**Analista en Infraestructura Informática**

**Administración de Servidores Linux**

**Obligatorio 1er semestre 2024**

**Profesor:** Roberto Wagner

**Integrante**

|  |  |
| --- | --- |
| **Alumno** | **Número de estudiante** |
| Joaquín Manzanar | 28 03 73 |

**Índice**

[**Presentación 3**](#_Toc170176498)

[**Máquinas 3**](#_Toc170176499)

[**Servicios en router 4**](#_Toc170176500)

[**Network Address Traslation 4**](#_Toc170176501)

[**Port forwarding 7**](#_Toc170176502)

[**Servicios en servidores 9**](#_Toc170176503)

[**DHCP 9**](#_Toc170176504)

[**DNS 11**](#_Toc170176505)

[**Aplicación web en contenedor 13**](#_Toc170176506)

[**Servidor web de una aplicación Java y Base de datos 14**](#_Toc170176507)

[**Reverse proxy 16**](#_Toc170176508)

[**Configuración de SSL 17**](#_Toc170176509)

[**Cockpit 19**](#_Toc170176510)

[**Investigación: 20**](#_Toc170176511)

[**Bibliografía 21**](#_Toc170176512)

# **Presentación**

A diagram of a router

Description automatically generated

En este documento se muestra la implementación y configuración del entorno de red planteado. El mismo está formado por una red de servidores, una red de equipos interna y un router.

Se irá punto a punto mostrando capturas y explicando detalladamente cómo fueron configuradas las máquinas virtuales (VM) para cumplir con el escenario.

Fueron cubiertas tanto las funcionalidades obligatorias como las opcionales.

**Máquinas virtuales**

Fueron usadas las imágenes de sistema brindadas en clase. Sin GUI para VM Router y con GUI para VM Servidor y VM Equipo

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nombre de VM** | **RAM** | **Procesador** | **Interfaces de Red** | **IP** |
| Router | 512 Mb | 1 núcleo | NAT (enp0s3)  Red Servidores (enp0s8)  Red Equipos (enp0s9) | DHCP | 10.0.1.1 | 10.0.2.1 |
| Servidor | 10 Gb | 3 núcleos | Red Servidores | 10.0.1.10 |
| Equipo | 10 Gb | 2 núcleos | Red Equipos | 10.0.2.10 |

Todas las VM fueron actualizadas con el comando ***sudo dnf update*** previo a su administración para contar con las características más recientes.

**Servicios en Router**

## **Network Address Traslation**

La primera funcionalidad aplicada fue el **Acceso a Internet (NAT).** Consiste en permitir que los equipos de la red de servidores o de equipos tengan acceso a recursos externos mediante el router.

**Configuraciones**

Se habilitaron 3 ***adaptadores de red*** en el router de acuerdo con la siguiente tabla:

|  |  |  |
| --- | --- | --- |
| **Adaptador** | **ID** | **Configuración** |
| 1 | enp0s3 | NAT |
| 2 | enp0s8 | Servidores |
| 3 | enp0s9 | Equipos |

Se instaló el paquete ***iptables.***Este permite configurar las reglas NAT pensando en el correcto tráfico de datos. Luego se inició y habilitó el servicio.

Las reglas NAT serán:

*sudo iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE*

*sudo iptables -A FORWARD -i enp0s8 -o enp0s3 -j ACCEPT*

*sudo iptables -A FORWARD -i enp0s9 -o enp0s3 -j ACCEPT*

*sudo iptables -A FORWARD -i enp0s3 -o enp0s8 -m state –state*

*RELATED,ESTABLISHED -j ACCEPT*

*sudo iptables -A FORWARD -i enp0s3 -o enp0s9 -m state –state*

*RELATED,ESTABLISHED -j ACCEPT*

Se guardan con el siguiente comando:

*sudo service iptables save*

Se habilitó el ***reenvío de IP*** para permitir la transmisión de paquetes entre interfaces de red. Esto se logra editando el archivo /etc/sysctl.conf y agregando el comando *net.ipv4.ip\_forward = 1*. Finalmente se aplican los cambios con *sudo sysctl -p.*

