**Trabajo Integrador Arquitectura y Sistemas Operativos**

**Virtualización**

**Alumnos:**

* Giardini Silvia
* Cordero Marina

**Materia:**

**Arquitectura y Sistemas Operativos**

**Profesor:**

Mauricio Gabriel Pasti

**Tutor:**

Jonathan Zarate

**Fecha de entrega:**

de junio de 2025

**Índice**

1. Introducción
2. Marco Teórico
3. Caso Práctico
4. Metodología Utilizada
5. Resultados Obtenidos
6. Conclusiones
7. Bibliografía
8. Anexos

**Introducción**

La **virtualización** es un pilar esencial en la informática moderna. Permite crear entornos aislados, como **máquinas virtuales** y **contenedores**, optimizando el uso de recursos y facilitando el desarrollo, despliegue y mantenimiento de aplicaciones. Seleccionamos este tema por su relevancia en la implementación de soluciones tecnológicas eficientes: la virtualización posibilita ejecutar múltiples sistemas operativos o aplicaciones en un mismo hardware físico, mejorando la **eficiencia**, **escalabilidad** y **flexibilidad**. Las máquinas virtuales, que emulan sistemas operativos completos, y los contenedores, que ofrecen entornos ligeros y portátiles, son fundamentales para gestionar eficazmente entornos de desarrollo y producción.

Para un técnico en programación, comprender la virtualización es crucial. Nos permite configurar y administrar estos entornos usando herramientas como **VMware**, **VirtualBox**, **Hyper-V** para máquinas virtuales robustas, y **Docker** para la creación ágil de contenedores.

El objetivo de este trabajo es explorar los conceptos fundamentales de la virtualización, implementando una máquina virtual que ejecute un programa Python simple. Así, analizaremos su funcionamiento para comprender las ventajas y desafíos de estas tecnologías en un entorno práctico.

**Marco Teórico**

La **virtualización** es una tecnología que permite crear **representaciones abstractas** de recursos físicos o lógicos, como **hardware**, **sistemas operativos** o **aplicaciones**, para ejecutar múltiples entornos aislados en un solo sistema físico. Según Tanenbaum y Bos (2015), la virtualización abstrae los recursos físicos de un ordenador, permitiendo que múltiples sistemas operativos o aplicaciones se ejecuten de manera independiente en el mismo hardware.

**Máquinas Virtuales (VM)**

Las **Máquinas Virtuales (VM)** son entornos de computación que emulan un sistema informático completo, replicando tanto el hardware virtualizado (como CPU, memoria, almacenamiento y red) como un sistema operativo completo. Este enfoque permite ejecutar múltiples sistemas operativos de manera independiente en un solo hardware físico, cada uno en su propia VM, lo que las hace ideales para casos como pruebas de software, entornos de desarrollo, aislamiento de aplicaciones o ejecución de sistemas legacy (o sistemas heredados).

Según VMware, una empresa líder en tecnologías de virtualización, las máquinas virtuales se definen como:

"Una máquina virtual (VM) es una representación digital de un sistema físico, que ejecuta un sistema operativo completo y aplicaciones como una computadora real. Un hipervisor permite que múltiples VMs se ejecuten en un solo host físico, proporcionando aislamiento y optimización de recursos."

**Características y funcionamiento:**

1. **Emulación de hardware**: Las VMs simulan componentes físicos completos, lo que permite que cada máquina virtual funcione como si fuera un ordenador independiente. Esto incluye procesadores virtuales, discos duros, interfaces de red, etc.
2. **Sistema operativo completo**: Cada VM ejecuta su propio sistema operativo (por ejemplo, Windows, Linux, etc.), lo que proporciona un entorno totalmente funcional, pero también implica un mayor uso de recursos, ya que el sistema operativo consume memoria, CPU y almacenamiento.
3. **Hipervisor**: Las VMs son gestionadas por un hipervisor (como VMware ESXi, Microsoft Hyper-V, Oracle VirtualBox o KVM), que actúa como una capa intermedia entre el hardware físico y las máquinas virtuales. Hay dos tipos principales de hipervisores:
   * **Tipo 1 (nativos o bare-metal)**: Se ejecutan directamente sobre el hardware físico, ofreciendo mejor rendimiento y eficiencia (ejemplo: VMware ESXi, XenServer).
   * **Tipo 2 (hospedados)**: Se ejecutan sobre un sistema operativo anfitrión, siendo más fáciles de configurar, pero menos eficientes (ejemplo: VirtualBox, VMware Workstation).
4. **Aislamiento robusto**: Cada VM opera de forma aislada, lo que significa que un fallo o compromiso en una VM no afecta a otras VMs ni al sistema anfitrión. Esto las hace ideales para entornos que requieren alta seguridad o separación de cargas de trabajo.

