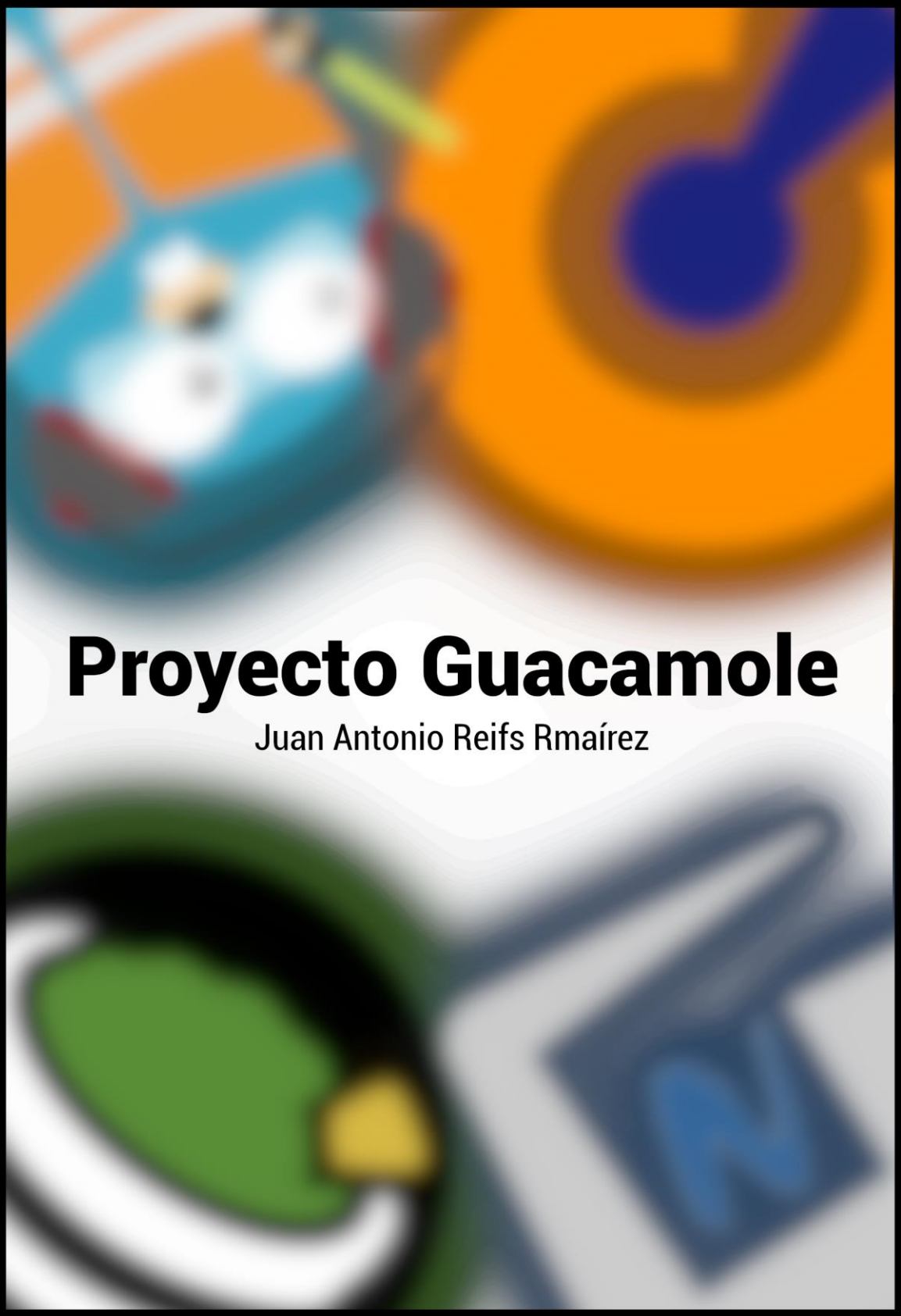
Proyecto Guacamole
Juan Antonio Reifs Rmaírez

Índice

1.- Definición del proyecto

Este proyecto consiste en una implementación funcional de la aplicación web Guacamole en un VPS accesible desde cualquier lugar para poder acceder a los equipos de mi red local desde cualquier sitio.

