CLASE 1 SEMANA 1

ARQUITECTURA CLIENTE SERVIDOR

La arquitectura cliente-servidor persigue el objetivo de procesar la información de un modo distribuido. De esta forma, pueden estar dispersos en distintos lugares y acceder a recursos compartidos.

Además de la transparencia y la independencia del hardware y del software, una implementación cliente-servidor debe tener las siguientes características:
-Utilizar protocolos asimétricos, donde el servidor se limita a escuchar en espera de que un cliente inicie una solicitud.
-El acceso es transparente, multiplataforma y multiarquitectura.
-Se facilitará la escalabilidad, de manera que sea fácil añadir nuevos clientes a la infraestructura —escalabilidad horizontal— o aumentar la potencia del servidor o servidores, —escalabilidad vertical—.

Características fundamentales de un S.O. de servidor:
-Soporte de red → Es indispensable que tengan un soporte completo para poder brindar conectividad.
-Amplia compatibilidad con el hardware → priorizar el uso de S.O. actualizados y con un soporte importante de controladores.
-Seguridad → que el S.O. instalado sea seguro: esté actualizado y tener políticas estrictas de acceso para prevenir accesos no autorizados o ataques. Reforzarse con la instalación de Firewalls (por software o hardware) y antivirus. En este ítem también debemos incluir el respaldo de la información.
-Tolerancia a fallos → mediante la generación de granja de servidores, que interconectados, operen como una gran unidad de proceso, dando la posibilidad que ante la caída de uno de los integrantes de la granja, otro puede tomar su rol y responsabilidad.

SISTEMAS OPERATIVOS

Servicios que ofrecen los sistemas operativos:
-Servicio de publicación web → Despachar el contenido de un sitio web al usuario.
Apache(multiplataforma), IIS(solo windows), Nginx(multiplataforma), Lite Speed(linux)
-Servicio de base de datos → database server o RDBMS → permiten la organización de la información mediante el uso de tablas, índices y registros. Provee de información a otras aplicaciones web o equipos.
MySQL(multiplataforma, relacional), PostgreSQL(multiplataforma, relacional), Microsoft SQL SERVER(multiplataforma, relacional), Mongo DB (multiplataforma, no relacional).
-Servicio de correo electrónico → encargado de enviar y recibir mensajes de correo electrónico entre hosts, usuarios o servidores. Usa protocolo TCP/IP.
MTA(transferir el email de un host a otro)→ Qmail,Exim,Postfix,Microsoft ES,Courier+Cyrus.
MDA(recibir el correo de un MTA y llevarlo al inbox comunicándose previo con POP o IMAP) → Dovecot, Procmail, Maildrop
-Servicio de archivos → proporciona un lugar de almacenamiento centralizado para los archivos en sus propios soportes de datos, disponible para todos los clientes autorizados.
CIFS/Samba(linux), NTFS Share(Microsoft), NFS (UNIX)

-Servicio de red → satisfacer necesidades de ruteo, firewall o proxy. PFSense, OPNSense y DD-WRT.
-Servicio de dominio → responde a las solicitudes de autenticación y verifica a los usuarios en las redes informáticas. Active Directory(windows) y NIS y NIS+ (linux).

DIFERENCIAS ENTRE LINUX Y WINDOWS:

-Interfaz de Usuario: Windows = interfaz gráfica; linux=consola.
*Linux → Shell. Interfaz de línea de comandos. Soporta nativamente un componente de script, llamado Bash → permite optimizar y automatizar multiplicidad de trabajos.
*Windows → GUI. interfaz gráfica. Powershell interfaz de consola muy potente.
-Filosofía y licenciamiento =
*Linux → open source, mayormente gratuito.
Aparecen distribuciones: se toman el núcleo Linux y se le agregan determinados paquetes. Debian, Ubuntu,RedHat, SUSE.
*Windows → Privativo, propietarios y pago.
De código cerrado, solo Microsoft puede modificarlo. Essentials(hasta 25 clientes), Standard(Sin limites de clientes, límite en virtualización), Datacenter(soporte completo para virtualización y contenedores).
-Sistema de paquetes =
*Linux → Gestor de paquetes: permite que se instale la versión/compilación de software más adecuada a la distribución. y repositorios: descarga legítimo.
DPKG(Debian Package Manager), RPM(Red Hat Package Manager), Pacman Package Manager.
*Windows → un poco más centralizado. Estandarización de paquetes, ejecutables .exe y .msi
Paquetes: 32bit Window 2000 y 2003; y 64bit Window 2003 64bit, 2008, 2012, 2016 y 2019
Gestores de paquetes alternativos: Chocolatey y Ninite → desarrollados por terceros.
-Aspectos técnicos=
*Kernel → software que constituye una parte fundamental del S.O. Responsable de facilitar a los distintos programas acceso seguro al hardware de la computadora.

Windows	Linux
• En dos capas, una de kernel y otra de usuario. • Usa una lista de control de acceso. • Incluye el kernel los aspectos relativos a la interfaz gráfica. • Almacena sus configuraciones en registros.	• De una sola capa, la del propio kernel. • Usa los permisos tradicionales de Unix para el control de acceso a archivos. • Los aspectos relativos a la interfaz gráfica se apoya en la capa de usuario. • Almacena sus configuraciones en archivos.

*Sistemas de archivos → encargado de administrar y facilitar el uso de las memorias periféricas. Que el usuario pueda identificar los archivos sin lugar a error y acceder a ellos lo más rápido posible.

Windows	Linux
• **FAT/ FAT16 / FAT32**, heredado desde el MS-DOS. Tiene un límite máximo de 4 GB por archivo. • **NTFS,** el más utilizado en Windows ya que brinda la posibilidad de tener campos específicos para incorporar permisos de Active Directory, en la actualidad va por su 3er evolución, incorporando soporte nativo para SSD.	• **EXT**, el cual esta ya es su cuarta evolución, es el más utilizado ya que tiene amplio soporte para las unidades SSD. • **ReiserFS,** de propósito general, apunta a la baja corrupción de archivos. • **swap,** es el sistema de archivos para los archivos de intercambio.

*Niveles de ejecución → estado init. Define qué servicios y recursos están disponibles para los usuarios.