**Ventajas:**

* **Flexibilidad**: Permiten ejecutar diferentes sistemas operativos (por ejemplo, Windows y Linux) en el mismo hardware físico.
* **Aislamiento**: Proporcionan un entorno seguro y aislado, útil para pruebas, desarrollo o ejecución de aplicaciones en entornos controlados.
* **Portabilidad**: Las VMs pueden trasladarse entre diferentes hipervisores o servidores con facilidad, siempre que sean compatibles.
* **Snapshots y backups**: Se pueden crear instantáneas (snapshots) para guardar el estado de una VM en un momento dado, facilitando la recuperación ante fallos o cambios no deseados.
* **Compatibilidad**: Ideales para sistemas legacy o aplicaciones que requieren entornos específicos.

**Desventajas:**

* **Consumo de recursos**: La emulación de hardware y la ejecución de un sistema operativo completo por cada VM generan un mayor uso de CPU, memoria y almacenamiento en comparación con otras tecnologías como contenedores.
* **Rendimiento**: El overhead del hipervisor y la emulación de hardware pueden reducir el rendimiento en comparación con sistemas nativos o contenedores.
* **Tiempo de arranque**: Las VMs requieren iniciar un sistema operativo completo, lo que puede ser más lento que otras soluciones ligeras.
* **Gestión compleja**: Configurar y mantener múltiples VMs puede ser más complicado, especialmente en entornos con muchas máquinas virtuales.

**Casos de uso:**

* **Entornos de prueba y desarrollo**: Los desarrolladores pueden probar aplicaciones en diferentes sistemas operativos sin necesidad de hardware dedicado.
* **Servidores consolidados**: En centros de datos, las VMs permiten ejecutar múltiples aplicaciones en un solo servidor físico, optimizando recursos.
* **Seguridad**: Aislar aplicaciones críticas o sospechosas en VMs para minimizar riesgos.
* **Migración de sistemas**: Facilitan la migración de servidores físicos a entornos virtualizados o a la nube.
* **Recuperación ante desastres**: Las VMs pueden replicarse o respaldarse fácilmente para garantizar la continuidad del negocio.

**Comparación con contenedores:**

A diferencia de los contenedores (como Docker), que comparten el kernel del sistema operativo anfitrión y son más ligeros, las VMs incluyen un sistema operativo completo y emulan hardware, lo que las hace más pesadas pero también más aisladas y versátiles para ciertos casos. Mientras que los contenedores son ideales para aplicaciones modernas y escalables, las VMs son preferibles cuando se requiere un aislamiento completo o compatibilidad con sistemas operativos distintos.

En resumen, las máquinas virtuales ofrecen una solución robusta y flexible para virtualización, pero su mayor consumo de recursos las hace menos eficientes en comparación con tecnologías más ligeras como los contenedores. Su elección depende de las necesidades específicas de aislamiento, compatibilidad y rendimiento del entorno.

**Contenedores**

Los **contenedores** son entornos virtualizados ligeros que permiten **empaquetar** y **ejecutar** aplicaciones junto con sus dependencias (bibliotecas, configuraciones, herramientas específicas) de manera aislada, compartiendo el kernel del sistema operativo anfitrión.

A diferencia de las máquinas virtuales, que incluyen un sistema operativo completo y consumen más recursos, los contenedores son mucho **más eficientes en términos de memoria, CPU y almacenamiento, ya que no duplican el sistema operativo base**. Esto los hace ideales para entornos donde se requiere alta portabilidad, escalabilidad y rapidez en el despliegue.