Lo siguiente es configurar los adaptadores de red mediante el comando *sudo nano /etc/sysconfig/network-scripts/ifcfg-enp0sX*

* Para enp0s3: DEVICE=enp0s3

BOOTPROTO=dhcp

ONBOOT=yes

* Para enp0s8: DEVICE=enp0s8

BOOTPROTO=static

IPADDR=10.0.1.1

NETMASK=255.255.255.0

ONBOOT=yes

* Para enp0s9: DEVICE=enp0s9

BOOTPROTO=static

IPADDR=10.0.2.1

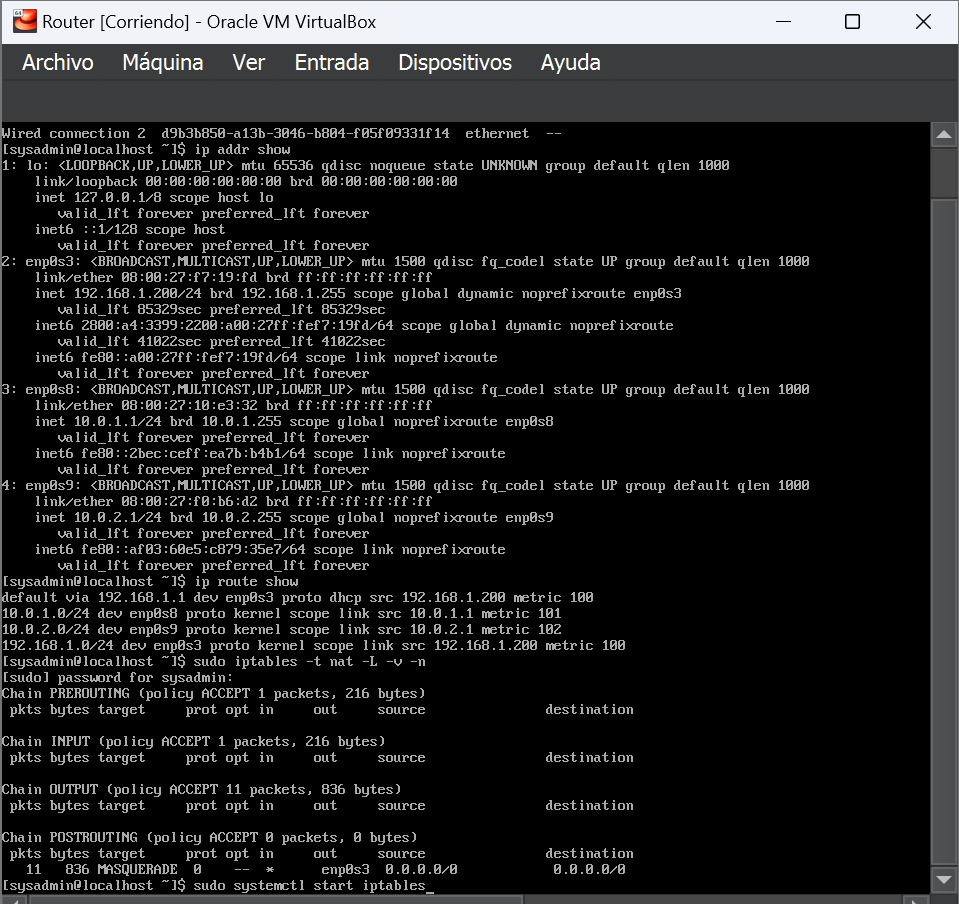
NETMASK=255.255.255.0

ONBOOT=yes

De esta forma, el router puede mediar entre las redes y los recursos externos con éxito mediante NAT.