Windows	Linux
• **Modo seguro con funciones de red**. Inicia en modo seguro e incluye los controladores y servicios de red • **Modo seguro con símbolo del sistema**. Inicia Windows en modo seguro con una ventana de símbolo del sistema en lugar • **Habilitar el registro de arranque**. Crea un archivo llamado ntbtlog.txt en el que que puede resultar útil para la solución avanzada de problemas. • **La última configuración válida conocida**. Inicia Windows con la última configuración del registro y los controladores que funcionó correctamente. • **Deshabilitar reinicio automático tras error.** Impide que se reinicie automáticamente en caso de que un error • **Iniciar Windows normalmente.**	• **Nivel 0**: Sistema Apagado • **Nivel 1:** Solo usuario root, sin red ni demonios • **Nivel 2:** Multiusuario sin red • **Nivel 3:** Inicio normal, con consola • **Nivel 5:** Inicio normal, con consola y ademas interfaz gráfica • **Nivel 6:** Reinicio del sistema

BASH

Las funciones Bash pueden:
●Eliminar tareas repetitivas.
● Ahorrar tiempo.
● Proporcionar una secuencia de actividades bien estructurada, modular y formateada.
● Con scripts, podemos proporcionar valores dinámicos a comandos usando argumentos de línea de comando.
● Puede simplificar comandos complejos en una sola unidad en ejecución.
● Una vez creada, se puede ejecutar cualquier cantidad de veces por cualquier persona. Construye una vez y ejecuta muchas veces.
● Los flujos lógicos se pueden construir utilizando funciones Bash.
● Las funciones Bash se pueden ejecutar al inicio del servidor o agregando un cron job programado.
● Los comandos pueden ser depurados.
● Tener comandos de shell interactivos.

Ejemplo función Bash:

```
#!/bin/bash
testfunction
testfuncion(){
        echo “My first function”
}
//My first function.
```

Pueden aceptar cualquier número de parámetros.
Los scripts bash soportan → while, for, if, and, or , Else if, case.

POWERSHELL

Ventaja en curva de aprendizaje, pocos cambios en el tiempo.
Los comandos están estructurados de la siguiente manera:
un verbo y un nombre separados por un guión (-): verbo-nombre. Get-Command

WSL

Característica introducida en Windows 10 que nos permite instalar un Kernel Linux directamente sobre el S.O. de Microsoft. Es posible gracias a la virtualización de Hyper-V de Microsoft. Permite usar todas las herramientas y todos los servicios de Linux sin tener que virtualizar. Requisitos:
-Windows 10 en sus versiones.
-Acceso administrativo
-3GB mínimo de espacio libre.

CLASE 2 SEMANA 1

VIRTUALIZACIÓN

1967 IBM → Para crear entornos aislados para múltiples usuarios, se crea el primer S.O. que utiliza virtualización. Rymarczyk.
1972 → se dispone comercialmente la virtualización para tecno del tipo mainframe para grandes corporaciones.
1980 → virtualización para servidores de arquitectura x86. Este tipo de arq poseen las computadoras de escritorio.
1999 → VMware. Virtualización para cualquier tipo de usuario hogareño.
2003 → Xen. Primera plataforma open source de virtualización.
2007 → Primera VirtualBox, plataforma open source para virtualización hogareña.

Componentes de la virtualización:
*Maquinas Virtuales - Maquinas Virtuales - Maquinas Virtuales
*Administrador de máquinas virtuales. → administran todos los recursos físicos y virtuales de los guest. Podremos establecer clustering con otros virtual machines manager para tener alta disponibilidad y tolerancia a fallos.
*Sistema operativo base → Encargado de administrar los dispositivos físicos(hardware) y proveer una capa de abstracción a los entornos virtuales.
*Hardware (servidores físicos) → los microprocesadores tienen una caract. llamada virtualización de CPU.
Beneficios de la virtualización:
*Tiempo de actividad → evitar tiempos de inactividad aprovechando al max los recursos.
*Despliegue → implementación de recursos + rápida c/ la virtualización. La implementación de M.V. s es + simple teniendo de antemano imágenes listas para desplegar.
*Ahorro de energía → el sistema es + eficiente energéticamente → no se utiliza ningún hardware o software + allá del previsto p/ la virtualización.
*Snapshots → proporcionan un registro de cambios p/ el disco virtual y se utilizan para restaurar una M.V.a un punto donde ocurre una falla o error del sistema.
*Backups → el proceso de recuperación ante desastres es sencillo en entorno virtualizado. Se puede realizar copias del servidor virtual y en las M.V. se pueden migrar entre sí.
*Alta disponibilidad → funcionalidad que asegura tener redundancia entre 2 o + ambientes. Garantiza que ante la falla de 1 de los 2 no exista disrupción del servicio.
*Costo → Es + económico → no requiere ningún componente de hardware.
*Eficiencia → Actualización de software y hardware de las M.V. sin grandes impactos en la productividad de los sistemas involucrados.

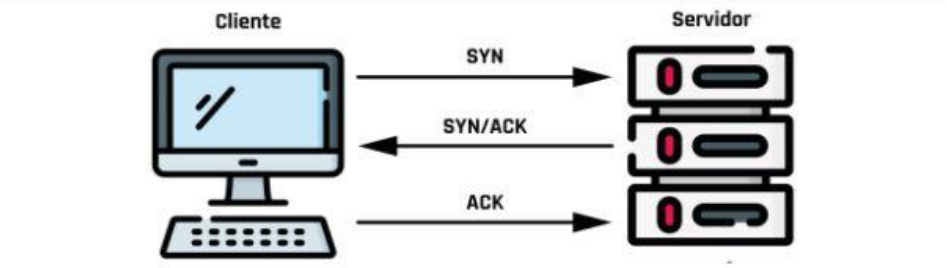
VirtualBox → tecnología de virtualización multiplataforma.
Una plataforma de virtualización utiliza las características necesarias de hardware y software para ejecutar múltiples M.V. en una misma computadora física.

Vagrant → Herramienta de línea de comando → desarrollada por HashiCorp. Permite:
-controlar el ciclo de vida de las M.V.,
-automatizar su configuración,
-generar imágenes con software preinstalado y
-gestionar ambientes de desarrollo de manera sencilla.

CLASE 4 SEMANA 2

REDES

Los tipos de comunicación de los dispositivos que se comunican mediante el protocolo IP se establecen en 2 protocolos: TCP y UDP.
UDP (protocolo de datagrama de usuario) → Se utiliza para enviar información de manera rápida y sin esperar la confirmación de recepción de los paquetes enviados.
TCP (protocolo de control de transmisión) establece una conexión antes de enviar el tráfico, luego envía los paquetes de datos y confirma la recepción.
Ese flujo de transmisión se lo conoce como 3 way handshake:



