# Conceptos básicos de sistemas de servidores

La base de cualquier servidor web comienza con una comprensión sólida de los sistemas que lo sustentan. En esta sección, exploraremos los sistemas operativos soportados y cómo los avances recientes en distribuciones Linux, Windows Server 2022 y entornos cloud-native han mejorado la eficiencia y escalabilidad de los servidores web. También abordaremos los fundamentos del protocolo TCP/IP, piedra angular de las comunicaciones en red, y el modelo cliente-servidor, que define la interacción entre usuarios y servidores.

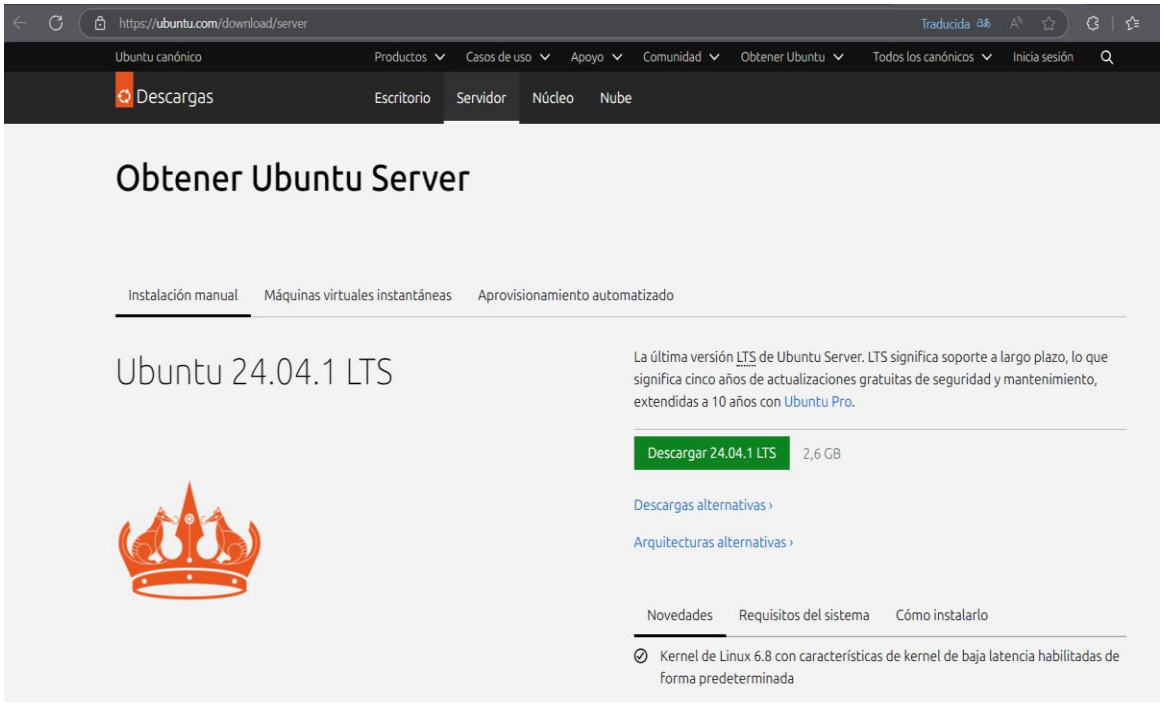
# 1. Sistemas operativos actuales y su soporte en servidores web.

La selección del sistema operativo para un servidor web es uno de los primeros pasos en su implementación, ya que determina la compatibilidad con el software del servidor, las herramientas disponibles y la facilidad de administración. En la actualidad, los sistemas operativos más utilizados para servidores web se dividen principalmente entre distribuciones de Linux, versiones de Windows Server y alternativas diseñadas para entornos cloud-native. A continuación, se detalla el estado actual de estos sistemas y su aplicación en servidores web.

**Distribuciones Linux: la elección preferida en servidores web.**

Linux domina ampliamente el mercado de servidores web debido a su flexibilidad, rendimiento, seguridad y coste (es gratuito en la mayoría de sus distribuciones). Ofrece una variedad de distribuciones optimizadas para diferentes usos, destacando las siguientes:

- Ubuntu Server (22.04 LTS y versiones posteriores): Popular por su facilidad de uso, actualizaciones regulares y amplia documentación. Es ideal para principiantes y expertos. Soporta perfectamente servidores web como Apache, NGINX y herramientas modernas como Docker y Kubernetes. Su comunidad activa asegura soluciones rápidas para problemas comunes.