**Verificación**

En las siguientes imágenes se verifican las correctas configuraciones de interfaces de red y reglas NAT:



**A screenshot of a computer

Description automatically generated**

Desde VM Servidor se intenta un ping a VM Router y a Google.com de manera exitosa:

Texto

Descripción generada automáticamente

También se comprueba desde VM Equipo:

Texto

Descripción generada automáticamente

Demostración de que VM Servidor y VM Equipo apuntan al router:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

**Conclusión**

Los equipos de la red de servidores y equipos van a tener acceso a Internet y demás recursos externos sin problemas gracias al acceso controlado por parte del router.

## **Port forwarding**

La segunda funcionalidad aplicada es el **reenvío de puertos.** Consiste en permitir a dispositivos externos el acceso a servicios alojados en servidores internos.

En este caso, se configurará VM Router para que las solicitudes HTTP entrantes desde Internet sean reenviadas al Servidor Web ubicado en la red de servidores.

**Configuraciones**

Se añaden dos reglas para reenviar el puerto 80 (HTTP) de la interfaz NAT (enp0s3) al servidor web en la red interna de servidores.

*sudo iptables -t nat -A PREROUTING -i enp0s3 -p tcp --dport 80 -j DNAT --to-destination 10.0.1.10:80*

*sudo iptables -A FORWARD -p tcp -d 10.0.1.10 --dport 80 -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT*

Se guardan las reglas incluso luego de reiniciar

*sudo sh -c “iptables-save > /etc/iptables/rules.v4”*

La primera regla indica que cualquier paquete TCP que llegue a enp0s3 en el puerto 80 debe ser redirigido a la IP 10.0.1.10 en el puerto 80.

La segunda regla permite que el tráfico reenviado pase a través del router hacia el servidor web.

Luego de configurada VM Router, se instala Apache en VM Servidor y se verifica que esté “escuchando” en el puerto 80.

**Verificación**

Efectivamente, apache está “escuchando” en el puerto 80.

Captura de pantalla de un celular

Descripción generada automáticamente

Finalmente, se verifican las reglas de Port Forwarding configuradas correctamente.

A screenshot of a computer

Description automatically generated

**Conclusión**

Las solicitudes entrantes al puerto 80 serán redirigidas al servidor web de la red interna. Esto permite que sea accesible desde el exterior sin exponer su IP privada.

**Servicios en Servidores**

## **DHCP**

La primera funcionalidad aplicada al servidor fue **DHCP.** Este servicio asigna direcciones IP dinámicamente a dispositivos de la red interna de equipos y en el caso de los servidores, lo hace basado en sus direcciones MAC.

**Configuraciones**

Se instala el servicio *dhcp-server* para luego editar el archivo de configuración */etc/dhcp/dhcpd.conf*

Se define a enp0s3 como la interfaz a “escuchar”

Texto

Descripción generada automáticamente

En el archivo *dhcp.conf* se especifica el rango de direcciones IP a asignar. Será desde 10.0.1.20 a 10.0.1.99.

Para el rango de IPs de la red de equipos será desde 10.0.2.50 a 10.0.2.99

Texto

Descripción generada automáticamente

**Verificación**

Se creó una VM llamada Prueba la cual pertenece a la red de Servidores. Como puede verse, se le asignó la dirección IP 10.0.1.20 lo que demuestra el correcto funcionamiento del servicio.

Texto

Descripción generada automáticamente

Luego se unió a VM Prueba a la red de equipos y como se ve a continuación, se le brinda una IP acorde al rango definido previamente para esta red.

Texto

Descripción generada automáticamente

**Conclusión**

La proporción de direcciones IP está asegurada por parte de VM Servidor gracias a la correcta configuración del servicio.

## **DNS**

El siguiente servicio implementado fue **DNS** el cual brinda resolución de nombres de dominio a direcciones IP, además de la traducción inversa.

