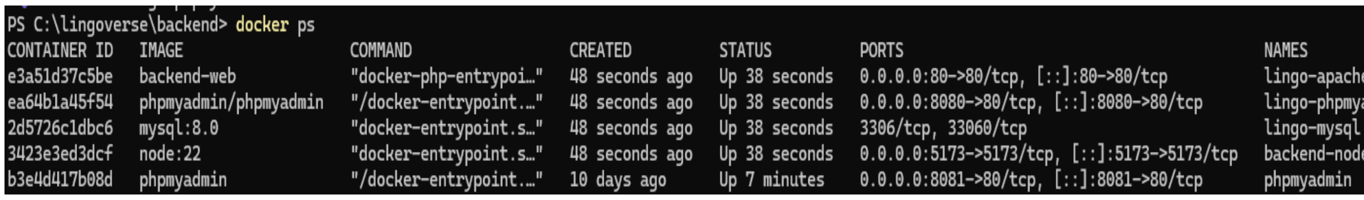
docker-compose up -d



```
PS C:\lingoverse\backend> docker-compose up -d
[+] Running 5/5
 ✔Network backend_lingo_network  Created
 ✔Container lingo-mysql          Started
 ✔Container backend-node-1       Started
 ✔Container lingo-apache         Started
 ✔Container lingo-phpmyadmin     Started
```

3. Comprobamos que estén en marcha:

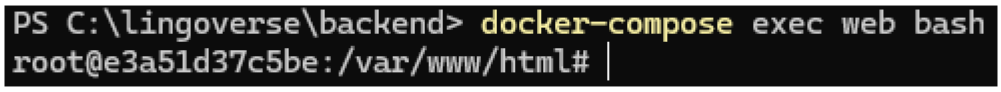
docker ps



```
PS C:\lingoverse\backend> docker ps
CONTAINER ID   IMAGE                   COMMAND                  CREATED          STATUS          PORTS                                         NAMES
e3a51d37c5be   backend-web             "docker-php-entrypoi…"   48 seconds ago   Up 38 seconds   0.0.0.0:80->80/tcp, [::]:80->80/tcp           lingo-apache
ea64b1a45f54   phpmyadmin/phpmyadmin   "/docker-entrypoint.…"   48 seconds ago   Up 38 seconds   0.0.0.0:8080->80/tcp, [::]:8080->80/tcp       lingo-phpmy
2d5726c1dbc6   mysql:8.0               "docker-entrypoint.s…"   48 seconds ago   Up 38 seconds   3306/tcp, 33060/tcp                           lingo-mysql
3423e3ed3dcf   node:22                 "docker-entrypoint.s…"   48 seconds ago   Up 38 seconds   0.0.0.0:5173->5173/tcp, [::]:5173->5173/tcp   backend-node
b3e4d417b08d   phpmyadmin              "/docker-entrypoint.…"   10 days ago      Up 7 minutes    0.0.0.0:8081->80/tcp, [::]:8081->80/tcp       phpmyadmin
```

4. Entramos en el contenedor web

docker-compose exec web bash

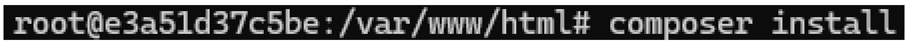


```
PS C:\lingoverse\backend> docker-compose exec web bash
root@e3a51d37c5be:/var/www/html#
```