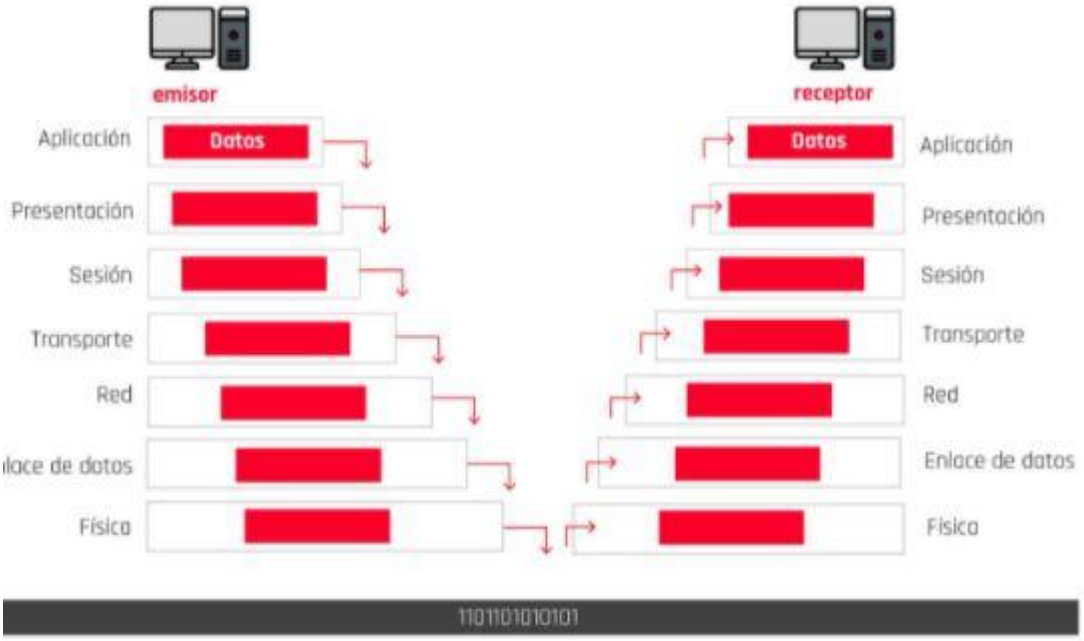
Mensaje	Descripción
Syn	Se utiliza para iniciar y establecer una conexión.
ACK	Ayuda a confirmar al otro lado que ha recibido el SYN.
SYN-ACK	Mensaje SYN del dispositivo local y ACK del paquete anterior.
FIN	Se usa para terminar una conexión.

MODELO OSI

Modelo conceptual de interconexión que permite que diversos sistemas se comuniquen mediante un estándar. Se puede entender como un lenguaje universal de comunicación entre redes, computadoras, servidores, etc., que se basa en la idea de dividir un sistema de comunicación en siete capas y cada una de ellas trabaja sobre la precedente.

7-Capa de aplicación → interactúa con los datos del usuario. Apps, nav web y clientes de correo electrónicos → dependen de esta capa p/ → iniciar comunicaciones.
6-Capa de presentación → definir el formato y el cifrado de los datos, gestionar la seguridad y confidencialidad de la red, compresión y empaquetado de texto. **Datos.**
5-Capa de sesión → responsable de la apertura y cierre de comunicaciones entre 2 dispositivos y su función es crear una sesión o conexión que permite que 2 dispositivos se comuniquen entre sí.

4-Capa de transporte → responsable de coordinar la transferencia de datos a través de las conexiones de red.TCP y UDP. Agrupa los datos en **Segmentos**, agrega un encabezado a cada segmento.
3-Capa de red → responsable de posibilitar las transferencias de datos entre 2 redes diferentes. Fragmenta, en el dispositivo emisor, los datos de la capa de transporte en unidades + pequeñas llamadas paquetes y rearmarse dsp en el dispositivo receptor. Enrutar. Convierte cada segmento en un **Paquete** adjuntando otro encabezado.
2-Capa de enlace de datos → facilitar la transferencia de datos entre 2 dispositivos ubicados en una misma red. Tramas = trozos + pequeños que rompe de los paquetes de la capa de red. Subcapa “MAC address” → ayuda a controlar el flujo de paquetes. Convierte cada paquete en una trama adjuntando otro encabezado y un trailer. **Frames**
1-Capa física → dispositivos físicos que participan en la transferencia de datos y se convierten en una secuencia de bits, que es una serie de unos y ceros. Electricidad a **Bits**.Traduce la trama en una serie de bits y transmite los datos al medio.



La trama es transmitida a través del medio en forma de bits

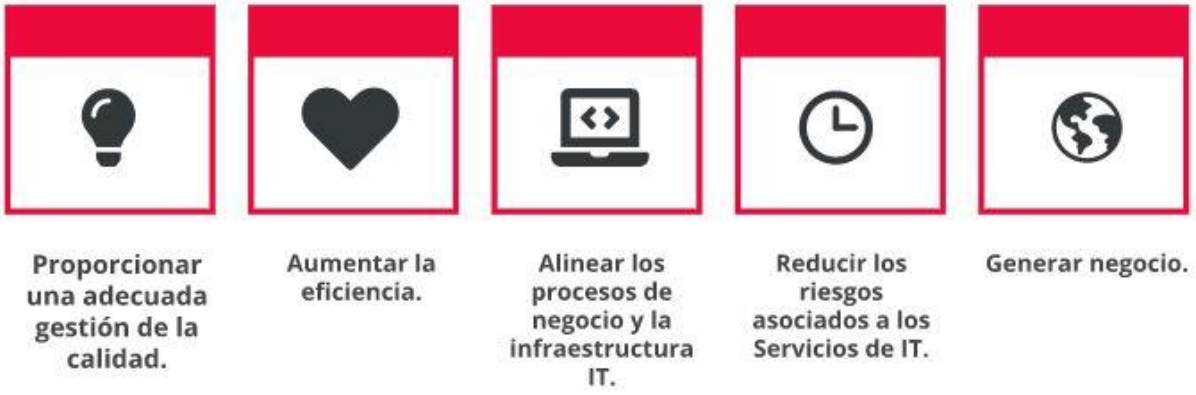
En el extremo receptor, los datos se deben desempaquetar.
La capa de enlace elimina el primer encabezado y el final de la trama y pasa el paquete adjunto a la capa de red.
Esta capa quita el encabezado del paquete y pasa el segmento adjunto a la capa de transporte.

La capa de transporte espera que lleguen suficientes segmentos y luego ensambla los segmentos para crear el flujo de datos original y lo pasa a los niveles superiores.
Todo este proceso también es denominado **encapsulación** y **desencapsulación de datos** en el modelo OSI.

Guia de troubleshooting de red
Paso 1 → chequear hardware p/ → asegurarnos que esté conectado correctamente, encendido y funcionando.
Paso 2 → Utilizar el comando ipconfig en la terminal. La puerta de enlace predeterminada es la IP del router.
Paso 3 → Utilizar los comandos ping y tracert. Para visualizar en qué parte del camino está surgiendo el error. (ping 8.8.8.8 -t) → siga haciendo ping a los servidores mientras solucionan el problema o “tracert 8.8.8.8” → nos mostrará cada salto entre los routers y los servidores de Google.
Paso 4 → Utilizar el comando nslookup p/ → determinar si hay un problema con el servidor al que estamos intentando conectarnos.

