Universidad de San Carlos de Guatemala
Ingeniería en Ciencias y Sistemas
Redes de Computadoras 2
Primer Semestre 2021
Ing. Pedro Pablo Hernandez
Auxiliares: Sergio Silva / Wilson Guerra

Práctica 4

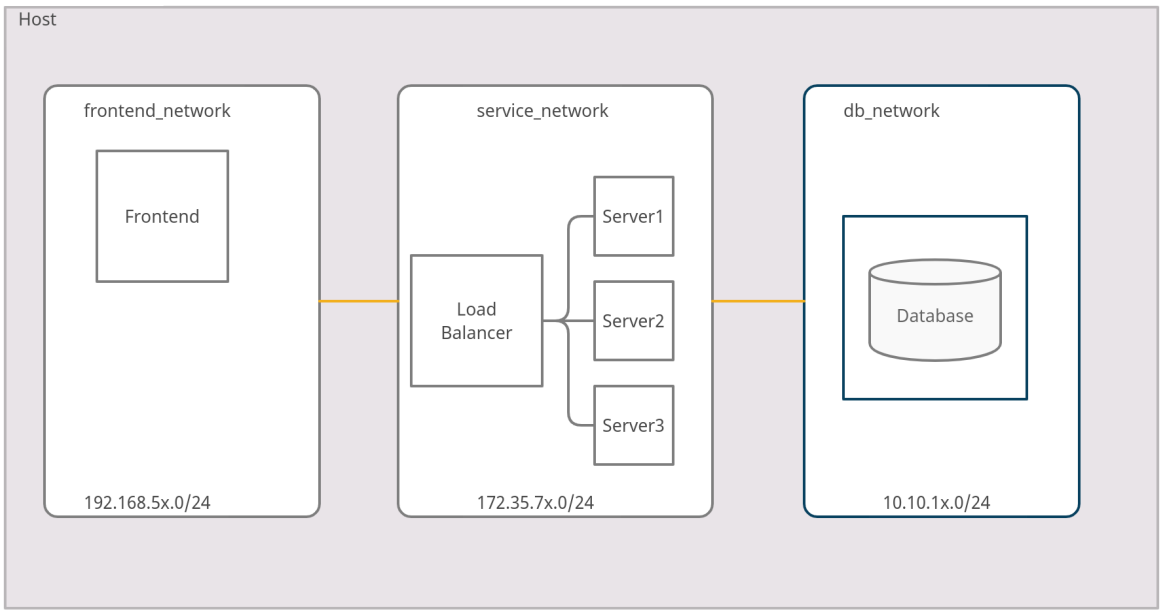
Objetivos

- Conocer las herramientas provistas por los sistemas linux para la creación y manejo de Virtual Networks.
- Conocer el dispositivo Bridge(Switch virtual) en linux.
- Conocer las interfaces virtuales Veth.
- Comprender el funcionamiento de las redes virtuales y como estas permiten la comunicación entre host virtuales(Maquinas virtuales / Contenedores).
- Conocer los comandos provistos por Docker para la creación de redes virtuales usando el driver Bridge.
- Realizar agrupamiento de contenedores y segmentación de redes en Docker.
- Utilizar balanceo de carga por software para distribuir el tráfico entre contenedores.

Descripción del problema

La Escuela de Ciencias y Sistemas de la Universidad de San Carlos de Guatemala desea reemplazar el sistema actual de envío de reportes de actividades para los alumnos que realizan sus prácticas finales. Dicho sistema se alojará en un único servidor en la nube. Se ha identificado que la mayoría de practicantes espera a las fechas límites para realizar y enviar sus reportes, por lo que en dichas fechas se experimenta una caída en el sistema actual. Es por esto, que es de vital importancia que el nuevo sistema a desarrollar sea fácilmente escalable. Se plantea el uso de docker para contenerizar las aplicaciones o servicios que conformen el nuevo sistema. Se le solicita a usted como estudiante de la escuela, realizar un demo funcional.

Arquitectura



Nota: Para las direcciones de red a utilizar en cada Network, se utilizará el mismo mecanismo que se ha venido utilizando para las prácticas anteriores, es decir, se debe reemplazar la “x” en la dirección de red por el número de grupo. Si su grupo es de 2 dígitos, se utilizará el segundo dígito.

db_network

En esta network se alojaran los contenedores con el dbms. Los contenedores pertenecientes a dicha network solo pueden ser accedidos por contenedores de la service_network. No se permite el acceso directo a través de contenedores de otras networks(excepto service_network) ni tampoco es accesible desde el host.
Inicialmente solo se contará con un contenedor de base de datos.

service_network

En esta network se alojan los contenedores que proveerán de servicios para interactuar con la base de datos a través de una api. Esta network también contendrá un balanceador de carga, el cual se encargará de distribuir el tráfico entre los distintos contenedores que contienen la api. Es importante resaltar que el acceso a la api, sólo se puede realizar a través del balanceador de carga, es decir, ninguno de los servidores con la api pueden ser accedidos de manera directa.
Inicialmente se contará con 3 tres contenedores que provean la api.

frontend_network

En esta network se alojara el contenedor que proveerá del frontend, dicho frontend consumirá el api a través del balanceador de carga. Se hará uso de un solo contenedor con frontend.