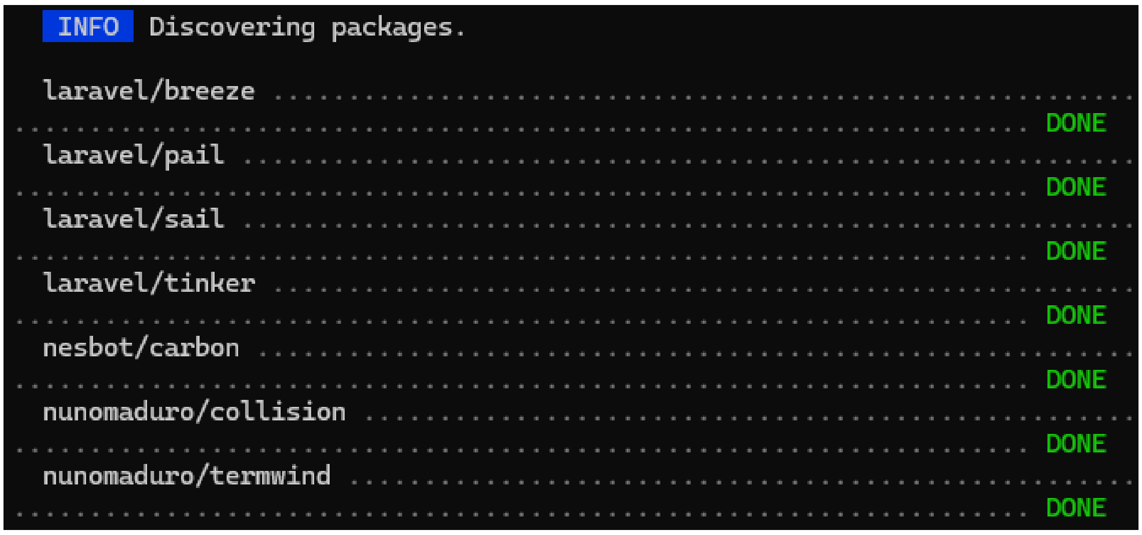
Una vez dentro, ejecutamos las siguientes líneas:

5. Instalamos las dependencias de laravel:

composer install



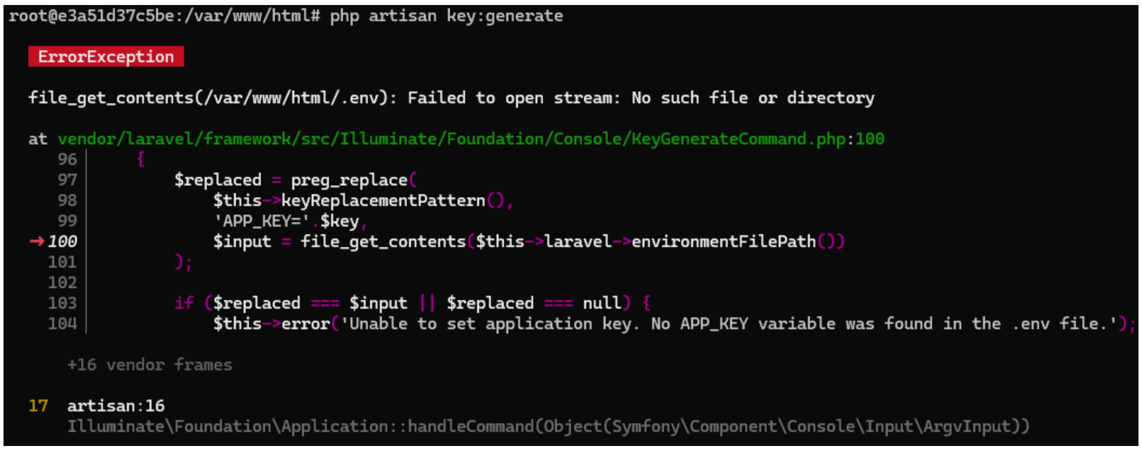
```
root@e3a51d37c5be:/var/www/html# composer install
```



```
  INFO  Discovering packages.

  laravel/breeze ..........................................................
 ...................................................................... DONE
  laravel/pail ............................................................
 ...................................................................... DONE
  laravel/sail ............................................................
 ...................................................................... DONE
  laravel/tinker ..........................................................
 ...................................................................... DONE
  nesbot/carbon ...........................................................
 ...................................................................... DONE
  nunomaduro/collision ....................................................
 ...................................................................... DONE
  nunomaduro/termwind .....................................................
 ...................................................................... DONE
```