*Pie de imagen: Web de descarga de Ubuntu 24.04.1 LTS.*

- Debian (12 "Bookworm"): Reconocido por su estabilidad y fiabilidad, Debian es la base de muchas otras distribuciones, incluyendo Ubuntu. Es una elección común para servidores web que requieren actualizaciones menos frecuentes pero estables. A menudo, se utiliza en entornos críticos de producción.
- CentOS Stream: CentOS se transformó en CentOS Stream, una distribución que ahora actúa como un puente entre Fedora y Red Hat Enterprise Linux (RHEL). Es ideal para empresas que necesitan un entorno cercano a RHEL, pero sin los costes asociados. Sin embargo, es importante destacar que CentOS clásico ya no se actualiza, y los usuarios han migrado a alternativas como AlmaLinux o Rocky Linux.
- AlmaLinux y Rocky Linux: Surgieron como reemplazos de CentOS clásico. Ambos ofrecen compatibilidad total con RHEL, lo que los convierte en opciones fiables para servidores empresariales y de hosting.
- Fedora Server: Una opción innovadora con soporte para las últimas tecnologías. Aunque su ciclo de vida es corto (aproximadamente 13 meses), es una excelente elección para entornos de desarrollo o pruebas.
- Arch Linux: Más enfocado en usuarios avanzados debido a su personalización extrema y actualizaciones continuas. Aunque no es común en producción, algunos servidores personalizados lo utilizan para casos específicos.

Las distribuciones Linux son altamente compatibles con los principales servidores web como Apache y NGINX, así como con tecnologías emergentes como Caddy Server, que destaca por su facilidad de configuración automática de HTTPS. Además, soportan ampliamente entornos de contenedores con Docker y orquestadores como Kubernetes, esenciales para infraestructuras modernas.

**Windows Server: un enfoque versátil y empresarial.**

Microsoft ha evolucionado significativamente su oferta para servidores, siendo Windows Server 2022 la versión más reciente. Este sistema operativo se utiliza principalmente en entornos empresariales donde se requiere integración con servicios de Microsoft, como Active Directory, Exchange Server o SharePoint.

Ofrece mejoras significativas en seguridad (Secure Core Server, cifrado TLS 1.3 por defecto), rendimiento y soporte para aplicaciones en contenedores. También incluye integración nativa con Azure, facilitando la implementación de soluciones híbridas entre servidores locales y la nube.

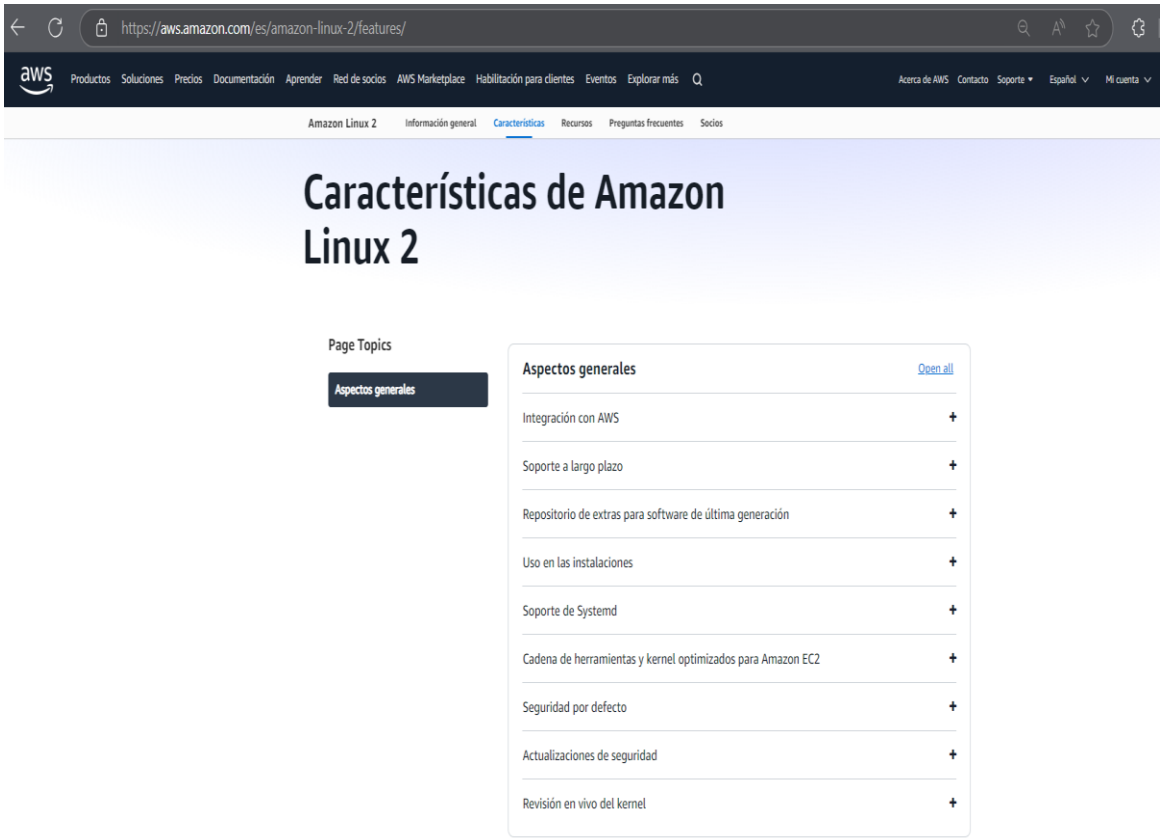
Además, Windows Server es compatible con IIS (Internet Information Services), el servidor web nativo de Microsoft. IIS es conocido por su facilidad de integración con tecnologías .NET, ASP.NET y bases de datos como SQL Server. Aunque Apache y NGINX también se pueden instalar en Windows, su rendimiento suele ser inferior al observado en Linux, especialmente en cargas altas.