[](https://www.docker.com/resources/what-container/)Según la documentación oficial de Docker, un contenedor es “una unidad estándar de software que empaqueta el código y todas sus dependencias, de modo que la aplicación se ejecuta de manera rápida y confiable en diferentes entornos de cómputo. Una imagen de contenedor de Docker es un paquete de software ligero, independiente y ejecutable que incluye todo lo necesario para ejecutar una aplicación: código, runtime, herramientas del sistema, bibliotecas y configuraciones”.

Los contenedores se basan en tecnologías como namespaces y cgroups en Linux, que garantizan el aislamiento de procesos y la gestión eficiente de recursos. Esto permite que múltiples contenedores se ejecuten simultáneamente en el mismo host sin conflictos, compartiendo recursos de manera óptima. Herramientas como **Docker** y **Kubernetes** han popularizado su uso al simplificar la creación, gestión y orquestación de contenedores, facilitando su adopción en entornos de desarrollo, pruebas y producción.

Los contenedores son especialmente valiosos en arquitecturas de microservicios, donde cada componente de una aplicación puede ejecutarse en un contenedor independiente, lo que mejora la modularidad y la facilidad de actualización. Por ejemplo, un contenedor puede alojar una aplicación web, mientras otro maneja la base de datos, y ambos se comunican sin necesidad de entornos monolíticos complejos. Su portabilidad asegura que una aplicación funcione de manera consistente en diferentes entornos, desde un portátil de desarrollo hasta un clúster en la nube, siempre que el motor de contenedores (como Docker) esté presente.

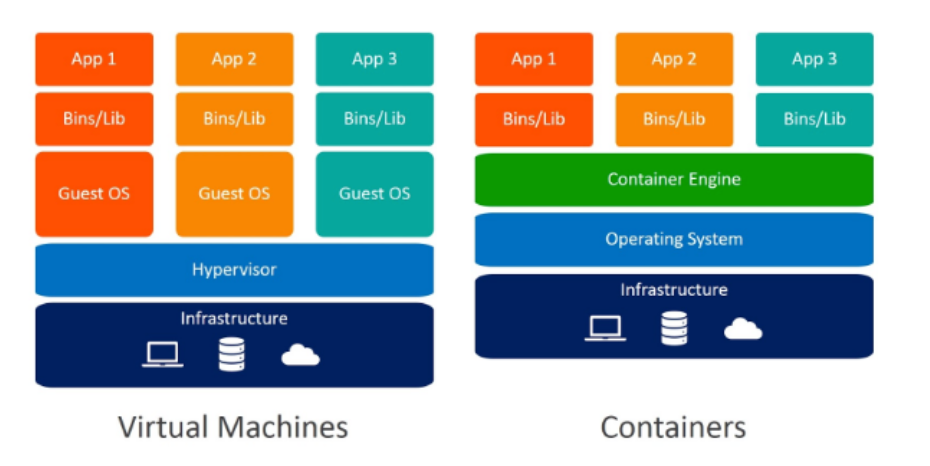
En términos prácticos, los contenedores ofrecen ventajas como:

* **Eficiencia de recursos**: Al compartir el kernel, consumen menos memoria y CPU que las máquinas virtuales.
* **Portabilidad**: Los contenedores pueden ejecutarse en cualquier sistema compatible con el motor de contenedores, independientemente del entorno subyacente.
* **Escalabilidad**: Herramientas como Kubernetes permiten escalar contenedores horizontalmente (añadiendo más instancias) o verticalmente (asignando más recursos) con facilidad.
* **Despliegue rápido**: Los contenedores se inician en segundos, a diferencia de las máquinas virtuales, que requieren minutos.
* **Ecosistema robusto**: Imágenes preconfiguradas en repositorios como **Docker Hub** facilitan el acceso a aplicaciones y servicios listos para usar.

En resumen, los contenedores representan una solución eficiente y moderna para el desarrollo y despliegue de aplicaciones, siendo una tecnología clave en la informática actual, especialmente en entornos de nube y DevOps.

