PROMETEO

Unidad 7:
Virtualización, contenedores y tendencias

Sistemas informáticos

Técnico Superior de DAM / DAW

Sesión 24 – Comparación: máquinas virtuales vs contenedores

Fundamentos

Cuando trabajas en entornos de sistemas o despliegue de aplicaciones, uno de los primeros dilemas que aparece es elegir entre máquinas virtuales (VMs) y contenedores. Ambas tecnologías permiten ejecutar aplicaciones de forma aislada, pero lo hacen de maneras muy diferentes, y eso condiciona su eficiencia, su seguridad y el tipo de proyectos para los que son más adecuadas.

Máquinas Virtuales

Una máquina virtual replica un ordenador completo dentro de otro. Esto significa que cada VM tiene su propio sistema operativo, su kernel y su conjunto de librerías y servicios. Este nivel de aislamiento te permite ejecutar sistemas operativos completamente distintos en un mismo servidor físico: Linux, Windows, BSD… todos pueden convivir sin interferirse.

Contenedores

Los contenedores, en cambio, siguen una filosofía de ligereza. No replican un sistema operativo entero, sino que utilizan el kernel del sistema anfitrión. Todo lo que está dentro del contenedor —aplicación, dependencias, configuración— está empaquetado en una unidad portable, pero sin duplicar la infraestructura del sistema operativo.

Por eso las VMs destacan por la seguridad y por garantizar compatibilidad absoluta. En la práctica, son una capa de separación muy sólida entre cargas de trabajo, algo esencial en sectores como banca, análisis forense, ciberseguridad o entornos multicliente.

Como consecuencia, arrancan en milisegundos, consumen muchos menos recursos y permiten escalar aplicaciones de forma masiva sin necesidad de grandes servidores.

Tecnologías complementarias

Ambas tecnologías no compiten realmente; cubren necesidades distintas. Las VMs proporcionan aislamiento profundo y flexibilidad de sistemas, mientras que los contenedores permiten ejecutar aplicaciones modernas —sobre todo microservicios— de forma rápida, portable y altamente automatizable.

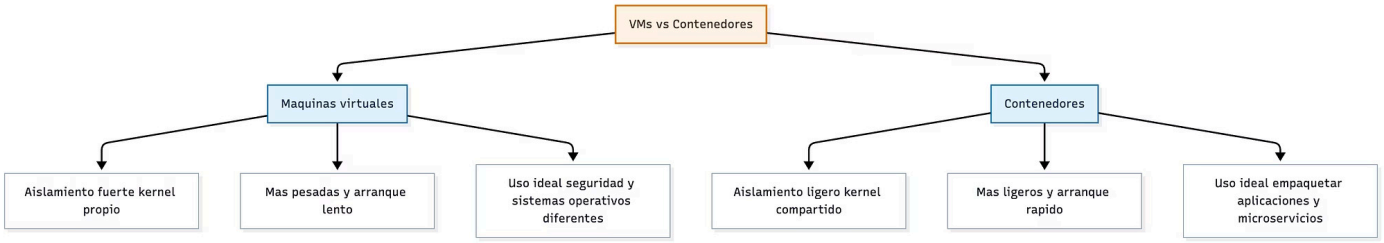
Tendencia actual

Hoy en día, la tendencia en empresas es combinar ambos mundos: usar máquinas virtuales como base estable y segura, y desplegar contenedores dentro de esas VMs para obtener eficiencia operativa, despliegues continuos y escalabilidad en la nube.

Es un modelo que encontrarás en prácticamente todos los proveedores cloud (AWS, Azure, Google Cloud) y en la mayoría de organizaciones que operan aplicaciones críticas. Comprender estas diferencias desde una perspectiva profesional te permite seleccionar la tecnología adecuada para cada caso de uso, optimizar costes y mejorar la estabilidad de los sistemas que gestionas.

Esquema Visual

A continuación tienes un diagrama en formato Mermaid, seguido de una descripción detallada para que puedas recrearlo sin ambigüedades.