A diferencia de Linux, Windows Server requiere licencias pagadas, lo que puede ser un factor limitante para pequeñas empresas o proyectos personales. Sin embargo, su robustez en entornos corporativos justifica su uso en muchas organizaciones.

**Alternativas cloud-native: sistemas operativos para la nube.**

El crecimiento de la computación en la nube ha impulsado la adopción de sistemas operativos diseñados específicamente para entornos cloud-native. Estas plataformas minimizan los requisitos de recursos, optimizan el rendimiento en contenedores y proporcionan integración directa con servicios en la nube.

- CoreOS/Flatcar Container Linux: Diseñado para entornos basados en contenedores, CoreOS (ahora mantenido como Flatcar) es minimalista y está optimizado para Kubernetes. Su diseño inmutable reduce errores relacionados con configuraciones incorrectas.
- RancherOS: Una distribución de Linux extremadamente ligera donde todo el sistema operativo se ejecuta como contenedores Docker. Es ideal para infraestructuras de microservicios.
- Amazon Linux 2: Proporcionado por AWS, es una opción optimizada para ejecutar servicios en la nube de Amazon. Incluye configuraciones predefinidas para facilitar la implementación de servidores web en instancias EC2.




*Pie de imagen: Características de Amazon Linux 2*

- Google Container-Optimized OS: Diseñado específicamente para Google Cloud Platform, es otra solución enfocada en contenedores y cargas de trabajo en la nube.

Los sistemas cloud-native están diseñados para trabajar con recursos limitados y maximizar la eficiencia en plataformas virtualizadas. Además, su mantenimiento automatizado reduce el trabajo manual asociado a la gestión de servidores tradicionales.

Tabla con las comparaciones clave entre los sistemas operativos:

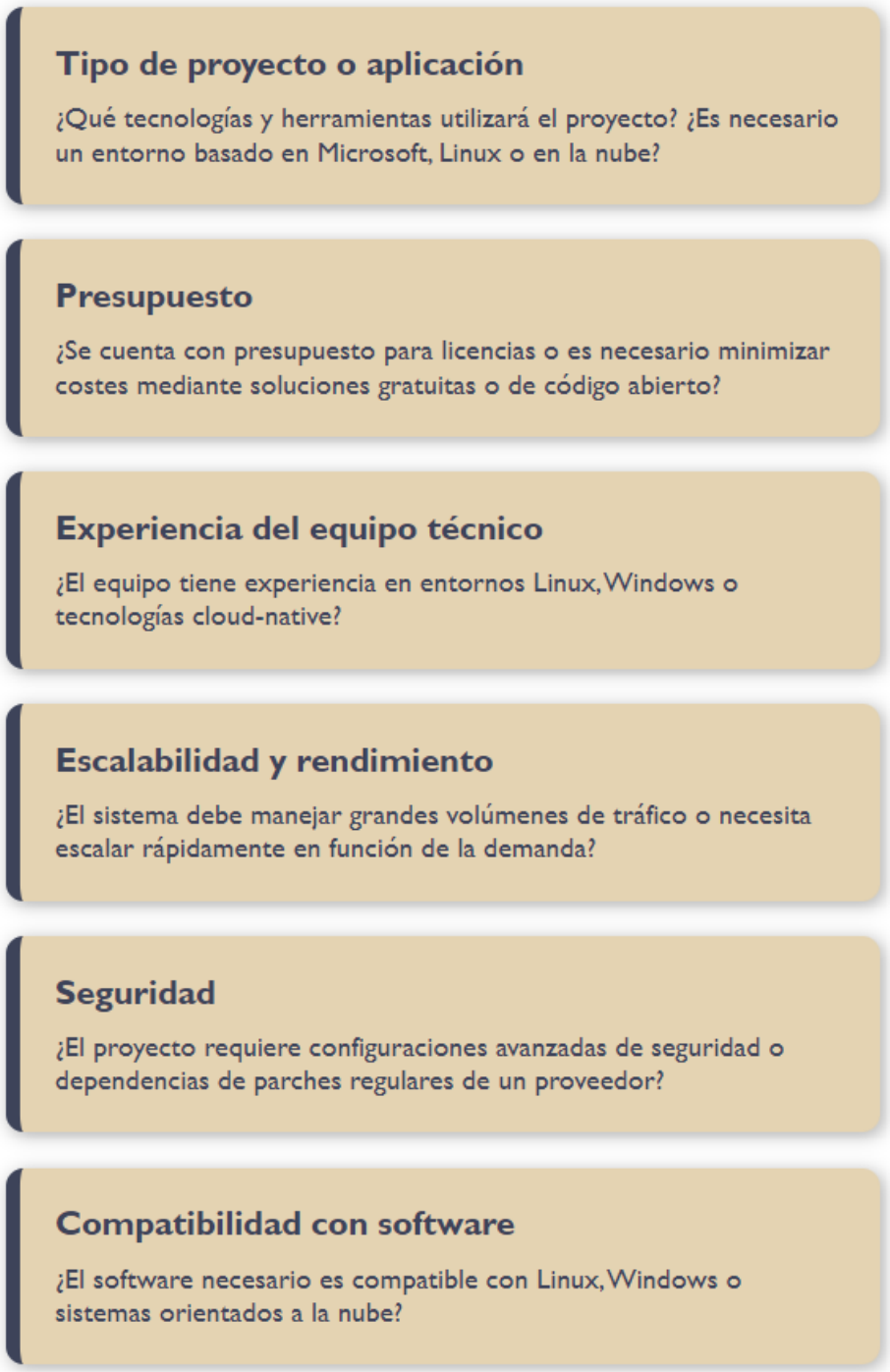
| Comparación | Características y ventajas |
|---|---|
| Linux vs Windows Server | • Linux es ideal para entornos de código abierto y flexibles.<br>• Windows Server es preferido en empresas que dependen de software de Microsoft.<br>• El coste de licencias favorece a Linux en proyectos con presupuestos ajustados.<br>• Linux tiene mejor rendimiento en servidores web como NGINX o Apache. |
| Linux vs Cloud-native | • Linux tradicional es más versátil en general.<br>• Cloud-native es superior para microservicios o infraestructura como código.<br>• En plataformas como AWS, las opciones nativas tienen mejor integración. |
| Windows Server vs Cloud-native | • Windows Server es adecuado para empresas tradicionales con entornos cerrados.<br>• Cloud-native es ideal para proyectos que priorizan agilidad y escalabilidad. |