ITSM

Gestión de servicios de tecnologías de la información , disciplina basada en procesos, enfocada en alinear los servicios de TI proporcionados con las necesidades de las empresas, poniendo énfasis en los beneficios que puede recibir el cliente final.
Los objetivos de una buena gestión de servicios IT son:



ITIL → guía de soluciones y buenas prácticas para la gestión de servicios de tecnologías de la información (TI). Marco de trabajo para ITSM. Versiones a lo largo del tiempo:

¿Cómo se relacionan ITIL e ITSM?
ITSM → necesidad de las organizaciones para gobernar sus procesos tecnológicos.
ITIL → es una respuesta a la necesidad que plantea ITSM

ITSM permite a las empresas de IT:
-Estandarizar los procesos → Mantener la cohesión. Elimina las conjeturas junto con la toma de decisiones individuales de la administración de servicios de IT.
-Agilizar las tareas simples → Creación de un portal de autoservicio de IT permite que el personal de IT se centre en cuestiones más críticas e iniciativas comerciales.
-Tomar decisiones basadas en datos → Analizar información para la toma de decisiones y realizar ajustes más inteligentes basados en las necesidades reales.

Procesos tradicionales de **ITIL**:
1- Gestión de solicitud de servicio → registrarlas y manejarlas según la urgencia. Ejemplos: Agregar un usuario a un grupo o crear una máquina virtual.
2- Gestión de incidentes → recuperar el nivel habitual de funcionamiento del servicio y minimizar el impacto negativo en la organización de forma que la calidad del servicio y disponibilidad se mantengan. Ejemplos: Fallo de una app o Capac. del disco duro excedida.
3- Gestión de problemas → determinar y eliminar la causa. Prevención de incidentes y minimización del impacto de aquellos que no pueden prevenirse. Solución alternativa.
4- Habilitación del cambio o control de cambios → proceso de controlar y gestionar un cambio a lo largo de todo su ciclo de vida, con el objetivo de minimizar su riesgo. Ejemplo: reemplazo de impresoras, servidores, entre otros. Cambiar de dominio una máquina.
5- Gestión de la configuración → permite gestionar los cambios de la configuración de nuestros activos informáticos., permitiendo a la organización mantener un registro histórico y a su vez aplicar controles. Ejemplo: Reemplazo de módulos de memoria en un servidor. Instalación de software.
6- CMDB Configuration management database → base de datos donde administra y gestiona todos los elementos de la compañía(configuration items o CI) que son necesarios para la prestación de servicios.

CLASE 5 SEMANA 2

CRIPTOGRAFÍA

Técnica que tiene por objetivo cifrar, codificar o encriptar datos y/o mensajes con el fin de hacerlos ininteligibles a receptores no autorizados.

Algoritmos y claves:
Para lograr esconder un mensaje son necesarios dos componentes:
-*Un algoritmo*: es una secuencia de instrucciones a las que será sometido el mensaje para cifrarlo o codificarlo.
-*Una clave:* una variable que se inserta en el algoritmo, para lograr el resultado deseado.
Mismo mensaje + tratado con un mismo algoritmo, pero codificado con 2 claves distintas, debería producir 2 resultados diferentes.

Tipos de algoritmos criptográficos:
-Transposición: Algoritmo que se basa en dividir un mensaje, cifrarlo en columnas y luego transcribirlo usando el nuevo orden de los caracteres dado por esas columnas.
cantidad de columnas = clave que se utilizará para codificar y decodificar el mensaje.
Mensaje a cifrar: 'Este mensaje es secreto' Clave: Utilizar una matriz de 5x4.
Mensaje codificado: 'EEER SNEE TSST EAEO MJC'
ESTEM
ENSAJ
EESEC
RETO
-Sustitución: produce un mensaje cifrado a partir del reemplazo de caracteres de un mensaje por otros, el elemento que establece las reglas para el reemplazo es la clave.
Por ejemplo: establecer un desplazamiento de dos caracteres en el alfabeto, de este modo todas las letras 'A' del mensaje a cifrar se reemplazan por letras 'C'; este algoritmo se lo conoce cómo cifrado César o cifrado por desplazamiento.
Mensaje a cifrar: 'Este mensaje es secreto' Clave: Desplazamiento de 4 caracteres
Mensaje codificado: 'IWXI QIRWENI IW WIGVIXS'

-Ocultación: El objetivo es esconder el mensaje en otro mensaje u objeto. La clave en este caso son las indicaciones que permiten al individuo encontrar el mensaje. Ejemplo: la primera letra de cada oración contendrá el mensaje.

-Esteganografía: es una forma de ocultación, pero vale la pena mencionarla por separado ya que es una forma habitual de cifrar mensajes. Consta de ocultar un mensaje dentro de un archivo de datos -imagen, audio- sin alterar el contenido original del archivo.

La criptografía en la historia (ver video)

La criptografía en los sistemas:
En el ámbito de la **informática** puede ser utilizada para tres propósitos distintos:
1) Confidencialidad → Implementar confidencialidad para proteger los datos y sistemas de accesos no autorizados

2) Integridad → Proteger los datos, recursos y sistemas de cambios no autorizados y así asegurar confiabilidad.
3) Disponibilidad → Garantizar que aquellos usuarios autorizados tengan acceso a los datos, recursos y sistemas que necesitan
En **criptografía** la letra “A” → nuevo significado → Autenticar → verificar la identidad de un sujeto.

La criptografía cómo vector de ataque:
Desde mediados de la década del 2010, la criptografía se comenzó a popularizar cómo un vector de ataque informático, esto es a través de lo que se conoce cómo ‘ransomware’.

Ransomware: Es un software que encripta la información de un individuo y organización con propósitos extorsivos. Son cifrados utilizando un algoritmo complejo y una clave únicamente conocida por el autor del ataque. El ámbito de ataque puede limitarse a una computadora en particular o a toda una red.
Esto puede suceder tentando al usuario a descargar un archivo desde internet y luego ejecutarlo, mediante un correo electrónico con links que lleven al usuario a descargar el ransomware, o con un dispositivo USB infectado.
Una vez que el ransomware comienza a ejecutarse en el dispositivo, localizara los archivo que está programado para encriptar. Deja intactos los archivos del sistema operativo. De esta manera garantiza el correcto funcionamiento del mismo, y así permite al usuario describir los efectos del ransomware y acceder a la información necesaria para realizar el pago.

¿Cómo defenderse de un ransomware?
Tomar acciones antes del ataque, de no hacerlo, una vez sucedido, no hay mucho que se pueda hacer.
*Educar al usuario a:
-No abrir correos sospechosos,
-Identificar y reportar correos sospechosos al departamento de sistemas.
-No insertar dispositivos USB de los que se desconoce su procedencia.
*Tener una red debidamente segmentada, en donde los usuarios no compartan el mismo espacio de la red que los servidores.
*Mantener todos los sistemas operativos actualizados, protocolos deprecados desactivados y vulnerabilidades remediadas.
*Buena estrategia de backup, sólida, y que permita recuperar de nuestros respaldos cualquier archivo que haya podido ser cifrado con propósitos extorsivos.