2.- ¿Qué es Guacamole?

Guacamole es una Aplicación Web Java que nos permite conectarnos a los equipos de una red local sin necesidad de usar ninguna aplicación. Esto nos facilita mucho la vida a los administradores de sistemas, los cuales normalmente trabajamos con muchos equipos en una o varias redes locales.

3.- Escenario

El escenario que he elegido para realizar este proyecto es un VPS con la distribución Linux Debian 11. Este está alojado en OVH y tiene un dominio llamado “juananveintidi.es”. Dicha máquina cuenta con Docker y docker-compose instalados para facilitarnos el despliegue, mantenimiento y combinación con los servicios que necesitemos o queramos añadir.

4.- Configuración

Las configuraciones de los diferentes contenedores que ejecutan los servicios es bastante simple, ya que, como he mencionado anteriormente, Docker nos facilita bastante el despliegue de estas aplicaciones y para hacerlo más sencillo a la hora de desplegar los contenedores, he generado desde 0 un docker-compose, el cual lanza casi todos los servicios que vamos a usar.

Ahora lo vamos a ir describiendo poco a poco, pero al final del documento dejaré el docker-compose íntegro para que se pueda revisar fácilmente.

4.1.- MariaDB

Este es el primer contenedor que configuré en el docker-compose, ya que es la base de todo, porque sin la base de datos, ni el demonio ni la aplicación web pueden funcionar.

La configuración es la siguiente:

```
  mariadb_guacamole:
    container_name: mariadb
    image: mariadb: latest
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: guacamole_db
      MYSQL_USER: guacamole_user
      MYSQL_PASSWORD: guacamole_passwd
    volumes:
      -              ./initdb/001-create-schema.sql:/docker-entrypoint-initdb.d/001-create-schema.sql
      -          ./initdb/002-create-admin-user.sql:/docker-entrypoint-initdb.d/002-create-admin-user.sql
```

- **mariadb_guacamole:** Indicamos el inicio del contenedor y definimos el servicio
- **container_name: mariadb:** Definimos el nombre del contenedor
- **image: mariadb: latest:** Indicamos que vamos a usar la última versión de la imagen de mariadb almacenada en DockerHub
- **environment:** Iniciamos la sección de variables de entorno, las cuales definiremos a continuación:
    - **MYSQL_ROOT_PASSWORD: root:** Contraseña del usuario root de Mariadb
    - **MYSQL_DATABASE: guacamole_db:** Base de datos predeterminada que va a usar guacamole.
    - **MYSQL_USER: guacamole_user:** Usuario propietario de la base de datos mencionada anteriormente
    - **MYSQL_PASSWORD: guacamole_passwd:** Contraseña del usuario mencionado anteriormente
- **volumes:** Inicio de la sección de volúmenes, la cual la usaremos para introducir los scripts de creación de la base de datos.

4.2.- guacd

Una vez que tenemos la configuración de la base de datos que sostiene a esta aplicación, vamos a por la segunda parte de la configuración: el demonio de "`guacd`". Este contenedor va a contener al demonio de la aplicación web, el cual va a hacer de intermediario entre la aplicación web "`guacamole`" y la base de datos anteriormente configurada.

Dicha configuración en el docker-compose es la siguiente:

```
guacd:
    image: guacamole/guacd:latest
    container_name: guacd
    restart: always
```

En esta configuración no hay demasiado que explicar, ya que su configuración es muy simple, porque esta imagen no admite ningún tipo de variables ni configuración de ningún tipo (excepto la variable de activar el modo *debug* usada por los desarrolladores para depurar bugs o revisar cambios realizados en el demonio).

- **guacd:** Indicamos el inicio del contenedor y definimos el servicio
- **image: guacamole/guacd:latest:** Indicamos que vamos a usar la última versión de la imagen de guacd almacenada en DockerHub
- **container_name: guacd:** Definimos el nombre del contenedor
- **restart: always:** (Opcional) Le indicamos a Docker que al mínimo error reinicie siempre el servicio.

4.3.- guacamole

Esta es la propia aplicación web, es decir, el servicio que nos va a permitir ver la interfaz web y comunicarnos tanto con el demonio "guacd" como con la base de datos "mariadb" anteriormente configurados.