## Consejo

Evita las prácticas obsoletas, por ejemplo:

- Evitar CentOS clásico. Ya no recibe actualizaciones. Opta por AlmaLinux o Rocky Linux como reemplazos directos.
- Evitar servidores monolíticos. La tendencia actual es adoptar arquitecturas de microservicios y contenedores.
- Tanto en Linux como en Windows Server, usar versiones obsoletas aumenta los riesgos de seguridad y limita el acceso a funcionalidades modernas.

La elección del sistema operativo para tu servidor web depende de varios factores, que incluyen las necesidades específicas de tu proyecto, el presupuesto disponible, el nivel de experiencia del equipo técnico y las características del entorno en el que se va a implementar el servidor. A continuación, se detalla cómo decidir cuál es el más adecuado basándote en criterios clave:




**1. Tipo de proyecto o aplicación.**

Linux:

- Si tu servidor alojará sitios web con herramientas de código abierto (WordPress, Drupal, Joomla) o aplicaciones modernas basadas en tecnologías como PHP, Python, Ruby o Node.js.
- Si necesitas un entorno altamente personalizable y flexible.
- Para proyectos que requieren soporte para servidores web populares como Apache, NGINX o Caddy.

- Ideal para entornos de contenedores y microservicios con herramientas como Docker y Kubernetes.

Windows Server:

- Si tu proyecto utiliza tecnologías de Microsoft, como ASP.NET, .NET Core o SQL Server.
- Si necesitas integrar tu servidor con otros servicios empresariales, como Active Directory, Exchange Server o SharePoint.
- Para aplicaciones que dependen de herramientas específicas de Windows o donde el cliente exige un entorno Windows.

Cloud-native (CoreOS, RancherOS, etc.):

- Si tu infraestructura está basada completamente en la nube (AWS, Google Cloud, Azure) y buscas optimizar el rendimiento para entornos de contenedores.
- Para proyectos que priorizan escalabilidad y automatización en arquitecturas distribuidas.

**2. Presupuesto.**

Linux:

- La mayoría de las distribuciones son gratuitas, lo que lo hace ideal si buscas minimizar costes. Incluso en entornos empresariales, las opciones de soporte técnico (Red Hat, SUSE) suelen ser más económicas que las licencias de Windows Server.

Windows Server:

- Requiere licencias de pago, tanto para el sistema operativo como para los servicios adicionales (por ejemplo, CALs - Client Access Licenses). Es una opción más costosa, pero justificada en entornos donde la integración con otras tecnologías de Microsoft aporta valor.

Cloud-native:

- Aunque el sistema operativo en sí puede ser gratuito, los costes asociados a la infraestructura en la nube (instancias, almacenamiento, ancho de banda) pueden ser más altos dependiendo del proveedor y la configuración.

**3. Experiencia del equipo técnico.**

Linux:

- Si tienes un equipo con experiencia en administración de sistemas Linux o con ganas de aprender. Las distribuciones como Ubuntu Server y CentOS Stream son amigables incluso para principiantes.
- Requiere habilidades para gestionar la línea de comandos y configuraciones manuales, aunque herramientas como Ansible, Terraform o cPanel pueden simplificar el proceso.