Criptografía en la práctica:

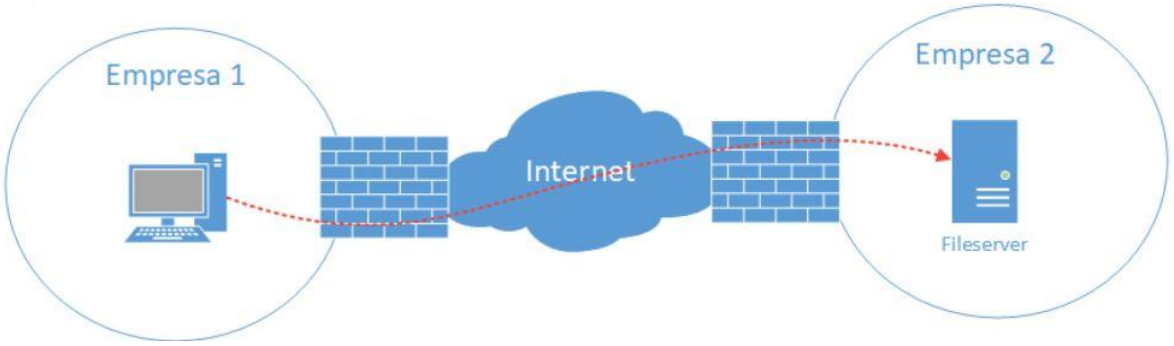
Uno de los usos principales de la criptografía es habilitar la confidencialidad, en otras palabras, mantener datos o información ininteligibles a los ojos no autorizados.
Los datos pueden tener 3 estados:
*Data at rest → en reposo, no están siendo consultados. Ejemplo: Registro de una base de datos
*Data in transit → datos viajando de los sistemas que los albergan en reposo hacia el cliente o sistema que los haya consultado.

*Data in use → datos siendo visualizados en el cliente o sistema que los haya consultado.

Protección de Datos según su estado:
Desde la perspectiva de la seguridad, dependiendo del estado de los datos se pueden implementar distintas técnicas o medidas para segurizarlos. Estos se pueden agrupar en dos grandes categorías: Seguridad Física y Seguridad Lógica. Algunos ejemplos:

	Seguridad Física	Seguridad Lógica
*Date at rest →	Centrales de acceso físicas Cámaras de seguridad	-Implementar mecanismos de encriptación en los file systems.
*Data in transit →	Evitar redes wifi(cableadas) Encriptación de dispositivos de de almacenamiento removibles	-Implementar una VPN -Encripción de los datos en tránsito (con TLS/SSL)
*Data in use →	Limitar el acceso a las áreas de trabajo a sólo personal autorizado	-Encripción de los datos en memoria. -Un sistema de gestión de id robusto.

Datos en tránsito: Criptografía simétrica
Tomemos en consideración este escenario: las empresas 1 y 2 están conectadas por medio de una VPN. Una computadora en la Empresa 1 necesita acceder a un recurso en la Empresa 2. Vamos a ver cómo la criptografía asimétrica nos puede ayudar en este escenario.



Criptografía simétrica: el paso a paso.
Ambas empresas se ponen de acuerdo en el algoritmo a utilizar y la clave que utilizarán.
Por cada mensaje que la Empresa 1 enviara, se produce un paquete cifrado, utilizando el algoritmo definido, la clave compartida y los datos a intercambiar.
Cuando la Empresa 2 recibe el mensaje cifrado, utiliza el mismo algoritmo y clave para realizar la operación inversa y así obtener el mensaje original.

Desafíos de la criptografía simétrica.
La clave a utilizar para codificar los mensajes debe ser conocida por todas las partes.
Un ejemplo podría ser:
Visitar un sitio de home banking desde nuestras casas. El banco no sabe desde que computadoras o redes accederemos al sitio, y aun asi es su responsabilidad asegurar que las comunicaciones sean protegidas.

Datos en tránsito: Criptografía Asimétrica.
Se hace uso de dos claves.

Ambas claves son generadas matemáticamente y en conjunto. De este modo, los mensajes que son codificados con la clave 1, pueden ser codificados con la clave 2 y viceversa.
En los pares de claves, una se conoce por el nombre de clave pública mientras que la otra se conoce por el nombre de clave privada.

Certificados.
Acceder a un sitio de home banking desde nuestras casas. Para poder proteger la comunicación, el banco debe distribuir la clave pública a quien sea que visite el sitio. Y esto es posible gracias a los certificados.

¿Que tiene un certificado adentro?

- Una forma de identificación del sitio que implementa el certificado (IP o nombre de la página)
- Las fechas entre las cuales será válido el certificado(**los certificados expiran!**)
- La clave pública que se usará para codificar la comunicación.
- La firma de la entidad certificante que emitió el certificado.
- Otros datos que hacen a la seguridad del certificado.

Entidad certificante
Para poder cumplir con el principio de autenticación, no cualquier puede emitir un certificado. Entidades conocidas cómo Certification Authorities son las encargadas de verificar que quien solicita el certificado es realmente quien dice ser. Si el certificado va a ser utilizado de forma externa(cómo en el caso del sitio de Home banking), entonces el certificado debe ser emitido por una organización pública que sea conocida.

Certificado Auto-Firmados (self-signed)
Existe la posibilidad de generar certificados auto-firmados. Sirve para hacer pruebas.

Criptografía asimétrica: el paso a paso.