6. Creamos la clave de la aplicación:

php artisan key:generate



```
root@e3a51d37c5be:/var/www/html# php artisan key:generate

  ErrorException

  file_get_contents(/var/www/html/.env): Failed to open stream: No such file or directory

  at vendor/laravel/framework/src/Illuminate/Foundation/Console/KeyGenerateCommand.php:100
     96▕     {
     97▕         $replaced = preg_replace(
     98▕             $this->keyReplacementPattern(),
     99▕             'APP_KEY='.$key,
  ➜100▕             $input = file_get_contents($this->laravel->environmentFilePath())
    101▕         );
    102▕
    103▕         if ($replaced === $input || $replaced === null) {
    104▕             $this->error('Unable to set application key. No APP_KEY variable was found in the .env file.');

      +16 vendor frames

  17  artisan:16
      Illuminate\Foundation\Application::handleCommand(Object(Symfony\Component\Console\Input\ArgvInput))
```

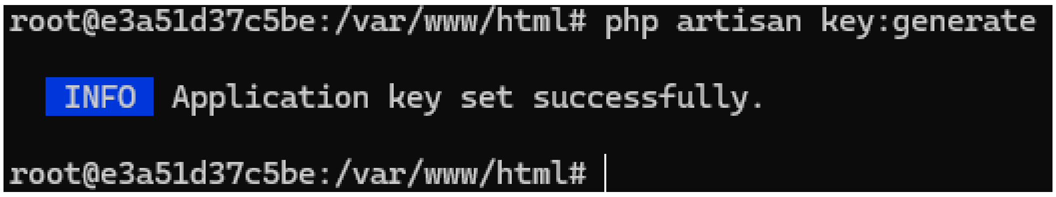
7. Esto nos dará un error, ya que cuando subimos nuestro proyecto a GitHub no se sube el .env, por eso no lo encuentra. Por eso, crearemos el .env en la carpeta src.

```
APP_NAME=Lingo
APP_ENV=local
```

```
APP_KEY=
APP_DEBUG=true
APP_URL=http://localhost

# ======================
# BASE DE DATOS
# ======================
DB_CONNECTION=mysql
DB_HOST=db
DB_PORT=3306
DB_DATABASE=lingo_db
DB_USERNAME=markel
DB_PASSWORD=daw3

# ======================
# CONFIGURACIONES VARIAS
# ======================
LOG_CHANNEL=stack
LOG_LEVEL=debug

BROADCAST_DRIVER=log
CACHE_DRIVER=file
FILESYSTEM_DISK=local
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

# ======================
# FRONTEND (Vite)
# ======================
VITE_APP_URL=http://localhost:5173
```

8. Volvemos a generar la clave:

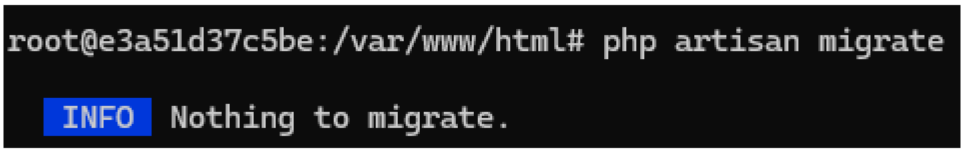
php artisan key:generate



```
root@e3a51d37c5be:/var/www/html# php artisan key:generate

  INFO  Application key set successfully.

root@e3a51d37c5be:/var/www/html#
```

9. Aplicamos las migraciones de base de datos.

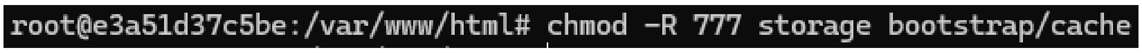
php artisan migrate



```
root@e3a51d37c5be:/var/www/html# php artisan migrate

  INFO  Nothing to migrate.
```

10. Damos los permisos a las carpetas:

chmod -R 777 storage bootstrap/cache



```
root@e3a51d37c5be:/var/www/html# chmod -R 777 storage bootstrap/cache
```