**Configuraciones**

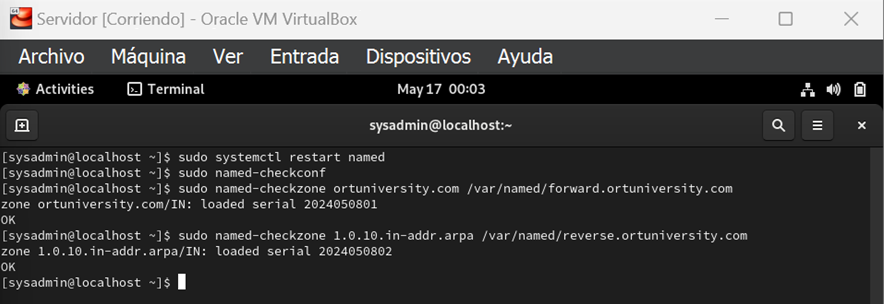
Se instala el servicio *DNS (bind)* para luego modificar el archivo *named.conf* con las zonas de búsqueda directa e inversa.

Texto

Descripción generada automáticamente

**Verificación**

En la siguiente imagen puede comprobarse la correcta definición de las zonas de búsqueda directa e inversa



Finalmente, en la última captura, se aprecia la correcta resolución para *ortuniversity.com*.

Texto

Descripción generada automáticamente

**Conclusión**

El servicio DNS brinda resolución de nombres de dominio a direcciones IP y viceversa correctamente dentro de las redes.

## **Aplicación web en contenedor**

Esta funcionalidad consiste en instalar y desplegar una **aplicación web Java** en un contenedor Docker utilizando Tomcat. Esto permite aislar la aplicación y simplificar su despliegue y gestión.

**Configuraciones**

Primero se instala Docker para luego iniciarlo y habilitarlo. Esto se logra con los comandos ya conocidos *sudo dnf install* y *sudo systemctl start/enable*.

Luego se clona el repositorio según los pasos indicados en el link de GitHub facilitado junto a la letra del obligatorio.

A screenshot of a computer

Description automatically generated

Se sigue por la creación del contenedor con el comando mostrado.

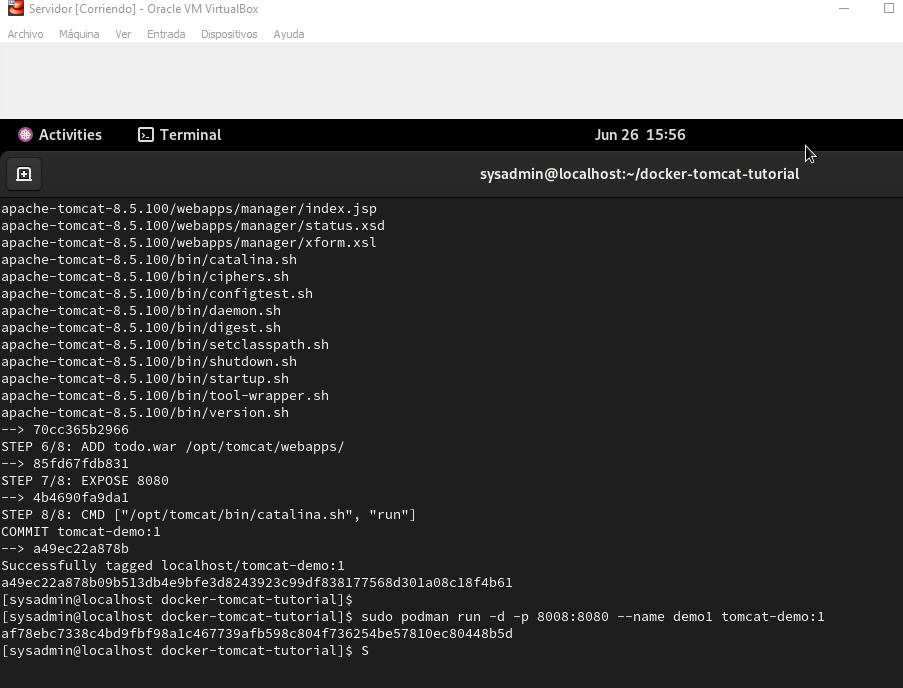
A screenshot of a computer

Description automatically generated

Finalmente, solo resta ejecutarlo para demostrar su funcionamiento.

