

TRABAJO INTEGRADOR DE ARQUITECTURA Y SISTEMAS OPERATIVOS

ENTORNOS DE DESARROLLO VIRTUALIZADOS

Integrantes

Cornejo, Diego Gastón

dieguitocornejo52@gmail.com

Dantur, Daniel Fernando Jesús

danidantur@gmail.com

Comisión

M2025-12

Materia

Arquitectura y sistemas operativos

Docente

Mauricio Gabriel Pasti

Docente Tutor

Jerónimo Cortez

Fecha de entrega

05/06/2025

Índice

Índice.....	2
Introducción.....	3
Marco Teórico.....	4
Virtualización.....	4
VirtualBox.....	4
Debian.....	4
Configuración de red: adaptador puente.....	5
Herramientas de administración remota: Remote SSH y Visual Studio Code.....	5
Integración de tecnologías.....	5
Caso Práctico.....	6
Metodología utilizada:.....	9
Resultados Obtenidos:.....	10
Conclusiones:.....	11
Complicaciones que pudieran surgir.....	11
Ventajas del enfoque utilizado.....	12
Desventajas del enfoque.....	12
Posibles mejoras.....	12
Bibliografía.....	14
Anexos.....	14

Introducción

La virtualización es una tecnología que permite ejecutar múltiples sistemas operativos de manera simultánea sobre una misma máquina física, mediante el uso de entornos virtuales aislados. Esta capacidad resulta especialmente útil en contextos de desarrollo, pruebas, formación o administración de sistemas, ya que permite simular distintas configuraciones de hardware y software sin necesidad de múltiples dispositivos físicos.

En el presente trabajo se exploran los principios fundamentales de la virtualización, utilizando como herramienta principal Oracle VirtualBox, una plataforma de virtualización de código abierto ampliamente utilizada por su facilidad de uso y compatibilidad con diversos sistemas operativos.

Como parte del desarrollo práctico, se procede a la creación e instalación de una máquina virtual (VM) que utiliza el sistema operativo Debian, una distribución de Linux reconocida por su estabilidad y seguridad. Dentro de esta máquina virtual se configura un entorno de red y de desarrollo básico, en el cual se realiza la implementación de un script en Python orientado al monitoreo de conectividad de red (a través de comandos como `ping`, resolución de IPs y nombres de host).

Este enfoque permite no solo poner en práctica los conceptos de virtualización, sino también experimentar con la configuración de entornos de red, instalación de servicios como SSH y la interacción remota desde editores como Visual Studio Code, mejorando así la comprensión integral del entorno virtualizado como herramienta de aprendizaje, prueba y desarrollo seguro.

Marco Teórico

Virtualización

La virtualización es una tecnología que permite crear entornos virtuales independientes dentro de un mismo equipo físico. Esto se logra mediante software que simula hardware, permitiendo la ejecución de uno o más sistemas operativos invitados sobre un sistema operativo anfitrión. Entre sus principales ventajas se encuentran el aprovechamiento eficiente de los recursos, la facilidad de administración, la reducción de costos y la capacidad de crear entornos de prueba seguros y aislados. Existen distintos tipos de virtualización, como la de servidores, redes, almacenamiento y escritorios, siendo la más común la de sistemas operativos.

VirtualBox

Oracle VM VirtualBox es un software de virtualización de código abierto que permite ejecutar sistemas operativos invitados sobre diferentes plataformas, como Windows, macOS y Linux. Es ampliamente utilizado por su facilidad de uso, compatibilidad con múltiples sistemas operativos y soporte para funcionalidades avanzadas como carpetas compartidas, captura de pantalla, instantáneas y configuración flexible de red. En este proyecto se eligió VirtualBox por su accesibilidad, buena integración con distintos entornos y capacidad para emular un entorno de red realista mediante configuraciones como el adaptador puente.

Debian

Debian es una distribución del sistema operativo GNU/Linux conocida por su estabilidad, seguridad y fuerte compromiso con el software libre. En entornos de virtualización, Debian es frecuentemente utilizado por su bajo consumo de recursos y su gran comunidad de soporte. Además, su arquitectura modular permite una alta personalización del entorno virtual, lo cual es ideal para prácticas educativas, pruebas de red y administración remota.