DNS

1. Dentro de la carpeta apache.conf, añadimos las lineas:

ServerName localhost
ServerAlias lingo.local

Debería quedar así:

```
<VirtualHost *:80>
    # La carpeta 'public' de Laravel es la raíz de la aplicación
    DocumentRoot /var/www/html/public
    ServerName lingo.local
    ServerAlias www.lingo.local

    <Directory /var/www/html/public>
        AllowOverride All
        Require all granted
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

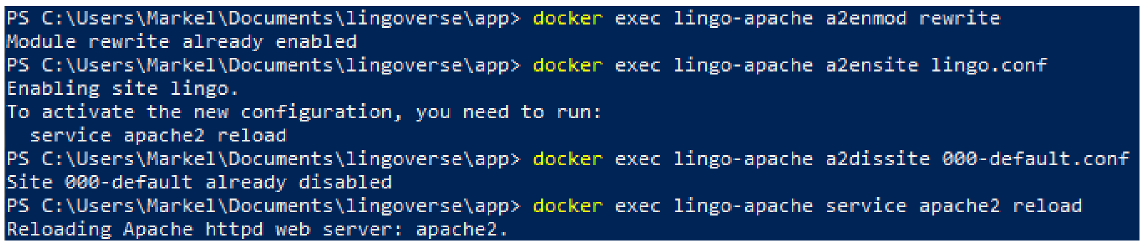
2. Activamos el módulo y sitio

En la consola:

a2enmod rewrite

a2ensite lingo.conf

a2dissite 000-default.conf