**Verificación**

Se ejecuta el conteiner con el comando Podman tal como se ve en la imagen:



**Conclusión**

La aplicación Web y el contenedor se crearon y configuraron de acuerdo con lo especificado en el link de github sin inconvenientes.

## **Servidor web de una aplicación Java y Base de datos**

Se trata de desplegar un **archivo WAR en un servidor** y configurar el acceso a la **base de datos.**

**Configuraciones**

Primero se copia el archivo *todo.war* en el directorio de despliegue de Tomcat dentro del contenedor configurado previamente.

Luego se configura la base de datos gracias a crear un archivo *(/opt/config/app.properties)* con los detalles de conexión a la base de datos:

A black screen with white text

Description automatically generated

Finalmente se instala el servicio MySQL y como se ve a continuación, se crea la base de datos además del usuario de prueba.

A screenshot of a computer

Description automatically generated

Debe recordarse reiniciar el contenedor para aplicar los cambios y redesplegar la aplicación.

**Verificación**

En la siguiente imagen puede apreciarse el prompt inicial de la aplicación *Todo* al ingresar a la IP del servidor, más precisamente al puerto 8080.

A screenshot of a computer

Description automatically generated

A continuación, puede verse también el contenido de la base de datos. La misma posee los datos de mi usuario:

A screenshot of a computer

Description automatically generated

**Conclusión**

Se logró el acceso a la aplicación además de las configuraciones necesarias para su funcionamiento. También se ingresaron datos a la base.

## **Reverse proxy**

Esta funcionalidad **redirigirá** **peticiones HTTP** al puerto 8080 donde la aplicación web Java está “escuchando”. Esto permite que Apache gestione las solicitudes proporcionando una capa adicional de seguridad.

**Configuraciones**

Primero se instala el servicio *httpd* y luego los módulos necesarios para habilitar el proxy y compatibilidad con Tomcat *mod\_proxy* y *mod\_proxy\_http.*

Luego fue modificado el archivo *reverse-proxy.conf* de forma tal que todas las solicitudes que lleguen al puerto 80 (predeterminado de HTTP) sean redirigidas al 8080, donde Tomcat está “escuchando”.

A screenshot of a computer

Description automatically generated

Finalmente se reinicia el servicio para aplicar los cambios.

**Verificación**

Al ingresar a la IP pública de VM Servidor se ingresa a la aplicación Todo:

**A screenshot of a computer

Description automatically generated**

**Conclusión**

Con el uso de reverse proxy, se centraliza el manejo de peticiones, además de poder aplicar reglas de seguridad a nivel de Apache lo que supone mayor seguridad. Se verificó que solo se ingresa a la aplicación a través de VM Servidor.

## **Configuración de SSL**

Este **certificado** asegura las comunicaciones entre clientes y servidor web encriptándolas para resguardar datos sensibles durante la transmisión.

**Configuraciones**

Se comienza instalando el módulo *mod\_ssl* para luego crear un certificado. En la imagen se muestra la creación de este:

A screenshot of a computer

Description automatically generated

Posteriormente se edita el archivo de configuración SSL de Apache para que apunte a los archivos del certificado y como siempre, se reinicia el servicio Apache para aplicar cambios.

A screenshot of a computer

Description automatically generated

Finalmente, se abre el puerto 443 en el firewall mediante el comando *sudo firewall-cmd –permanente –add-service=https* y *sudo firewall-cmd –reload* como siempre para aplicar cambios*.*

**Verificación**

Como puede observarse, se logra ingresar a través de HTTPS lo que significa una correcta configuración del certificado.

A screenshot of a computer

Description automatically generated

**Conclusiones**

Todas las comunicaciones entre clientes y servidor web están encriptadas protegiendo así la integridad y confidencialidad de la información que se intercambie.

## **Cockpit**

Esta **herramienta de administración** facilita la gestión de múltiples servidores Linux desde interfaz gráfica. Entre sus principales funciones destacan la gestión de servicios, monitoreo de logs, entre muchas otras.