Configuración de red: adaptador puente

En VirtualBox, la configuración de red en modo adaptador puente (bridge) permite que la máquina virtual se comporte como un dispositivo más dentro de la red física del host. Esto significa que el sistema operativo invitado recibe una dirección IP de la red local y puede interactuar con otros dispositivos como si fuera una computadora real. Esta configuración es útil para escenarios donde se necesita que la VM tenga acceso directo a internet, a otros equipos en la red o para realizar pruebas de conectividad realistas, como es el caso del presente trabajo.

Herramientas de administración remota: Remote SSH y Visual Studio Code

Visual Studio Code (VSCode) es un editor de código fuente altamente extensible, que a través de la extensión Remote SSH permite conectarse de forma remota a servidores o máquinas virtuales mediante el protocolo SSH (Secure Shell). Esta herramienta facilita el control, monitoreo y edición de archivos directamente sobre la máquina virtual, sin necesidad de interactuar visualmente con ella. En este proyecto, Remote SSH se utilizó para conectarse desde la máquina anfitriona al entorno Debian virtualizado, verificando el estado de los destinos ingresados y gestionando el sistema de forma eficiente y segura.

Integración de tecnologías

La combinación de VirtualBox, Debian, red en modo puente y Remote SSH permite simular un entorno real de red y administración remota. Este enfoque es especialmente útil en entornos educativos o de prueba, donde se requiere replicar el comportamiento de servidores reales, sin comprometer la infraestructura física. La gestión remota desde VSCode aporta una capa de comodidad y eficiencia, alineándose con las prácticas modernas de desarrollo y administración de sistemas.

Caso Práctico

1. Descargar lo necesario:

- Descargar e instalar VirtualBox desde <https://www.virtualbox.org/>
- Descargar ISO de Debian desde <https://www.debian.org/distrib/>

2. Crear máquina virtual en VirtualBox:

- Clic en 'Nueva', nombrar la VM (ej: TUPaD_Python).
- Seleccionar ISO de Debian.
- Asignar: 2-4 GB de RAM, 2 CPUs, y disco de 20-25 GB (VDI dinámico).
- Finalizar y guardar.

3. Configurar red:

- Ir a Configuración > Red. - Cambiar a Adaptador puente para que la VM esté en la red local.

4. Instalar Debian en la VM:

- Seguir asistente de instalación.
- Elegir idioma, zona, teclado, crear usuario y contraseña.
- Usar 'guiado - usar disco completo' para partición.

5. Configurar entorno en Debian (VM):

- Instalar Python y pip: `sudo apt install python3 python3-pip`
`python3-venv -y`
- Instalar servidor SSH: `sudo apt install openssh-server -y sudo`
- `systemctl status ssh` para verificar que esté activo

6. Conectar desde VS Code (máquina anfitriona):



- Instalar extensión Remote - SSH.
- Obtener IP de la VM: `ip a`
- En VS Code: F1 > Remote-SSH: Connect to Host...
- Agregar host: `ssh usuario@IP_VM`
- Conectar e ingresar contraseña.

7. Crear entorno de trabajo en la VM:

- Desde VS Code (conectado a la VM), crear carpeta `network_monitor`.
- Crear archivo `monitor.py` con el código del programa.

8. Crear el programa `monitor.py`

- **Importar módulos necesarios**
 - **subprocess**: para ejecutar comandos del sistema operativo (como `ping`).
 - **socket**: para trabajar con conexiones de red (obtener IPs y nombres de host).
- **Definir función `ping(host)`**
 - Ejecuta el comando `ping` con 3 intentos a la dirección indicada.
 - Devuelve `True` si el ping es exitoso, `False` si falla.
- **Definir función `obtener_ip(host)`**
 - Recibe un dominio o nombre y trata de obtener su dirección IP.
 - Devuelve la IP si la encuentra, o `None` si no puede resolverla.
- **Definir función `obtener_host(ip)`**
 - Recibe una dirección IP y trata de obtener el nombre del host asociado.
 - Devuelve el nombre del host si lo encuentra, o `None` si no.
- **Inicio del programa principal**
 - Ingreso de direcciones IP o dominios por parte del usuario
 - Se pide ingresar direcciones una por una.
 - Se termina el ingreso cuando el usuario deja el campo vacío.
 - Las direcciones se guardan en una lista llamada `hosts`.
 - Procesamiento de cada dirección en la lista `hosts`
 - Para cada dirección:
 - Intenta obtener su IP usando `obtener_ip`.
 - Si la IP es válida:
 - Muestra la IP y trata de obtener el nombre del host.