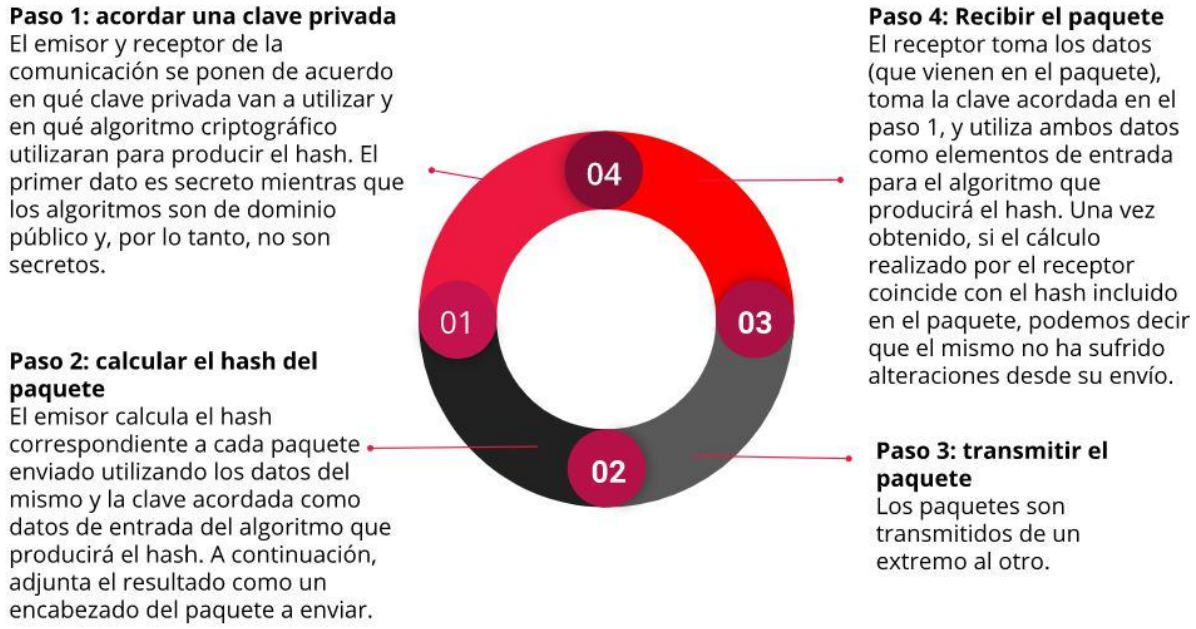
1. El cliente ingresa a la URL de home banking.
2. El servidor envía al cliente el certificado.
3. El navegador en el cliente verifica que el certificado sea válido (firmado por una entidad confiada, que no haya expirado y que la URL que estamos visitando coincida con la especificada en el certificado). De no cumplirse alguna de estas condiciones el navegador nos va a indicar que el certificado no es seguro.
4. Una vez que el navegador tiene el certificado, extrae la clave pública para poder intercambiar mensajes de manera segura con el servidor. Pero no es esta clave la que se va a usar para codificar cada mensaje enviado y decodificar cada mensaje recibido. Sino que ahora que pueden establecer una comunicación, lo que hacen es generar y negociar una ‘session key’. Y es esa ‘session key’ la que se utilizará para cifrar el resto de la comunicación.

Usar la criptografía para validar integridad:
Validar la integridad.
El dato para el cual queremos validar su integridad es sometido a un algoritmo (los más conocidos para este propósito son SHA y MD5), el resultado es una cadena de caracteres que se conoce cómo **hash**.
De alterarse los datos, al volver a calcular el hash, estos producirían un resultado diferente.

Integridad para "Data at Rest' → ejemplo, un repositorio de archivos.
La forma más habitual de garantizar la integridad de los datos es calcular el hash, almacenarlo. En una instancia posterior cuando el dato deba ser consultado, se puede verificar el hash y así determinar que los contenidos del archivo no fueron alterados.

Integridad para "Data in transit" → compu se conecta x medio de una VPN a un servidor. Por cada paquete que vamos a enviar mediante la red, podemos calcular el hash que le corresponde y agregar el resultado a uno de los encabezados del paquete. El servidor recibe el paquete y lo somete al mismo algoritmo y de producir el mismo hash → podemos determinar que el paquete es íntegro.

HMAC(hash message authentication code)
Se usa en escenarios de datos en tránsito donde podemos usar una clave simétrica cómo entrada adicional para el algoritmo de hashing. Para producir el hash ambos lados (emisor y receptor) ambos deben conocer la clave.

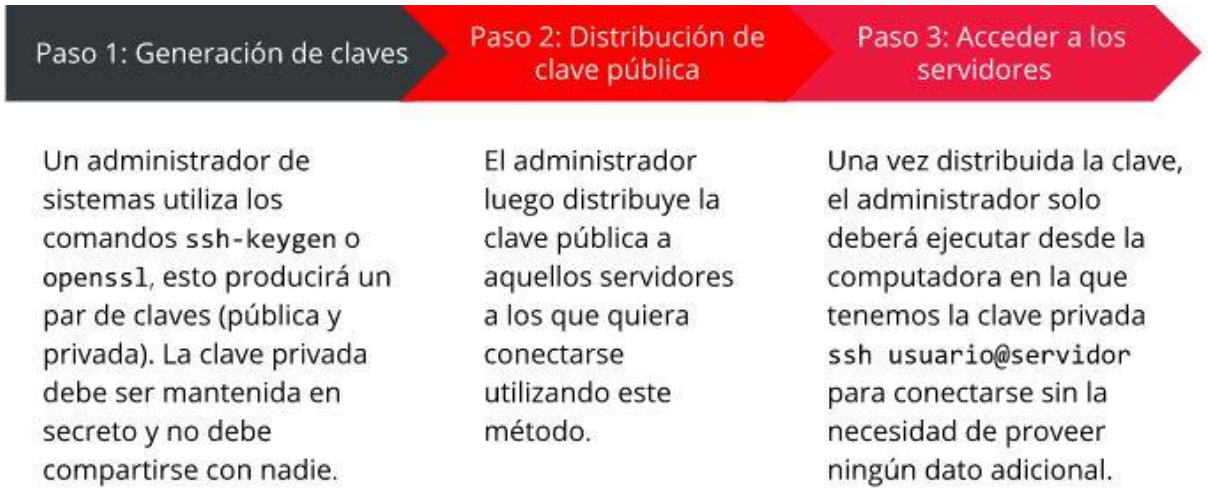


Usar la criptografía cómo mecanismo de autenticación
Al utilizar comunicaciones encriptadas de forma simétrica (confidencialidad) con una clase que es solo conocida por los receptores y emisores (autenticación) y sobre la que se monta HMAC(integridad) estamos cumpliendo con el principio de seguridad CIA → asegurar la confidencialidad, verificar la integridad y autenticar las partes.

Usar certificados SSL:
Hay organizaciones que deben proteger las comunicaciones contra sus sitios (bancos). Deben hacer uso de la criptografía asimétrica y el mecanismo de distribución para la clave pública son los certificados. Para que sean confiables → emitidos por una entidad certificante → para que genere el certificado en nombre de una compañía → poder verificar su identidad.Los certificados SSL sirven para verificar la identidad de una compañía y así cumplir con el principio CIA.

Usar pares de claves asimétricas:
En Linux para conectarnos de forma remota utilizar comando y protocolo SSH(Secure Shell) El administrador debe ingresar el comando ssh usuario@servidor en una consola para conectarse a otro servidor. Usuario deberá ingresar su contraseña. Usuario + password. Linux provee una alternativa: utilizar claves criptográficas asimétricas para autenticarnos.

¿Cómo utilizar pares de claves asimétricas?



Glosario de criptografía (VER PDF)

CLASE 7 SEMANA 3

SHELL SCRIPTING - INTRODUCCIÓN A LA TERMINAL DE LINUX

La interfaz de línea de comandos (CLI) → método de comunicación entre usuario y máquina que acepta instrucciones del usuario a través de líneas de texto → según reglas de sintaxis que puedan ser interpretadas por el S.O. → Shell → herramienta que posibilita la funcion de interfaz de usuario.