Windows Server:

- Más intuitivo para equipos acostumbrados a entornos gráficos y ecosistemas de Microsoft.
- Ideal si ya tienes personal con experiencia en herramientas como IIS, Active Directory y administración de sistemas Windows.

Cloud-native:

- Requiere conocimientos avanzados en contenedores (Docker), orquestadores (Kubernetes) y plataformas en la nube. Si tu equipo está especializado en tecnologías cloud, esta es una excelente opción.

**4. Escalabilidad y rendimiento.**

Linux:

- ► Sobresale en rendimiento bajo cargas intensas y es altamente eficiente en el uso de recursos. Ideal para servidores que manejarán grandes volúmenes de tráfico.
- ► Ofrece una amplia gama de herramientas para escalar horizontalmente (añadiendo más servidores) o verticalmente (mejorando los recursos de hardware).

Windows Server:

- ► Escalable, pero generalmente menos eficiente en el uso de recursos que Linux. Sin embargo, es robusto para aplicaciones empresariales específicas.

Cloud-native:

- ► Diseñado para entornos dinámicos y escalables. Perfecto para aplicaciones que deben crecer rápidamente según la demanda.

**5. Seguridad.**

Linux:

- ► Ofrece un alto nivel de seguridad y control. Las configuraciones avanzadas (como SELinux o AppArmor) refuerzan la protección.
- ► La comunidad activa de Linux asegura actualizaciones constantes y rápidas para vulnerabilidades.

Windows Server:

- ► También es seguro, pero históricamente ha sido un objetivo frecuente de ataques. Microsoft proporciona parches regulares, pero dependerás de ellos para mantener el sistema actualizado.

Cloud-native:

- ► Optimizado para entornos modernos, a menudo integra características de seguridad avanzadas como actualizaciones automáticas y configuraciones inmutables. Sin embargo, la seguridad depende en gran medida de la infraestructura subyacente (AWS, Azure, etc.).

**6. Compatibilidad con software.**

Linux:

- ► Compatible con una amplia gama de aplicaciones de código abierto. Es ideal si tu proyecto requiere integración con herramientas como MySQL, PostgreSQL, Redis, Elasticsearch o MongoDB.

Windows Server:

- ► Necesario si planeas usar aplicaciones que dependen de tecnologías Microsoft, como SQL Server, Visual Studio o PowerShell.

Cloud-native:

- ► Orientado a entornos basados en contenedores y microservicios. Es la mejor opción si tu aplicación utiliza tecnologías modernas como Kubernetes o funciones serverless.

# Actividad 1

Elije el sistema operativo recomendado según el caso y justifica tu respuesta:

Caso 1: Proyecto web con WordPress y NGINX

Sistema operativo recomendado: …

Justificación: …

Caso 2: Plataforma interna empresarial basada en .NET Core

Sistema operativo recomendado: …

Justificación: …

Caso 3: Aplicación basada en microservicios con alta escalabilidad

Sistema operativo recomendado: …

Justificación: …

## Recurso

¿Qué significa seleccionar un sistema operativo para un servidor web?

Es como elegir el motor de un coche. Según lo que quieras hacer con el coche (ir rápido, ahorrar combustible, transportar carga), eliges un motor que se adapte a tus necesidades. De manera similar, para un servidor web (la "máquina" que hace que las páginas de internet funcionen), necesitas un sistema operativo que se adapte al tipo de trabajo que hará el servidor.

¿Qué opciones tienes?

Distribuciones Linux (el favorito en servidores web):

Linux es como los coches eléctricos: Económico, eficiente y con muchas opciones para personalizar. A veces necesita algo más de conocimiento para configurarlo al principio, pero a largo plazo es muy rentable.

Aquí tienes algunas "marcas" populares de Linux:

- Ubuntu Server: Es como un coche híbrido fácil de manejar. Sirve para principiantes y expertos. Tiene muchísima "documentación" (instrucciones y ayudas) que te sacan de cualquier apuro. Además, se lleva muy bien con herramientas modernas como Docker (imaginemos que Docker es como una herramienta para "empaquetar mercancías" que quieres enviar con tu servidor).
- Debian: Es el "camión robusto" que no te deja tirado. Es más serio y estable. Perfecto para sitios web importantes que necesitan pocas interrupciones.
- CentOS Stream, AlmaLinux y Rocky Linux: Son opciones para "empresas grandes" que necesitan algo parecido a un Mercedes, pero no quieren pagar tanto por los extras.
- Arch Linux: Este es como un coche de carreras personalizado. Solo lo usan los que realmente saben cómo funciona porque necesita ajustes constantes.

Comparación: Si montaras una cafetería, usarías Linux porque es como un proveedor que te da los ingredientes gratis, pero tú decides cómo usarlos.

Windows Server (la opción empresarial):

Windows Server es como una oficina de Microsoft bien equipada. Si tu empresa ya usa herramientas como Word, Excel, Exchange o bases de datos SQL, tiene sentido usar un sistema operativo que se integre bien con estas herramientas. Además, trae su propio "servidor web" llamado IIS.