En la configuración que se mostrará a continuación habrá algunas líneas de "traefik", el cual explicaremos más adelante.

```
guacamole:
    image: guacamole/guacamole:latest
    container_name: guacamole
    restart: always
    environment:
      GUACD_HOSTNAME: guacd
      MYSQL_DATABASE: guacamole_db
      MYSQL_USER: guacamole_user
      MYSQL_PASSWORD: guacamole_passwd
      MYSQL_HOSTNAME: mariadb_guacamole
      MYSQL_PORT: 3306
    labels:
      -
"traefik.http.routers.guacamole.rule=Host(`apache.app.test`)"
      -
#"traefik.http.middlewares.autenticacion.basicauth.users=admin:$
$2y$$05$$LSs/ugzvDieAjeB6q1OzSeqRyJO0tpb7SXNDSpzZaHkHRHTN9Xgg2"
      -
#"traefik.http.routers.guacamole.middlewares=autenticacion"
    depends_on:
      - mariadb_guacamole
      - guacd
```

- **guacamole:** Indicamos el inicio del contenedor y definimos el servicio
- **image: guacamole/guacamole:latest:** Indicamos que vamos a usar la última versión de la imagen de guacamole almacenada en DockerHub
- **container_name: guacamole:** Definimos el nombre del contenedor
- **restart: always:** (Opcional) Le indicamos a Docker que al mínimo error reinicie siempre el servicio.

- **environment:** Iniciamos la sección de variables de entorno, las cuales definiremos a continuación:
    - **GUACD_HOSTNAME: guacd:** El nombre del contenedor que ejecuta el demonio guacd
    - **MYSQL_DATABASE: guacamole_db:** Nombre de la base de datos existente que usará guacamole predeterminadamente y donde almacenará todos los cambios que realicemos dentro de la aplicación web.
    - **MYSQL_USER: guacamole_user:** Usuario propietario de la base de datos mencionada anteriormente.
    - **MYSQL_PASSWORD: guacamole_passwd:** Contraseña de dicho usuario
    - **MYSQL_HOSTNAME: mariadb_guacamole:** Nombre del contenedor que está ejecutando la base de datos
    - **MYSQL_PORT: 3306:** Puerto por el que está escuchando y recibe las peticiones el contenedor de la base de datos
- **labels:** Inicio de la sección de etiquetes, las cuales son usadas por traefik (dicho contenedor se explicará más adelante).
    - **- "traefik.http.routers.guacamole.rule=Host(`apache.app.test`)":** Esta etiqueta tiene la función de indicarle a traefik que todo el tráfico que vaya para la dirección "`apache.app.test`" lo redirija a la aplicación web "`guacamole`".
    - **- #"traefik.http.middlewares.autenticacion.basicauth.users=admin:$$2y$$05$$LSs/ugzvDieAjeB6q1OzSeqRyJO0tpb7SXNDSpzZaHkHRHTN9Xgg2":** Esta etiqueta pertenece a un middleware de traefik, el cual se encarga de añadir una autenticación básica para acceder a un sitio web al que traefik tenga acceso.
    - **- #"traefik.http.routers.guacamole.middlewares=autenticacion":** Esta etiqueta le indica a traefik que aplique y ponga en uso la etiqueta anterior para que al entrar en dicha dirección nos pida una autenticación básica y así poder restringir el acceso a ciertos sitios webs que administremos.
- **depends_on:** Inicio de la sección de dependencias. En esta sección he incluido los contenedores de mariadb y guacd, porque si alguno de estos dos contenedores falla, este contenedor queda inservible.

4.4.- traefik

Este contenedor se va a encargar de las redirecciones y seguridad de este proyecto. Este tendrá la función de proxy inverso y de aplicar los certificados SSL de Let'sEncrypt a nuestras páginas.

La configuración de este servicio es un poco extensa así que la dividiré en sub apartados para que se puedan diferenciar los diferentes archivos de configuración.