Descripción del esquema

01

Nodo central

"VMs vs Contenedores" representa la comparación general entre ambas tecnologías. Desde él salen dos ramas principales:

02

Máquinas Virtuales

- Aislamiento Fuerte (Kernel Propio) indica que cada VM ejecuta su propio sistema operativo.
- Más Pesadas – Arranque Lento resume el costo de recursos y el tiempo de inicio.
- Ideal para: Seguridad y SO diferentes muestra los principales casos de uso corporativos.

03

Contenedores

- Aislamiento Ligero (Kernel Compartido) indica que comparten kernel con el host.
- Más Ligeros – Arranque Rápido destaca su eficiencia.
- Ideal para: Empaquetar Apps y Microservicios muestra su papel en el desarrollo moderno.

04

Conclusión visual

El esquema visualiza claramente por qué ambas tecnologías se complementan.

Caso de Estudio – Kubernetes

Para entender el impacto real de los contenedores en la industria, conviene observar cómo se gestionan cuando pasas de un puñado de contenedores a cientos o miles. Ahí entra en juego Kubernetes, una plataforma de orquestación diseñada originalmente por Google para administrar la enorme escala de sus servicios internos.

Contexto

Google necesitaba un sistema que automatizara tareas como:

- desplegar contenedores en cientos de servidores,
- reiniciar servicios automáticamente si fallaban,
- escalar recursos según la demanda,
- actualizar aplicaciones sin tiempo de inactividad,
- distribuir carga entre nodos,
- detectar fallos y mover contenedores automáticamente.

De este problema interno nació Kubernetes, posteriormente liberado como software de código abierto. Hoy es el estándar para ejecutar aplicaciones contenerizadas en la nube.

Empresas como Spotify, Uber, ING, BBVA o BMW no pueden gestionar manualmente la cantidad de microservicios que ejecutan. Necesitan automatización completa, tolerancia a fallos y escalado dinámico.

Estrategia

Implementan Kubernetes como plataforma de orquestación. Docker u otras herramientas se encargan de crear los contenedores, pero Kubernetes:

- los reparte entre nodos del clúster,
- monitoriza su estado,
- crea copias adicionales según el tráfico,
- actualiza versiones sin interrumpir servicios (rolling updates),
- gestiona la red interna de contenedores.

Resultado

El impacto es enorme:

- Reducción drástica de tiempos de despliegue.
- Escalabilidad automática para soportar picos de demanda.
- Alta disponibilidad garantizada por diseño.
- Menos errores humanos gracias a la automatización.
- Estandarización del ciclo de desarrollo y despliegue.

Kubernetes no compite con Docker; lo complementa para operar servicios en producción. Por eso prácticamente todos los proveedores cloud ofrecen Kubernetes como servicio gestionado (AKS, EKS, GKE).

Herramientas y Consejos

Usa máquinas virtuales como base y contenedores dentro

Es una práctica extendida en empresas tecnológicas:

- VM = aislamiento y seguridad
- Contenedor = rapidez y escalabilidad

Esto reduce riesgos y simplifica la portabilidad entre nubes.

K3s para aprender Kubernetes

Si Kubernetes te parece complejo, K3s es una versión ligera pensada para entornos pequeños o educativos. Se instala en minutos y consume muy pocos recursos.

Podman como alternativa segura a Docker

Podman es compatible con la sintaxis de Docker pero no necesita un demonio ejecutándose como root, lo que reduce la superficie de ataque en entornos de producción.

Automatiza despliegues y pruebas

Herramientas como:

- GitHub Actions,
- GitLab CI,
- ArgoCD,
- Helm

permiten automatizar desde pruebas hasta despliegues en Kubernetes.

Monitoriza contenedores en producción

Prometheus + Grafana se han convertido en el estándar para observar métricas, alertas y rendimiento de contenedores y nodos.

Mitos y Realidades

❌ Mito 1: "Los contenedores reemplazarán a las máquinas virtuales."