¿El lado no tan bueno? Necesitas pagar licencias, como si fuera una suscripción premium. Por eso, si tienes poco presupuesto, puede no ser la mejor opción.

Comparación: Si montarás un restaurante elegante y ya tienes contacto con proveedores premium (Microsoft), usas Windows Server porque todo está integrado.

Alternativas cloud-native (diseñadas para la nube):

Esto es como usar drones en lugar de camiones. Si el servidor no está físicamente en tu oficina y en vez de eso está en "la nube" (imaginemos que es un gran almacén compartido con otros usuarios), estos sistemas operativos están optimizados para sacar el máximo partido de esos recursos.

- Amazon Linux 2: Es el sistema que Amazon pone para quienes usan su almacén.
- CoreOS/Flatcar: Ideal si usas "drones" (contenedores como Docker o Kubernetes) para mover cosas en la nube.
- RancherOS: Todo el sistema funciona como contenedores, lo que lo hace muy eficiente.

Comparación: Si tienes un negocio moderno, con pedidos online y logística automatizada, eliges sistemas cloud-native porque están hechos para trabajar "en el aire."

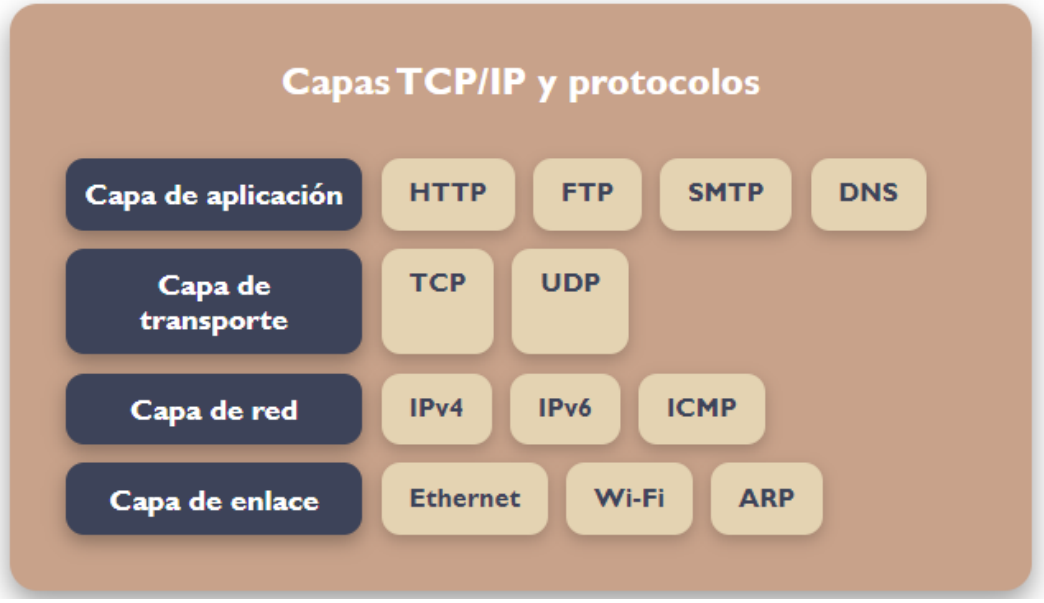
Entonces, ¿cómo decidir?

- Si necesitas algo barato y flexible: Elige Linux (por ejemplo, Ubuntu o Debian).
- Si ya estás usando servicios de Microsoft: Elige Windows Server.
- Si tu servidor estará en la nube: Usa alternativas cloud-native como Amazon Linux o RancherOS.

# 2. Fundamentos de TCP/IP.

TCP/IP (Protocolo de Control de Transmisión/Protocolo de Internet) es el conjunto de protocolos que forma la base de las comunicaciones en internet y redes locales. Su importancia radica en permitir que los dispositivos intercambien datos independientemente del hardware, el sistema operativo o la red subyacente. Entender cómo funciona es esencial para gestionar un servidor web, ya que los servidores utilizan TCP/IP para interactuar con los clientes.

TCP/IP consta de varias capas que organizan y gestionan la transmisión de datos. Estas capas incluyen:




Capa de aplicación:

- Proporciona servicios específicos al usuario. Por ejemplo, HTTP para la web, FTP para transferencia de archivos o SMTP para correo electrónico.
- Es la capa más relevante para un servidor web, ya que define cómo los datos llegan y se procesan en aplicaciones como navegadores o clientes FTP.

Capa de transporte:

- Gestiona la transmisión de datos entre sistemas finales.
- Protocolos destacados:
  - TCP (Protocolo de Control de Transmisión): Establece conexiones fiables y orientadas a la comunicación, garantizando que los paquetes lleguen completos y en orden.
  - UDP (Protocolo de Datagrama de Usuario): Más rápido, pero no garantiza la entrega de paquetes. Aunque se usa menos en servidores web, es útil para aplicaciones como transmisiones en vivo.