4.4.1.- Configuración de traefik: docker-compose

Primero comencemos por el fichero docker-compose, ya que es la base de todo este proyecto:

```
  traefik:
    container_name: traefik
    image: traefik:latest
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - ./traefik/traefik.yaml:/traefik.yaml
      - ./traefik/acme.json:/acme.json
    command:
      - "--providers.docker"
      - "--entrypoints.guacamole.address=:80"
      - "--
entrypoints.guacamole.http.redirections.entrypoint.to=guacamole_
seguro"
      - "--
entrypoints.guacamole.http.redirections.entrypoint.scheme=https"
      - "--entrypoints.guacamole_seguro.address=:443"
    ports:
      - "80:80"
      - "443:443"
```

- **traefik:** Indicamos el inicio del contenedor y definimos el servicio
- **container_name: traefik:** Definimos el nombre del contenedor
- **image: traefik:latest:** Indicamos que vamos a usar la última versión de la imagen de traefik almacenada en DockerHub

- **command:** Inicio de la sección de comandos, todo el contenido de esta sección sirve para redirigir todo el tráfico que entre por "*http*" a "*https*".
- **volumes:** Inicio de la sección de volúmenes, la cual la usaremos para indicar dónde se encuentran los archivos de configuración de traefik.
    - **- /var/run/docker.sock:/var/run/docker.sock:** Uso del socket de Docker, ya que traefik necesita usar la API de Docker para poder interactuar con los demás contenedores y así poder accederá los datos que definamos en estos.
    - **- ./traefik/traefik.yaml:/traefik.yaml:** Fichero de configuración de traefik, el cual explicaremos en el siguiente apartado.

    - **- ./traefik/acme.json:/acme.json:** Fichero de la API ACME utilizada para cifrar nuestra conexión mediante Let'sEncrypt

- **ports:** Iniciamos la sección en la cual hacemos el enlace de los puertos de nuestro contenedor con los de nuestra máquina host, en este caso, queremos abrir los puertos "*80:80*" para recibir peticiones "*http*" y los puertos "*443:443*" para recibir peticiones "*https*".

4.4.2.- Configuración de traefik: traefik.yaml

Ahora vamos a explicar el fichero de configuración de traefik:

```
entryPoints:
  guacamole:
    address: ":80"
  guacamole_seguro:
    address: ":443"

providers:
  docker:
    endpoint: "unix:///var/run/docker.sock"

certificatesResolvers:
  myresolver:
    acme:
      email: amdin@juananveintidi.es
      storage: /acme.json
      httpchallenge:
        entryPoint: guacamole
```

- **entryPoints:** Iniciamos el punto de entrada para los puertos que va a usar traefik.
    - **guacamole:** Nombre descriptivo que le asignamos al puerto 80 en este caso
    - **guacamole_seguro:** Nombre descriptivo que le asignamos al puerto 443 en este caso
- **providers:** Iniciamos el apartado de configuración de los proveedores
    - **docker:** Definimos que nuestro proveedor va a ser Docker
        - **endpoint: "unix:///var/run/docker.sock":** Definimos el socket de docker para que traefik pueda usarlo
- **certificatesResolvers:** Iniciamos el apartado de los resolvers, los cuales se van a encargar de generar y regenerar los certificados de Let'sEncrypt.
    - **myresolver:** Nombre que le damos al resolver para identificarlo
        - **acme:** Protocolo que usamos para generar los certificados y a continuación, pondremos nuestros datos para que se nos genere nuestro certificado.

5.- OpenVPN

OpenVPN es un servicio de VPN el cual nos va a permitir conectarnos a otra red local distinta a la nuestra mediante un túnel.

Es necesario usarlo en este proyecto, ya que si nuestro servidor guacamole no está conectado a la red local de donde queremos acceder a los equipos no vamos a poder tener acceso a los mismos.

Para facilitar la tarea de creación de la VPN he usado un script llamado openvpn-install.sh, el cual está muy completo, ya que te genera los certificados a tu gusto y una vez que tienes instalado el servidor VPN en tu máquina, puedes usar el mismo script tanto para generar diferentes certificados para diferentes clientes, como revocar los mismos y desinstalar por completo de nuestro sistema OpenVPN.

Para conectar el proyecto con la red local de mi casa he generado un archivo de configuración "*.ovpn*" el cual he instalado en el servidor VPS de OVH.