→ FALSO. Los contenedores no sustituyen a las VMs. Ofrecen un aislamiento más ligero y se enfocan en la ejecución eficiente de aplicaciones. Las VMs continúan siendo esenciales para ejecutar sistemas operativos distintos, garantizar seguridad fuerte y proporcionar entornos aislados para clientes y auditorías. En la práctica, ambas tecnologías suelen coexistir dentro de la misma arquitectura híbrida.

❌ Mito 2: "Solo puedo ejecutar aplicaciones de Linux en contenedores Docker."

→ FALSO. Aunque los contenedores Linux son los más comunes, Microsoft lleva años incorporando contenedores nativos en Windows Server. Es posible ejecutar aplicaciones Windows en contenedores Windows, lo que permite modernizar aplicaciones heredadas sin migrarlas a Linux.

Resumen Final

- VMs: aislamiento completo, kernel propio, más pesadas, ideales para seguridad y compatibilidad.
- Contenedores: kernel compartido, muy ligeros, rápidos y perfectos para microservicios.
- Tecnologías complementarias; suelen usarse juntas.
- Kubernetes es el estándar para orquestar contenedores a gran escala.

Sesión 25 – Monitorización (Nagios, Zabbix) y seguridad del sistema (logs, hardening, actualizaciones) + introducción a DevOps

Cuando gestionas sistemas en producción, una de las claves para asegurar su continuidad es comprender que instalarlos no es suficiente. El verdadero trabajo empieza después: mantenerlos funcionando de forma estable, segura y predecible. Aquí entran en juego tres pilares fundamentales: monitorización, seguridad del sistema y actualización continua, todos integrados bajo la filosofía DevOps.

La monitorización

La monitorización consiste en vigilar el estado de los servidores, aplicaciones y servicios en tiempo real. Herramientas como Nagios y Zabbix detectan anomalías antes de que los usuarios se den cuenta.

Imagina un servidor que empieza a quedarse sin espacio en disco o un servicio web que responde más lentamente de lo habitual. Sin monitorización, te enterarías cuando el sistema ya ha caído; con ella, recibes alertas anticipadas y puedes actuar antes de que el problema se convierta en una interrupción del servicio.

La seguridad del sistema

La seguridad del sistema, por su parte, se construye sobre tres prácticas constantes:

- Revisión de logs, para detectar comportamientos sospechosos o errores recurrentes.
- Hardening, que consiste en reducir la superficie de ataque deshabilitando servicios innecesarios y aplicando configuraciones seguras.
- Actualizaciones periódicas, para corregir vulnerabilidades que podrían comprometer la seguridad del sistema.

En entornos profesionales, la monitorización es crítica porque permite anticiparse, no simplemente reaccionar.

Estas prácticas no son opcionales: forman parte del mantenimiento básico de cualquier infraestructura profesional.

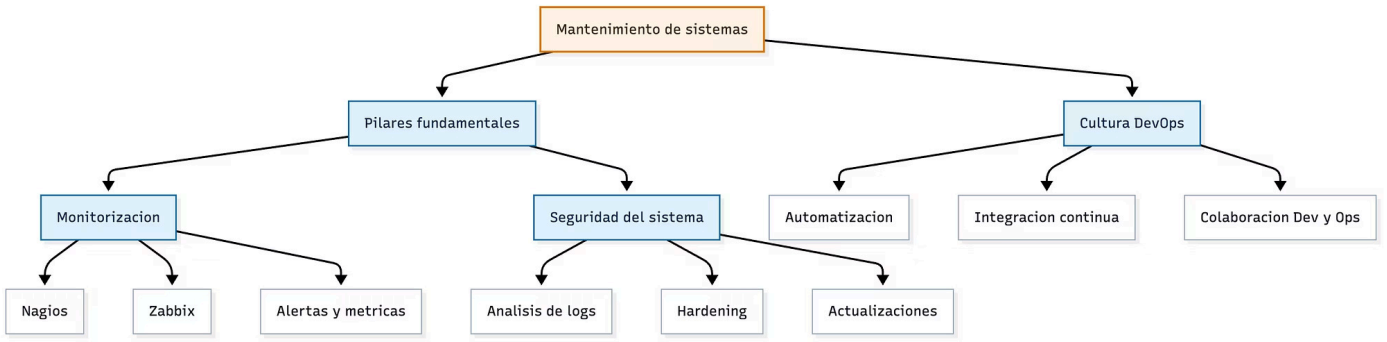
DevOps: uniendo desarrollo y operaciones