**Estructura de Docker:**

* **Imágenes:** Archivos inmutables que contienen todo lo necesario para ejecutar una aplicación (código, dependencias, configuraciones). Las imágenes se crean a partir de **Dockerfiles** y se almacenan en registros como **Docker Hub**.
* **Contenedores:** Instancias ejecutables de una imagen, que se ejecutan de forma aislada pero comparten el kernel del sistema operativo anfitrión.
* **Dockerfiles:** Archivos de texto con instrucciones para construir imágenes de Docker. Contienen comandos como FROM, RUN, COPY, y ENTRYPOINT para definir el entorno y comportamiento de la imagen.



**Diferencias clave:**

***VMs****:* Mayor aislamiento, pero mayor consumo de recursos (memoria, CPU). Cada VM ejecuta un sistema operativo completo.

***Contenedores****:* Menor aislamiento (comparten el kernel), pero más ligeros y rápidos. Ideales para microservicios.

**Comparación entre VMs y Contenedores**

|  |  |  |
| --- | --- | --- |
| **Característica** | **Máquinas**  **Virtuales** | **Contenedores** |
| **Aislamiento** | Alto (hardware y SO  completos) | Moderado (comparte  kernel) |
| **Uso de recursos** | Alto (SO completo  por VM) | Bajo (sin SO  completo) |
| **Tiempo de inicio** | Minutos | Segundos |
| **Portabilidad** | Moderada (depende  del hipervisor) | Alta (estándar  Docker) |
| **E**jemplo | VMware, VirtualBox | Docker, Podman |

**Diagrama Comparativo Referencias**

* + Tanenbaum, A. S., & Bos, H. (2015). *Modern Operating Systems*. Pearson.
  + Docker Inc. (2025). Documentación oficial de Docker. Recuperado el 30/05/2025 de https://docs.docker.com.

**Caso Practico:**

💻 Aquí se debe presentar un problema o situación concreta que haya sido desarrollada o simulada para aplicar el contenido del trabajo.

# Incluye:

* + Breve descripción del problema a resolver.
  + Capturas de pantalla si corresponde.
  + Validación del funcionamiento.

# Metodología Utilizada

 Describe los pasos seguidos durante el desarrollo del trabajo. Se pueden incluir:

* + Investigación previa (fuentes utilizadas).
  + Etapas de diseño y prueba del código.
  + Herramientas y recursos utilizados (IDE, librerías, control de versiones, etc.).
  + Trabajo colaborativo (reparto de tareas en el grupo de trabajo).

# Resultados Obtenidos

📊 Detalla qué se logró con el caso práctico, qué aspectos funcionaron correctamente y qué dificultades se presentaron.

Se pueden incluir:

* + Casos de prueba realizados.
  + Errores corregidos.

# Conclusiones

🎯 Reflexión final del grupo de trabajo. Aquí se recomienda incluir:

* + Qué se aprendió al hacer el trabajo.
  + Posibles mejoras o extensiones futuras.
  + Dificultades que surgieron si las hubo y cómo se resolvieron.

# Bibliografía

<https://www.vmware.com/topics/glossary/content/virtual-machine.html> , consultada el 30 de mayo de 2025)

sitio web oficial de Docker:

<https://docs.docker.com>

<https://docs.docker.com/get-started/overview/>

Tanenbaum, A. S., & Bos, H. (2015). *Modern Operating Systems*. Pearson.

Docker Inc. (2025). Documentación oficial de Docker.

📚 Listado de fuentes consultadas, utilizando normas básicas APA u otro formato consistente. Se sugiere incluir:

* + Libros.
  + Documentación oficial.
  + Sitios web con fecha de acceso.
  + Artículos o materiales recomendados por el docente. Ejemplo:
  + Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). Operating System Concepts. John Wiley & Sons. Capítulo 3, "Procesos".
  + Stallings, W. (2023). Operating Systems: Internals and Design Principles.

# 📥 Recomendaciones para la presentación

* + Formato del archivo: **.pdf**
  + Tipografía: Arial o Calibri, tamaño 11 o 12
  + Interlineado: 1,5
  + Márgenes estándar (2.5 cm)
  + Portada opcional con el logo de la institución (si se requiere)
  + Entrega digital mediante plataforma institucional o por correo