Capa de red:

- Encargada de enrutar los datos desde el origen hasta el destino utilizando direcciones IP.
- Soporta dos versiones principales:
  - IPv4: La versión más utilizada, aunque su espacio de direcciones está agotado.
  - IPv6: Diseñado para reemplazar a IPv4, ofrece un espacio de direcciones mucho mayor y mejor eficiencia en el enrutamiento.

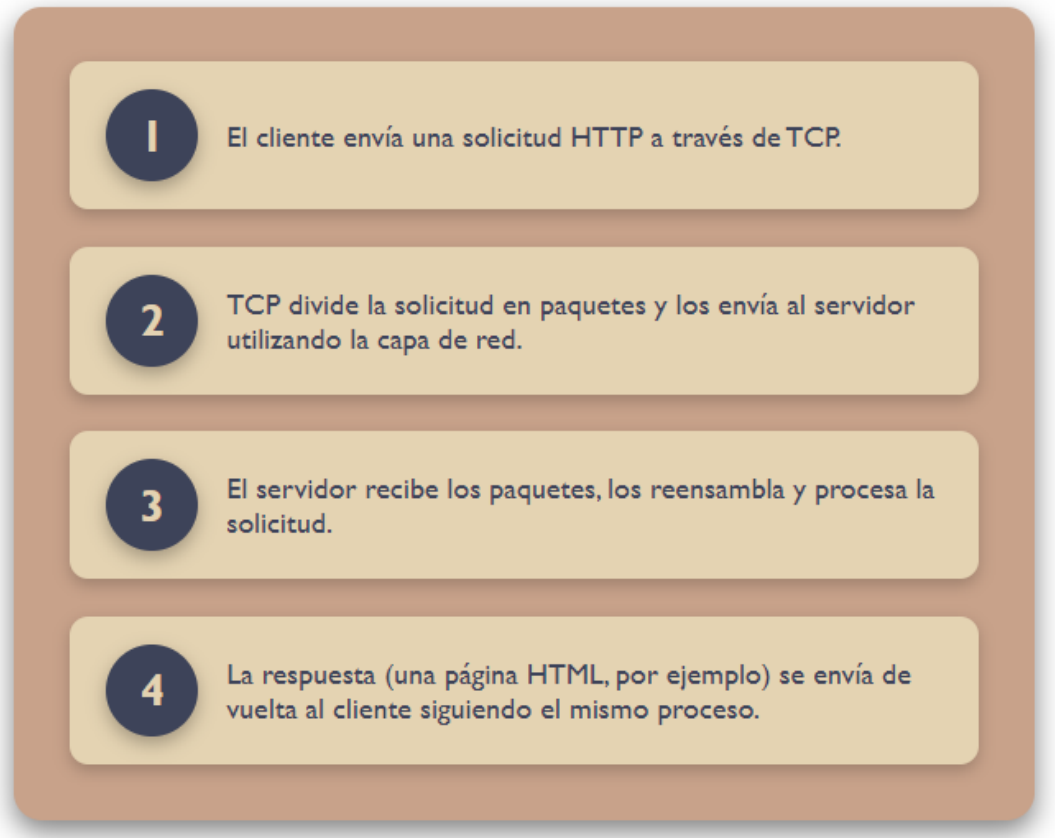
Capa de enlace:

- Proporciona acceso a los medios físicos de la red (cables, Wi-Fi, etc.) y garantiza la transmisión de datos entre dispositivos conectados directamente.

## Anotación

Las direcciones IP identifican de manera única a un dispositivo en una red, como 192.179.2.1 en el caso de IPv4, o direcciones más extensas en IPv6, cuya adopción es prioritaria debido al agotamiento de IPv4. Los puertos, por su parte, son números que permiten asociar servicios específicos en un servidor, como el puerto 80 para HTTP, el 443 para HTTPS y el 22 para SSH. Sin embargo, el uso de HTTP sin cifrado ya no es aceptable en entornos seguros, siendo HTTPS el estándar actual para garantizar la seguridad en la comunicación.

Cuando un cliente (por ejemplo, un navegador) solicita una página web:



# 3. Estructura Cliente / Servidor.

El modelo cliente/servidor es una arquitectura que describe cómo interactúan dos partes: un cliente, que solicita recursos o servicios, y un servidor, que los proporciona. Este modelo es el núcleo de cómo funciona un servidor web.

Componentes del modelo cliente/servidor

Cliente:

- Un dispositivo o software que inicia la comunicación.
- Ejemplos: navegadores web, aplicaciones móviles o herramientas de línea de comandos como curl.

Servidor:

- Una máquina o software que espera solicitudes y responde con los recursos solicitados.
- Ejemplos: servidores web como Apache, NGINX o IIS.

Proceso de comunicación:

1. El cliente inicia una conexión enviando una solicitud al servidor a través de una red (local o internet).
2. El servidor escucha las solicitudes en un puerto específico (por ejemplo, el puerto 80 para HTTP).
3. El servidor procesa la solicitud, accede al recurso solicitado y devuelve una respuesta.
4. La conexión puede mantenerse abierta (en el caso de HTTP Keep-Alive) o cerrarse tras completar la transacción.