```
PS C:\Users\Markel\Documents\lingoverse\app> docker exec lingo-apache a2enmod rewrite
Module rewrite already enabled
PS C:\Users\Markel\Documents\lingoverse\app> docker exec lingo-apache a2ensite lingo.conf
Enabling site lingo.
To activate the new configuration, you need to run:
  service apache2 reload
PS C:\Users\Markel\Documents\lingoverse\app> docker exec lingo-apache a2dissite 000-default.conf
Site 000-default already disabled
PS C:\Users\Markel\Documents\lingoverse\app> docker exec lingo-apache service apache2 reload
Reloading Apache httpd web server: apache2.
```

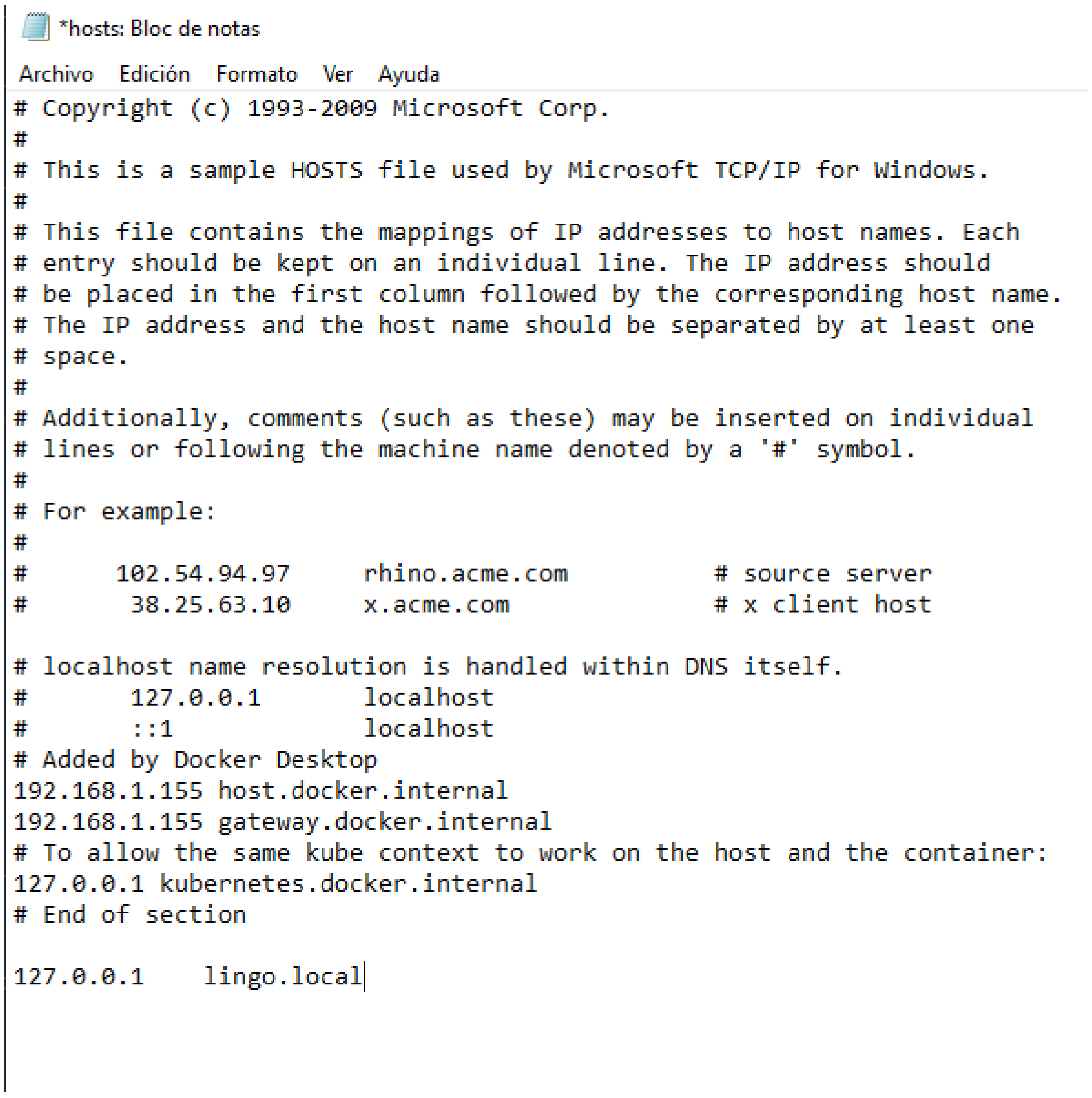
Reiniciamos apache: service apache2 reload

3. Abrimos bloc de notas como administrador.
4. Abrimos:

C:\Windows\System32\drivers\etc\hosts

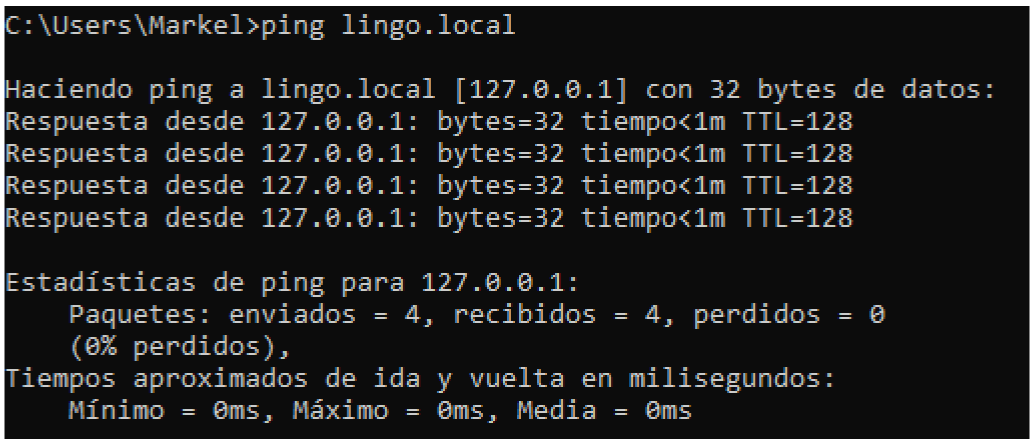
5. Añadimos al final:

127.0.0.1   lingo.local



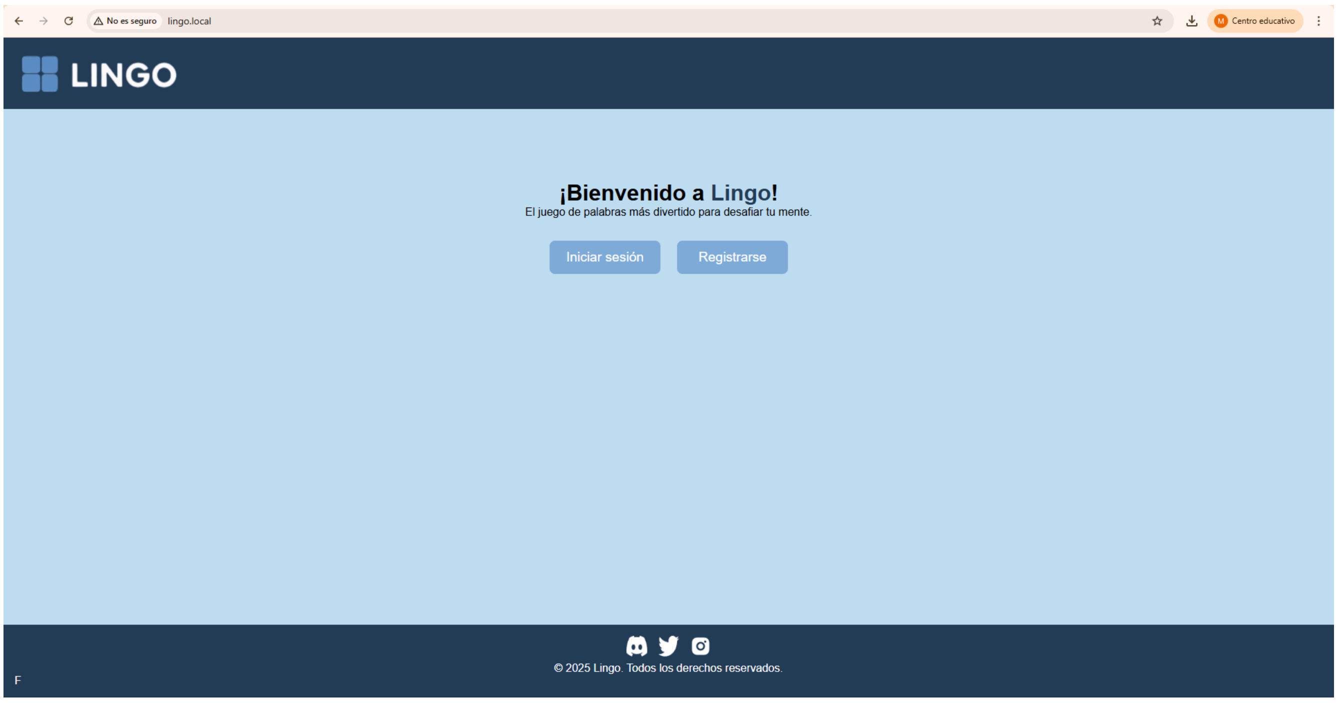
```
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      102.54.94.97     rhino.acme.com          # source server
#       38.25.63.10     x.acme.com              # x client host