- Ejecuta **ping** a la IP y muestra si fue exitoso o fallido.
- Si no pudo resolver la IP, muestra un mensaje de error.
- **Mensajes de salida claros**
 - Indican si el ping fue exitoso  o fallido .
 - También muestran si no se pudo resolver la dirección.

9. Probar el programa en la terminal integrada:

- Ejecutar: `python3 monitor.py`
 - Probar con:
 - dominios conocidos ([google.com](https://www.google.com))
 - IPs locales (router)
 - dominios inexistentes.

Metodología utilizada:

Para el desarrollo del caso práctico, se empleó una metodología basada en la virtualización de sistemas y la configuración de entornos controlados, con el objetivo de crear un espacio de trabajo seguro, aislado y fácilmente replicable. El proceso se inició con la instalación y configuración de una máquina virtual utilizando Oracle VirtualBox, sobre la cual se implementó el sistema operativo Debian, conocido por su estabilidad y uso extendido en entornos de servidores y desarrollo.

Una vez instalado el sistema operativo dentro de la máquina virtual, se procedió a configurar las herramientas necesarias para el desarrollo, incluyendo Python, su gestor de paquetes (pip) y el servidor SSH, el cual permite conexiones remotas seguras. Esta preparación del entorno fue esencial para garantizar que todas las pruebas y ejecuciones del programa se llevarán a cabo de forma independiente del sistema anfitrión.

Posteriormente, se estableció una conexión remota mediante la extensión Remote - SSH de Visual Studio Code, lo que facilitó una interacción fluida y eficiente con la máquina virtual desde el equipo principal. Esto permitió editar, ejecutar y depurar el código directamente sobre el entorno virtual, manteniendo al mismo tiempo la comodidad de trabajar desde un entorno gráfico moderno.

Esta metodología no solo favoreció la organización y el aislamiento del entorno de trabajo, sino que también permitió reducir riesgos como conflictos de versiones, errores en la configuración del sistema base o problemas de compatibilidad. Además, al ser un entorno replicable, puede ser fácilmente reutilizado o compartido para fines educativos, de desarrollo o pruebas. En resumen, se trata de un enfoque robusto que promueve buenas prácticas en la gestión de entornos de software.

Resultados Obtenidos:

Al finalizar la práctica, logramos configurar con éxito un entorno virtualizado estable y funcional utilizando Oracle VirtualBox. En dicho entorno se instaló el sistema operativo Debian, sobre el cual se prepararon todas las herramientas necesarias para desarrollar y probar un programa de monitoreo de red. Esta preparación incluyó la instalación de Python, sus dependencias básicas, y la configuración del servidor SSH, elemento clave para permitir el acceso remoto al entorno virtual.

Una vez completada esta configuración, pudimos establecer una conexión remota desde Visual Studio Code utilizando la extensión Remote - SSH, lo que facilitó significativamente el flujo de trabajo. Esta integración ofrece una experiencia de desarrollo muy fluida y similar a la de trabajar localmente, pero con las ventajas que conlleva el aislamiento que proporciona la virtualización.

Esta combinación entre virtualización y acceso remoto no sólo permitió crear un entorno completamente controlado y seguro, sino que también ofreció beneficios concretos, como evitar conflictos entre librerías, minimizar problemas de compatibilidad y poder realizar pruebas sobre distintos sistemas operativos sin comprometer la estabilidad del sistema principal.

En resumen, esta metodología demostró ser una solución muy eficiente tanto para entornos de aprendizaje, donde es necesario experimentar sin temor a errores críticos, como para proyectos de desarrollo profesional, donde la estabilidad, la seguridad y la replicabilidad son fundamentales.

Conclusiones:

La virtualización, implementada mediante VirtualBox, ha demostrado ser una herramienta fundamental para crear entornos de desarrollo aislados y controlados. Este enfoque permite ejecutar sistemas operativos completos de manera independiente sobre una misma máquina física, lo que facilita la gestión de recursos, la configuración personalizada y la replicabilidad del entorno. Al combinarse con la conexión remota a través de Visual Studio Code y su extensión Remote SSH, se optimiza la experiencia de desarrollo, permitiendo trabajar en la máquina virtual como si fuera local, pero con las ventajas de aislamiento y seguridad que ofrece la virtualización. Esta metodología es especialmente valiosa para evitar conflictos entre librerías o versiones, así como para realizar pruebas y experimentos en diferentes sistemas operativos sin riesgo para el sistema anfitrión, contribuyendo significativamente a la eficiencia y seguridad en los procesos de desarrollo y pruebas.