**Ejemplo**

Por ejemplo:

Un usuario introduce una URL en su navegador (https://ejemplo.com).

El navegador (cliente):

1. Realiza una resolución DNS para obtener la dirección IP del servidor (93.184.216.34).
2. Envía una solicitud HTTP GET al puerto 443 (HTTPS) del servidor.
3. El servidor (NGINX, por ejemplo) recibe la solicitud, busca el archivo correspondiente en su sistema de archivos y lo devuelve al cliente.
4. El navegador procesa la respuesta y muestra la página al usuario.

El modelo cliente/servidor presenta varias ventajas clave que lo hacen ampliamente utilizado en aplicaciones web y de red. Una de sus principales características es la eficiencia, ya que permite a los servidores gestionar múltiples solicitudes de clientes al mismo tiempo, lo que optimiza los recursos y el rendimiento. Además, ofrece una gran escalabilidad, permitiendo añadir más servidores en forma de clústeres o balanceadores de carga para soportar una mayor demanda. Por último, destaca la separación de responsabilidades, donde los clientes se encargan de la interfaz de

usuario, mientras que los servidores se concentran en el procesamiento de datos y el almacenamiento, garantizando una distribución eficiente de tareas.

En cuanto a las prácticas modernas y obsoletas, el uso de servidores distribuidos y balanceo de carga es esencial para mejorar la disponibilidad y resiliencia del sistema. Asimismo, la implementación de servicios RESTful optimiza la comunicación entre cliente y servidor, facilitando la interoperabilidad. El uso de HTTPS se ha convertido en un estándar indispensable para proteger las comunicaciones frente a ataques como la intercepción de datos. Por el contrario, prácticas como el uso de servidores monolíticos, que concentran toda la lógica en una única instancia, están quedando atrás en favor de arquitecturas de microservicios más flexibles. De igual manera, HTTP sin cifrar solo se considera aceptable en redes internas o durante pruebas, ya que carece de las garantías de seguridad necesarias en entornos modernos.

Por otro lado, el modelo cliente/servidor se diferencia claramente de otros enfoques como el peer-to-peer (P2P). En el modelo cliente/servidor, el servidor asume un rol fijo, proporcionando servicios centralizados a múltiples clientes, lo que es ideal para aplicaciones web con un punto de acceso único. En contraste, en el modelo P2P, los nodos actúan tanto como clientes como servidores, compartiendo recursos de manera descentralizada, como en redes de intercambio de archivos tipo BitTorrent. Esta diferencia refuerza el control centralizado y la gestión más eficiente del modelo cliente/servidor en escenarios donde la estabilidad y el control son prioritarios.

# Actividad 2

Determina si las siguientes afirmaciones sobre el modelo cliente/servidor y sus características son verdaderas (V) o falsas (F).

El modelo cliente/servidor permite a los servidores gestionar múltiples solicitudes de clientes al mismo tiempo, lo que optimiza los recursos y el rendimiento.

En arquitecturas modernas, el uso de servidores monolíticos es preferido sobre los microservicios debido a su simplicidad y rendimiento.

El uso de HTTPS se considera innecesario en la mayoría de los casos modernos, excepto en entornos de prueba.

El modelo cliente/servidor es más adecuado que el modelo P2P para aplicaciones con un único punto de acceso centralizado.

La implementación de servicios RESTful facilita la comunicación entre cliente y servidor al optimizar la interoperabilidad.

# 4. Prueba de autoevaluación.

***¿Cuál de las siguientes distribuciones de Linux es conocida por su estabilidad y ser la base de otras distribuciones como Ubuntu?***

*a) Fedora Server*

*b) Debian*

*c) Arch Linux*

***¿Qué sistema operativo es ideal para entornos empresariales que requieren integración con servicios de Microsoft?***

*a) Ubuntu Server*

*b) Windows Server 2022*

*c) CoreOS*

***En el modelo TCP/IP, ¿qué protocolo garantiza conexiones fiables y orientadas a la comunicación?***

*a) UDP*

*b) HTTP*

*c) TCP*

***¿Cuál de las siguientes es una alternativa cloud-native diseñada para entornos basados en contenedores?***

*a) CentOS Stream*

*b) RancherOS*

*c) Windows Server*

***¿Qué práctica obsoleta se debe evitar al seleccionar un sistema operativo para servidores web?***

*a) Usar arquitecturas de microservicios y contenedores*

*b) Implementar servidores monolíticos*

*c) Adoptar sistemas cloud-native*

***_____ es el conjunto de protocolos que forma la base de las comunicaciones en internet y redes locales.***

***En el modelo cliente-servidor, el _____ inicia la comunicación enviando una solicitud al servidor.***

***Las distribuciones _____ son la elección preferida para servidores web debido a su flexibilidad y rendimiento.***

***_____ es el servidor web nativo de Microsoft compatible con Windows Server.***

***En entornos modernos, es indispensable el uso de _____ para proteger las comunicaciones frente a ataques.***

# Manejo del protocolo http