Base de datos

La base de datos a utilizar queda a discreción del estudiante. Se debe utilizar un volumen para tener persistencia de información.

Servidor API

La tecnología a utilizar para realizar la api también queda a discreción del estudiante. Cada servidor debe tener una firma o identificador único, el cual adjuntará a cada respuesta para las peticiones que procese. Ejemplo: ***"Solicitud atendida por el servidor xxxxxxx".***
Ya que son 3 servidores, se tomará como identificador/firma los carnets de los miembros del grupo.
Se recomienda el uso de variables de entorno para manejar los identificadores de cada servidor sin necesidad de modificar el código para cada contenedor.

Balanceador de carga

Se debe crear un contenedor con un balanceador de carga usando NGINX. Dicho balanceador de carga distribuye las peticiones recibidas entre los servidores API.
Se debe utilizar el mecanismo de balanceo de carga Round-Robin.

Frontend

La tecnología o herramientas a utilizar quedan a discreción del estudiante. Debido a que el objetivo de esta demo solo es testear la segmentación de redes en docker, y el balanceo de carga, solo se solicita que el usuario pueda realizar las siguientes acciones:

- Enviar reporte: Un reporte estará conformado por los siguientes datos:
    - Carnet del practicante que envía el reporte.
    - Nombre del estudiante.
    - Curso/proyecto al cual está asignado.
    - Cuerpo de reporte o mensaje.
- Listar reportes almacenados: Devuelve el listado de todos los reportes enviados por los practicantes, si recibe un carnet como parámetro, devolverá sólo los reportes enviados por dicho practicante.
- Ver reporte individual: Permite ver un reporte de manera individual. Los datos a mostrar son:
    - Carnet del practicante que envía el reporte.
    - Nombre del estudiante.
    - Curso/proyecto al cual está asignado.
    - Cuerpo de reporte o mensaje.
    - Servidor que proceso/ingreso dicho reporte.

Pantalla de Ingreso de reporte

Pantalla de Vista de reporte individual

Nota: el valor del campo “Procesado por” es el identificador(carnet) del servidor que guardó dicho reporte en la base de datos.

Pantalla de listado de reportes

Si al presionar en el botón buscar el campo carnet se encuentra vacío, se retornan todos los reportes almacenados en el sistema. Si el campo carnet contiene un valor, se retorna solo los reportes que coincidan con dicho carnet.

Restricciones

- La práctica se realizará en grupos de máximo 3 integrantes.
- Todos los integrantes del grupo deben de tener conocimiento del desarrollo de la práctica.
- La práctica se debe realizar sobre un host con linux, este puede ser una instalación física, una máquina virtual o un servidor en la nube.
- Es obligatorio el uso de Docker y Docker Compose.
- Para la calificación se debe de presentar la práctica en una computadora de los integrantes del grupo.
- Se debe de crear un repositorio de GitHub donde se irá actualizando el desarrollo de la práctica, el cual debe de contener como mínimo 2 commits por semana por parte de cada uno de los integrantes del grupo.
- Durante la calificación se preguntará información relevante de la práctica para comprobar la autoría del mismo.

- El manual técnico debe ser un pdf con el nombre **Practica4_Manual_#grupo.pdf**

Penalizaciones

- Falta de seguimiento de desarrollo continuo por medio de Github tendrá una penalización del 10%.
- Falta de seguimiento de instrucciones conforme al método de entrega (nombre del repositorio) tendrá una penalización del 5%.
- Falta de puntualidad conforme a la entrega tendrá una penalización de la siguiente manera:
    a. 1-10 minutos – 10%.
    b. 11-59 minutos – 30%.
    c. Pasados 60 minutos tendrá una nota de 0 y no se calificará.

Observaciones

- La entrega se realizará por medio de **Github**, cada grupo deberá crear un repositorio con el nombre: **REDES2_1S2021_GRUPO#**, ejemplo: REDES2_1S2021_GRUPO3, y agregar a su auxiliar correspondiente como colaborador del mismo, para poder analizar su progreso y finalmente a partir del mismo repositorio realizar la calificación correspondiente. ***Este será el repositorio a utilizar a partir de ahora para la entrega de prácticas y proyectos, por lo que para cada práctica/proyecto, se deberá crear un directorio con el nombre de la práctica o proyecto, ejemplo:***
    - **Practica 1**
    - **Practica 2**
    - **…….**
    - **Proyecto 1**
- Dentro del repositorio anterior se debe crear un directorio llamado “Práctica 4”, el cual debe contener todos los archivos utilizados en la práctica(Código fuente, Dockerfiles, archivos yaml de compose,etc.)
- Además de tener a su auxiliar como colaborador del repositorio para tener un control y orden de las personas que entreguen deberán de colocar el Link de su repositorio en la Tarea que cada auxiliar asignará en su plataforma correspondiente.
- Fecha y hora de entrega: **Miércoles 24 de Marzo, antes de las 23:59 horas.**
- **Las copias serán penalizadas con una nota de 0 y castigadas según lo indique el reglamento.**