# localhost name resolution is handled within DNS itself.
#       127.0.0.1       localhost
#       ::1             localhost
# Added by Docker Desktop
192.168.1.155 host.docker.internal
192.168.1.155 gateway.docker.internal
# To allow the same kube context to work on the host and the container:
127.0.0.1 kubernetes.docker.internal
# End of section

127.0.0.1    lingo.local
```

6. Hacemos ping a lingo.local:



```
C:\Users\Markel>ping lingo.local

Haciendo ping a lingo.local [127.0.0.1] con 32 bytes de datos:
Respuesta desde 127.0.0.1: bytes=32 tiempo<1m TTL=128
Respuesta desde 127.0.0.1: bytes=32 tiempo<1m TTL=128
Respuesta desde 127.0.0.1: bytes=32 tiempo<1m TTL=128
Respuesta desde 127.0.0.1: bytes=32 tiempo<1m TTL=128

Estadísticas de ping para 127.0.0.1:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 0ms, Máximo = 0ms, Media = 0ms
```

Y ya tenemos el DNS configurado. Ponemos lingo.local en el navegador.


No es seguro lingo.local

Centro educativo

LINGO

¡Bienvenido a Lingo!

El juego de palabras más divertido para desafiar tu mente.

© 2025 Lingo. Todos los derechos reservados.

F