**Configuraciones**

Se instala Cockpit y se ejecuta el siguiente comando *sudo systemctl enable –now cockpit.socket* para su habilitación.

Se habilita el puerto 9090 en el firewall mediante *sudo firewall-cmd –permanent –zone=public –add-service=cockpit* para permitir el acceso a la interfaz web de Cockpit.

A continuación, se puede apreciar la configuración:

A screenshot of a computer program

Description automatically generated

**Verificación**

Se accede a <https://10.0.1.10:9090> para luego loguearse con las credenciales del servidor (Usr: sysadmin | Pwd: aslxlab)

En la imagen pueden verse, por ejemplo, los logs.

A screenshot of a computer

Description automatically generated

**Conclusiones**

Cockpit simplifica notoriamente la administración de servidores Linux al proporcionar una gran variedad de posibilidades referente a la gestión de estos. Además, presenta una interfaz gráfica sumamente intuitiva para el administrador.

**Investigación: Contenedores**

Son una tecnología de virtualización a nivel del sistema operativo que permite empaquetar una aplicación y todas sus dependencias en una única unidad. Esto garantiza que la aplicación se ejecute de manera consistente sin importar dónde se despliegue.

Se destacan:

* Docker: Es la plataforma de contenedores más conocida. Permite crear, desplegar y gestionar contenedores de manera sencilla.
* Kubernetes: gestiona el despliegue, escalado y operación de aplicaciones en contenedores. Está pensada para grandes escalas.
* Podman: es similar a Docker, pero sin un demon centralizado, lo que da más flexibilidad y seguridad.

Ventajas de los Contenedores

* Portabilidad: Al empaquetar la aplicación con todas sus dependencias, los contenedores garantizan que la aplicación se ejecute de la misma manera en cualquier entorno, ya sea en un servidor local o en la nube.
* Consistencia: Los desarrolladores pueden estar seguros de que el entorno de producción es idéntico al de desarrollo, lo que reduce los problemas de configuración de entorno.
* Eficiencia de Recursos: Los contenedores comparten el núcleo del sistema operativo, lo que permite ejecutar múltiples contenedores en un solo sistema.
* Escalabilidad: se pueden replicar rápidamente para manejar una mayor carga.

# 

# **Bibliografía**

Los links aquí detallados fueron la base de las configuraciones realizadas en este documento, siendo lo aprendido en clase la principal fuente de información.

[NAT](https://www.redeszone.net/tutoriales/redes-cable/configuracion-red-maquina-virtual-vmware/)

[Port forwarding](https://docs.vmware.com/en/VMware-Greenplum/7/greenplum-database/install_guide-enable_iptables.html)

[DHCP](https://www.alibabacloud.com/blog/how-to-set-up-a-dhcp-server-on-a-linux-virtual-machine_595821)

[DNS](https://forums.virtualbox.org/viewtopic.php?t=104164)

[Aplicación Web](https://github.com/emverdes/docker-tomcat-tutorial)

[Reverse proxy](https://www.bing.com/search?q=reverse%20proxy%20linux%20virtualbox&qs=n&form=QBRE&=Buscar%20%7B1%7D%20en%20%7B0%7D&=Buscar%20trabajo%20para%20%7B0%7D&=%25eAdministra%20tu%20historial%20de%20búsqueda%25E&sp=-1&lq=0&pq=reverse%20proxy%20linux%20virtualbox&sc=0-30&sk=&cvid=325363D2C19B44D1825D473D82458CB9&ghsh=0&ghacc=0&ghpl=)

[SSL](How%20to%20generate%20a%20self-signed%20SSL%20certificate%20on%20Linux%20-%20Linux%20Tutorials%20-%20Learn%20Linux%20Configuration)

[Cockpit](https://www.bing.com/search?q=cockpit%20linux%20virtualbox&qs=n&form=QBRE&sp=-1&lq=0&pq=cockpit%20linux%20virtualbox&sc=7-24&sk=&cvid=96186AAE28274735AFE9DC943F89DB2D&ghsh=0&ghacc=0&ghpl=)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**