Finalmente, la filosofía DevOps une estos dos mundos: desarrollo (Dev) y operaciones (Ops). Su objetivo es que los equipos trabajen juntos para automatizar tareas, reducir errores humanos y acelerar la entrega de software. DevOps introduce prácticas como la integración continua, los despliegues automatizados y la infraestructura como código.

En la práctica, DevOps no es una herramienta ni un rol único, sino una cultura que transforma la forma en que se desarrolla, despliega y mantiene software. La combinación de monitorización, seguridad y DevOps forma el núcleo operativo de cualquier organización moderna. Incluso en empresas pequeñas, estas prácticas aumentan la estabilidad y reducen el riesgo. Y en grandes empresas —desde bancos hasta plataformas de streaming— son indispensables para garantizar disponibilidad 24/7.

Comprender estos pilares te permite actuar con criterio en entornos reales: detectar anomalías, endurecer sistemas vulnerables, automatizar procesos repetitivos y responder a incidentes de forma eficaz. Lo que aprendes en esta sesión es el tipo de conocimiento que diferencia a un técnico capaz de instalar sistemas de uno capaz de mantenerlos en producción.

Esquema Visual



Descripción detallada del diagrama

Mantenimiento de Sistemas

Es el nodo principal, representando la actividad continua de operar entornos de TI.

Desde él salen dos ramas:

Pilares Fundamentales

Divididos en:

Monitorización, que contiene tres nodos secundarios:

- Nagios: herramienta veterana de supervisión.
- Zabbix: plataforma moderna con interfaz visual avanzada.
- Alertas y Métricas: el objetivo central del monitoreo.

Seguridad del Sistema

Con:

- Análisis de Logs: revisión de eventos y auditoría.
- Hardening: reducción de servicios y configuración segura.
- Actualizaciones: aplicación de parches de seguridad.

Cultura DevOps

Se despliega en:

- Automatización: infraestructura como código, scripts, pipelines.
- Integración Continua: pruebas automáticas antes de desplegar.
- Colaboración Dev + Ops: eliminación de silos entre equipos.

Este esquema muestra cómo la monitorización, la seguridad y DevOps se relacionan entre sí para mantener sistemas estables y confiables.

Caso de Estudio – Nagios en empresas de hosting

Contexto

Las empresas de hosting, como OVHcloud o Hetzner, gestionan miles de servidores físicos y virtuales. Cualquier interrupción puede provocar fallos en aplicaciones de clientes, pérdida de datos o incumplimiento de acuerdos de disponibilidad (SLAs). Antes de adoptar herramientas modernas como Prometheus o Zabbix, la mayoría de estas empresas usaban —y muchas siguen usando— Nagios como corazón de su sistema de monitorización.

En un entorno con miles de servidores, los problemas se multiplican: discos que fallan, servicios que se detienen, picos de CPU inesperados, caídas de red… Sin un sistema de monitorización centralizado, el personal de soporte solo reaccionaría cuando un cliente reportara un fallo, lo que generaría desconfianza y pérdidas económicas.

Estrategia

Hetzner implementó un sistema completo de monitorización basado en Nagios, ampliado mediante cientos de plugins personalizados. Cada servidor reporta variables como:

- uso de CPU, RAM y disco,
- temperatura del hardware,
- disponibilidad de servicios como HTTP, MySQL, SSH,
- tiempo de respuesta de aplicaciones,
- estado de ventiladores y fuentes de alimentación.