Diferentes tipos de Shell
-Shell Bourne → primera shell utilizada para el S.O. Unix. Todas las versiones de Linux Unix permiten a los usuarios cambiar esta shell, también conocida cómo "**sh**". No tienen funcionalidades cómo: completado de nombres de archivo + historial de comandos.

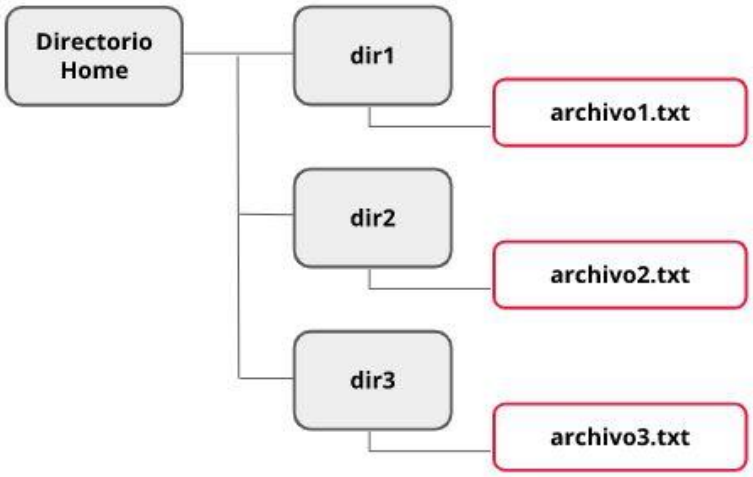
-Shell C / TC → Posterior al Bourne shell, para facilitar el control del sistema al programador en lenguaje C. Su sintaxis es similar a este lenguaje. "**csh**" → presente en otros S.O (Mac/OS). Posee una evolución → "**tsh**" → incorpora funcionalidades avanzadas y mayores atajos de teclado.
-Shell Korn → Intenta combinar las características de la Shell C, Shell TC y Shell Bourne en un solo paquete. Incluye capacidad para crear nuevos comandos s/ necesidad. Posee funciones avanzadas p/ manejar archivos de comandos → la colocan a la par a lenguajes cómo → awk y perl.
-Shell Bourne-Again (BASH) → versión actualizada de la Shell Bourne original. Utilizada en la comunidad de código abierto. Incorpora funcionalidades + avanzadas que tienen la C, TC y Korn → capacidad p/ *completar nombres de archivos con TAB, *recordar un historial de comandos recientes y *ejecutar múltiples programas en segundo plano a la vez.

Ejecución de la consola
-Ejecución en Inicio → Cuando el S.O. se inicia en los niveles 1,2,3 y 4 nos lleva por defecto a la consola.
-Ejecución desde GUI → Si el S.O inicia en nivel 5 (con GUI) p/ poder utilizar la terminal tenemos distintas opciones → varían s/ distribución instalada.
*Ubuntu → 2 opciones:
→ Lanzando un TTY o espacio de trabajo sin entorno gráfico. Podemos ejecutar 7 terminales al mismo tiempo. 1-6 no tienen interfaz gráfica. P/ cambiar de TTY en Linux → Ctrl + Alt + F1 a F7 (s/ cual TTY queramos ejecutar)
→ P/ usar la terminal cómo tal → encontrar una app dedicada (se ejecuta en una ventana) dentro del panel de app de nuestra distro. Ubuntu → podemos encontrar esta terminal dentro del cajón de programas del entorno gráfico→ GNOME

Elevación de privilegios
Privilegios del superusuario root → En Linux → se separa la cuenta de usuario común de la superusuario → root → posee todos los privilegios y permisos para realizar acciones sobre el sistema.
P/ ejecución de algunos comandos debemos ingresar dicho acceso (clave de root) → tener conocimiento sobre las acciones que realizan → p/ evitar daños importantes en el sistema.
Usar el comando sudo, previo al comando a ejecutar → pedirá contraseña de root y se ejecuta cómo root satisfactoriamente.

COMANDOS

Consolidando nuestro ambiente
Ambientes = M.V. o WSL. Tener replicada esta estructura de carpetas y archivos:



$ → prompt. Agregar estas líneas de código para lograrlo:
mkdir dir1 dir2 dir3
touch dir1/archivo1.txt dir2/archivo2.txt dir3/archivo3.txt

Verificando nuestro ambiente
Listamos los directorios → ls -R → obtenemos:

```
edorio@DESKTOP-W10:~$ ls -R
.:
dir1  dir2  dir3
./dir1:
archivo1.txt
./dir2:
archivo2.txt
./dir3:
archivo3.txt
```

Comandos para el manejo de archivos
ls → listar directorios y archivos→ de la carpeta de trabajo en la que estas.
ls -a → en forma de lista todo el contenido que se encuentre dentro del directorio de trabajo.(incluye archivos y carpetas ocultos)

```
edorio@DESKTOP-W10:~$ ls -a
.          .aws           .bash_logout   .config     .landscape
.profile                   .vagrant.d  dir3 ..          .azure
.bashrc        .docker     .local        .ssh     dir1 .ansible
.bash_history  .cache         .fastlane   .motd_shown
.sudo_as_admin_successful  dir2
```

ls -l → muestra el contenido en forma de lista e incluye info referente a c/ elemento.

```
edorio@DESKTOP-W10:~$ ls -l
total 12
drwxr-xr-x 2 edorio edorio 4096 May 21 01:59 dir1
drwxr-xr-x 2 edorio edorio 4096 May 21 01:59 dir2
drwxr-xr-x 2 edorio edorio 4096 May 21 01:59 dir3
```

mkdir → crea directorios c/ nombre y ruta que especifiques → mkdir dir1/subdir1 → path
rmdir → borra directorios que deben estar vacíos → rmdir dir4 ó rmdir dir1/subdir1 → path
rm → borra directorios no vacíos y archivos →
rm dir1/archivo1.txt → elimina 1 archivo específico dentro de dir1
rm -r → eliminamos el directorio y todo su contenido. → rm -r dir2
cp → copia archivos y directorios y/o ubica en otras rutas →
cp dir3/archivo3.txt dir1/archivo1.txt → renombramos el archivo
cp -r → copiamos el directorio dir3 en uno llamado dir2(el mismo comando lo crea) →
cp -r dir3 dir2
mv → mueve archivos y directorios. Ubicación inicio+nombre archivo + ubicación destino.
mv dir1/archivo1.txt dir3/archivo1.txt / mv dir3/archivo1.txt dir3/archivo3bis.txt → renombra.