Complicaciones que pudieran surgir

1. **Problemas de rendimiento:** Dependiendo del hardware del equipo anfitrión, la ejecución de una máquina virtual puede ser lenta, especialmente si se asignan pocos recursos o si se ejecutan múltiples procesos en paralelo.
2. **Configuración de red:** Configurar correctamente el adaptador puente en VirtualBox para que la máquina virtual esté en la misma red local puede generar confusión si no se tiene experiencia previa.
3. **Errores durante la instalación del sistema operativo o paquetes:** Pueden presentarse inconvenientes al instalar Debian o servicios como SSH y Python si no se cuenta con conectividad adecuada o si faltan repositorios.
4. **Conexión remota desde VS Code:** En ocasiones, la extensión Remote - SSH puede fallar si hay errores en la configuración del archivo `~/.ssh/config` o si la IP de la VM cambia.

Ventajas del enfoque utilizado

1. **Aislamiento del entorno:** Permite realizar pruebas sin poner en riesgo el sistema operativo principal.
2. **Facilidad para replicar entornos:** Una vez configurada la VM, puede exportarse y utilizarse en otros equipos sin necesidad de repetir el proceso desde cero.
3. **Aprendizaje práctico:** Favorece la comprensión del funcionamiento de sistemas operativos, servicios y redes.
4. **Flexibilidad:** Es posible crear distintos entornos con diferentes configuraciones para simular escenarios reales de manera controlada.

Desventajas del enfoque

1. **Requiere recursos de hardware:** La virtualización consume memoria RAM, CPU y espacio en disco, lo que puede afectar el rendimiento del equipo anfitrión si no tiene buenas especificaciones.
2. **Curva de aprendizaje inicial:** La configuración de VMs, redes, servicios y conexión remota puede ser compleja para usuarios sin experiencia.
3. **Dependencia de herramientas externas:** El desarrollo depende del buen funcionamiento de VirtualBox, VS Code y sus extensiones.

Posibles mejoras

1. **Uso de snapshots o imágenes base:** Para acelerar el proceso, se podrían crear imágenes de VM ya configuradas que se puedan clonar fácilmente para nuevos proyectos.
2. **Automatización del entorno:** Usar herramientas del tipo *Infraestructura como Software* (como Ansible) para automatizar la creación y configuración de máquinas virtuales, facilitando la reproducibilidad y reduciendo errores manuales.

3. **Virtualización más eficiente:** Evaluar el uso de soluciones más ligeras o específicas como *contenedores* (como Docker o Podman en casos donde no se necesita un sistema completo) o *hipervisores de Tipo 1* (como Proxmox) para gestión más avanzada de entornos virtuales.
4. **Mejor gestión de red:** Utilizar scripts para configurar automáticamente adaptadores de red y evitar problemas de conectividad.

Bibliografía

Debian Project. (s.f.). *Debian Documentation*. Recuperado el 3 de junio de 2025, de <https://www.debian.org/doc/>

Microsoft. (s.f.). *Remote Development using SSH*. Visual Studio Code Documentation. Recuperado el 3 de junio de 2025, de <https://code.visualstudio.com/docs/remote/ssh>

Microsoft Learn. (s.f.). *Visual Studio Code Documentation*. Recuperado el 3 de junio de 2025, de <https://learn.microsoft.com/en-us/visualstudio/code/>

Oracle. (s.f.). *VirtualBox User Manual*. Recuperado el 3 de junio de 2025, de <https://www.virtualbox.org/manual/UserManual.html>

Oracle. (s.f.). *VirtualBox Networking Overview*. Recuperado el 3 de junio de 2025, de <https://www.virtualbox.org/manual/ch06.html>

Red Hat. (s.f.). *Introduction to virtualization*. Recuperado el 3 de junio de 2025, de <https://www.redhat.com/en/topics/virtualization>

ArchWiki. (s.f.). *VirtualBox networking*. Recuperado el 3 de junio de 2025, de <https://wiki.archlinux.org/title/VirtualBox>

Python Software Foundation. (n.d.). 37.1. *socket — Low-level networking interface*. Python 3.10.13 Documentation. Recuperado el 4 de junio de 2025, de <https://docs.python.org/es/3.10/library/socket.html>

Anexos

Link del repositorio en GitHub :

<https://github.com/DaniDantur/integrador-ayso-cornejo-dantur-com12>

Link del video explicativo en YouTube:

https://www.youtube.com/watch?v=Mrz5zX_saug