Además, configuraron umbrales críticos y advertencias. Por ejemplo:

- disco > 90% = advertencia,
- disco > 95% = alerta crítica.

Notificaciones automáticas

Cuando se detecta un fallo, Nagios envía notificaciones automáticamente al equipo de guardia mediante:

- SMS,
- email,
- sistemas de mensajería como Slack o Mattermost.

Resultado

La implementación permitió:

- detectar fallos antes de que los clientes se vieran afectados,
- reducir los tiempos de respuesta a incidencias,
- automatizar acciones correctivas (reinicios controlados, limpieza de logs),
- mejorar la satisfacción de los clientes,
- disminuir penalizaciones por SLA incumplidos.

Gracias a esta monitorización proactiva, la empresa consiguió aumentar su tiempo de disponibilidad por encima del 99,9 %, un valor crítico para cualquier plataforma de servicios en la nube. Aunque muchas compañías han migrado a soluciones más modernas, Nagios sigue en funcionamiento por su flexibilidad y su fiabilidad probada.

Herramientas y Consejos

Centraliza los logs para ganar visibilidad

En entornos profesionales, los logs no se revisan uno por uno en cada servidor. Se agrupan en plataformas como:

- ELK Stack (Elasticsearch + Logstash + Kibana)
- Graylog
- Splunk (enterprise)

Estas herramientas permiten buscar, filtrar y visualizar eventos de forma rápida. En una investigación de seguridad o en un análisis de rendimiento, esta visibilidad es clave.

Usa Ansible para aplicar hardening de forma consistente

Configurar la seguridad manualmente servidor a servidor no es viable. Con Ansible puedes:

- desactivar servicios innecesarios,
- aplicar reglas de firewall,
- configurar SSH seguro,
- actualizar paquetes,
- definir parámetros de seguridad del kernel.

Todo con un solo comando. Es una de las herramientas más utilizadas en equipos DevOps.

Zabbix para monitorización moderna

Zabbix ofrece una interfaz clara, autodetección de servicios, plantillas ya diseñadas para equipos de red y servidores, integración con Grafana y una comunidad muy activa. Es excelente para:

- redes corporativas,
- centros de datos,
- entornos multicliente.

Automatiza actualizaciones y parches

En sistemas Linux, puedes apoyarte en:

- Unattended Upgrades (Debian/Ubuntu),
- dnf-automatic (Fedora/CentOS),
- WSUS en entornos Windows.

La automatización reduce la vulnerabilidad ante ataques recientes.

Trabaja con pipelines DevOps desde el inicio

Empieza a practicar con:

- GitHub Actions,
- GitLab CI/CD,
- Jenkins,
- Azure DevOps.

Automatizar pruebas y despliegues acelera enormemente el trabajo real en empresas.

Mitos y Realidades

❌ Mito 1: "Si el sistema funciona, no hace falta monitorizarlo."

→ FALSO. La monitorización no se usa para saber si algo ha fallado, sino para saber antes de que falle. Sistemas que quedan sin disco, procesos que empiezan a consumir más memoria o equipos que se sobrecalientan pueden anticipar una caída total. Monitorizar es ser proactivo.

❌ Mito 2: "DevOps es solo para grandes empresas como Google o Netflix."

→ FALSO. DevOps es una cultura aplicable a cualquier tamaño de equipo. Gracias a herramientas accesibles como GitHub Actions, Docker o Ansible, cualquier estudiante o pequeña empresa puede adoptar prácticas DevOps sin inversiones enormes.

Resumen Final

- La monitorización (Nagios, Zabbix) permite anticiparse a fallos.
- La seguridad incluye análisis de logs, hardening y actualizaciones.
- DevOps une desarrollo y operaciones con automatización y colaboración.
- Herramientas clave: ELK, Ansible, Zabbix, GitHub Actions.
- La clave profesional: prevención, automatización y visibilidad constante.