

# Simulación Profesional de Redes Segmentadas con Docker y Linux

## Resumen Ejecutivo

El presente documento describe una metodología avanzada para simular redes corporativas segmentadas utilizando Docker, Docker Compose y herramientas de red de bajo nivel en sistemas basados en Linux. Esta guía permite replicar entornos de producción, probar políticas de seguridad, segmentación por VLAN, reglas de firewall y esquemas de escalabilidad en entornos controlados, ligeros y fácilmente replicables. Todo esto sin requerir software de virtualización pesado ni hardware físico especializado.

## Introducción a la Simulación de Redes con Docker

Docker ha superado su rol inicial como entorno de contenedores para microservicios, permitiendo también la construcción de topologías de red complejas a nivel de capa 2 y 3. Gracias al soporte para redes tipo **bridge**, direccionamiento IP personalizado, y uso de **iptables** dentro de contenedores, es posible crear entornos que simulen switches, routers, VLANs lógicas, y segmentos aislados.

Esto convierte a Docker en una poderosa herramienta para enseñar, probar o documentar:

- Enrutamiento IP entre subredes.
- Segmentación VLAN-to-VLAN.
- Firewalls distribuidos.
- Control de acceso a servidores internos.
- Simulación de ataques y defensa en laboratorio.

## Diseño Lógico de la Red

La red corporativa simulada incluye:

- Cuatro VLANs: Administración, Desarrollo, Soporte Técnico y Servidores.
- Un router virtual: responsable del enrutamiento inter-VLAN y aplicación de políticas de firewall.
- Clientes y servidores simulados: contenedores conectados a redes específicas.
- Opcionalmente, una VLAN para invitados o una salida a Internet simulada.

El diseño está estructurado bajo un enfoque jerárquico:

- Capa Core: router central (contenedor Alpine con iptables).
- Capa de Acceso: dispositivos finales (contenedores simulando PCs y servidores).
- VLANs Lógicas: cada red bridge de Docker representa una VLAN.

### Configuración de VLANs mediante Docker Compose

Docker Compose permite definir múltiples redes tipo bridge, asignando subredes personalizadas y direcciones fijas.

Ejemplo de definición de redes:

networks:

vlan10:

driver: bridge

ipam:

config:

- subnet: 192.168.10.0/24

vlan20:

driver: bridge

ipam:

config:

- subnet: 192.168.20.0/24

Cada servicio (contenedor) se conecta a una o más de estas redes según su función. Así se simula la pertenencia a una VLAN específica.

## Definición de Servicios de Red (Router, Clientes, Servidores)

Los dispositivos son representados por contenedores Alpine con configuración mínima.

Router:

router:

image: alpine

command: ["sh", "-c", "apk add iptables && sleep infinity"]

privileged: true

networks:

vlan10:

ipv4\_address: 192.168.10.1

vlan20:

ipv4\_address: 192.168.20.1

...

Clientes:

admin\_pc:

image: alpine

command: ["sh", "-c", "sleep infinity"]

networks:

vlan10:

ipv4\_address: 192.168.10.10

Este patrón se repite para cada grupo de dispositivos en cada VLAN.

## Simulación de Seguridad y Firewall con IPTables

Desde el contenedor que hace de router, se utilizan reglas `iptables` para controlar el tráfico entre VLANs.

Ejemplo de política segmentada:

# Política por defecto: bloquear forwarding

```
iptables -P FORWARD DROP
```

# Permitir Administración hacia todas las VLANs

```
iptables -A FORWARD -s 192.168.10.0/24 -j ACCEPT
```

# Bloquear desarrollo y soporte entre sí

```
iptables -A FORWARD -s 192.168.20.0/24 -d 192.168.30.0/24 -j DROP
```

```
iptables -A FORWARD -s 192.168.30.0/24 -d 192.168.20.0/24 -j DROP
```

# Permitir acceso a servidores

```
iptables -A FORWARD -d 192.168.40.0/24 -j ACCEPT
```

Estas reglas replican exactamente la lógica definida en redes reales, con separación de privilegios y políticas de acceso.

## Simulación de Tráfico y Diagnóstico

Dentro de cualquier contenedor, se pueden instalar herramientas como `ping`, `traceroute`, `curl`, o `tcpdump`.

Por ejemplo:

```
apk add iputils iproute2 tcpdump
```

```
ping 192.168.20.10
```

```
tcpdump -i eth0
```

Esto permite analizar el flujo de paquetes, verificar rutas, observar bloqueos por firewall, etc.

## Expansión del Entorno y Escalabilidad

Para escalar el entorno:

- Agregar más VLANs es tan simple como definir más redes en el `docker-compose.yml`.
- Agregar más servicios: duplicar secciones `admin_pc`, `dev_pc`, etc.
- Conectar a Internet real: habilitar NAT y forwarding desde el router.
- Implementar DNS interno: montar un contenedor con `dnsmasq`.
- Usar volúmenes compartidos o contenedores NGINX para simular web interna.

## Casos de uso posibles

- Simulación de incidentes (MITM, escaneo de puertos).
- Pruebas de respuesta a ataques (con herramientas como `nmap`, `hping3`, etc.).
- Entrenamiento para administradores de red o ciberseguridad.
- Laboratorios para cursos de networking, hacking ético o DevSecOps.

### **Ventajas de este enfoque**

- Ligereza: no se requieren VMs ni sistemas operativos completos.
- Repetibilidad: puedes reiniciar, levantar y destruir laboratorios en segundos.
- Control completo: puedes manipular las reglas de red a nivel de paquete.
- Portabilidad: puedes subir este entorno a cualquier host Linux o WSL.

### **Conclusión**

El uso de Docker como entorno de simulación de redes empresariales representa una alternativa potente, moderna y flexible frente a simuladores tradicionales. Combinando Docker Compose, Linux, iptables y herramientas básicas, es posible representar con fidelidad la segmentación lógica, las políticas de seguridad, el flujo de datos y los desafíos reales de una red empresarial.

Esta solución no solo es técnicamente sólida, sino también educativa y escalable. Su adopción puede transformar la forma en que equipos técnicos diseñan, prueban y enseñan redes seguras.