Comandos para leer archivos de texto
cat → leer y modificar archivos de texto (formato .txt) desde la terminal. Crear un archivo, imprimir por pantalla su contenido.
cat >dir1/archivo1.txt → abrirá ese archivo. Ctrl D se guarda la edición.
cat dir1/archivo1.txt //Hola Digital House,esto es CAT→sin “>” muestra contenido en pantalla
-n → numerar las líneas -b → no muestra las líneas en blanco.
more → leer archivos visualizandolo x páginas al contenido → + adecuado p/ leer archivos largos. more /var/log/dpkg.log
nano → Editor de textos en la terminal → leer, modificar y editar archivos.
nano dir1/archivo1.txt
grep → Busca un patrón en archivos. 1ero cadena de texto a buscar + archivo a donde buscar (acepta *) -r recorrer recursivamente → grep “Digital House” * -r
tee → leer una entrada y escribirla→ ls -l | tee listado.txt→muestra dir + guarda en archivo
ls -l | -a listado.txt → c/ modificacor “a” se agrega el contenido al archivo sin pisar lo anterior.

Obtener datos desde un web service
Terminal Linux tiene mucha versatilidad y potencia→nos permite vincularla c/1 Webservice, obtener datos de allí y procesarlos con propósitos cómo agregarlo a archivos en nuestro servidor, modificarlos y republicarlos. Obtener un JSON desde una URL externa, procesar su contenido, obtener el atributo que interesa y crear nuevos archivos, insertar a base de datos.
curl → Aspectos técnicos. <<Client URL>> Diseñados para funcionar como una forma de verificar la conectividad a las URL y gran herramienta para transferir datos. Amplia compatibilidad con protocolos: FTP, HTTP, POP/SMTP/IMAP, SCP.
curl https://www.digitalhouse.com → muestra la página de inicio de digital house.
curl https://www.digitalhouse.com **-o** mipagina.html →guardará todo el HTML en ese archivo
-Descargas
Puede extenderse a procesar descargas (ISO de la URL de referencia y nombrarla)

curl https://ubuntu.zero.com…. -O -C 0 → no renombra el archivo de destino (“-O”) y permite continuidad de la descarga con “-C”
-Encabezados y verificaciones
curl https://www.digitalhouse.com **-v** → brinda contenido + IP de destino, protocolos de seguridad y certificados utilizados.
curl https://www.digitalhouse.com **-I** → muestra encabezados de la solicitud → ruta por defecto, publicador web.
-Contenido JSON
obtener el contenido en formato JSON desde un endpoint que lo entregue en dicho formato.
curl **“**https://nominatim.openstreetmap.org…**”** -o resultado.json → guardamos lo obtenido en el web service en el archivo resultado.json. URL entre “ ”

jq → potente procesador JSON para la consola. JSON→ naturaleza liviana+compatibilidad con Javascript. Se basa en el concepto de filtros que funcionan sobre un flujo de JSON. Tomando el archivo json obtenido en curl, una ejecución sencilla de jq nos devuelve todo el contenido del JSON → jq ‘**.**’ resultado.json → no accedemos a un atributo específico → por el modificador ‘**.**’ Para acceder a propiedades:
jq ‘.display_name**,**.type’ resultado.json → p/ acceder a + de 1 propiedad usamos la coma.
TIP: si alguna propiedad tuviese un espacio en su nombre → envolverla con comillas dobles
-Para combinar uso de ambos comandos (curl + jq)→ uso de pipelines → |
Pipeline o tubería → función que permite utilizar la salida de un programa cómo entrada de en otro.
ip address | grep “192.168” → ip nos devuelve muchos datos, para filtrar esa cadena→ grep

curl p/ obtener el JSON + jq p/ obtener el valor de la propiedad + tee p/ guardar en 1 archivo

```
edorio@DESKTOP-W10:~$ curl
"https://nominatim.openstreetmap.org/reverse.php?lat=-34.60378&lo
n=-58.38161&zoom=18&format=jsonv2" | jq ".display_name,.type" |
tee consultapipe.txt
```

La sentencia obtiene el JSON con curl, lo procesa con jq p/ obtener el display_name y el type y lo guarda en un archivo llamado consultapipe.txt

BASH

Bash es una herramienta **open source** perteneciente al proyecto GNU.
Bash hereda muchas propiedades de otras shells como sh, csh o zsh.
Actualmente es la interfaz de línea de comando predeterminada en la mayoría de distribuciones GNU/Linux así como en macOS.
Como ocurre con otras shells de Unix, además de **intérprete de comandos, Bash es también un lenguaje de scripting.** Esto lo hace extremadamente potente para una multitud de tareas relacionadas con **la administración de sistemas, automatización de tareas**. Para ejecutar múltiples comandos en un solo paso desde el shell, podemos escribirlos en una línea y separarlos con punto y coma.

-Como empezamos a escribir en BASH
Primero, creamos un nuevo archivo usando el comando **touch**. Al comienzo de cualquier

script de Bash, debemos definir **qué shell usaremos** porque hay muchos shells en Linux, Bash shell es uno de ellos. La primera línea que escribimos al escribir un script bash es el (#!) seguido del shell que utilizaremos.
#! <=== este signo se llama shebang.
#!/bin/bash

Si utilizamos el signo de numeral (#) delante de cualquier línea en su script de Bash, esta línea se comentará, lo que significa que no se procesará, pero la línea anterior es un caso especial.
Esta línea define **qué shell utilizaremos**, que es el Bash de shell en nuestro caso. Los comandos de shell se ingresan uno por línea, como el siguiente ejemplo:

```
#!/bin/bash
# This is a comment
pwd
whoami
```

Podemos escribir múltiples comandos en la misma línea, pero debemos separarlos con punto y coma, aunque es **preferible escribir comandos en líneas separadas**, esto hará que sea más fácil de leer luego.

-Establecer los permisos de scripts
Después de escribir el script de Bash, guardamos el archivo.
Ahora, configuramos ese archivo **para que sea ejecutable**, de lo contrario, mostrará permisos denegados.
chmod +x ./myscript

Luego intentamos **ejecutarlo** simplemente escribiendo en el shell:
./myscript

Tipos de variables:
-Variables globales o de entorno → en LETRAS MAYÚSCULAS. Para ver las variables de entorno que estamos usando y que están cargadas en nuestra sesión → **printenv** ó **env** en la shell. Para declarar una variable global→ **export** NOMBREVARIABLE=valor
Para acceder a la misma → **$**NOMBREVARIABLE
-Variables de usuario o locales → pueden ser accedidas solo por el usuario y la sesión en la que fueron creadas. Se declara → nombrevariable=valor Para acceder→ $nombrevariable

Estructuras de control:

if-then → if command; then
hacer algo
fi

```
#!/bin/bash
if whoami; then
    echo "It works"
fi
```

if-else → if-then-else statement
Si el comando se ejecuta y retorna cero → éxito → no ejecuta los comando después de la sentencia else, pero si la sentencia if retorna un número distinto de cero (la condición no se cumplió) → el shell ejecuta los comando después de la sentencia else.

```
#!/bin/bash
ping -c 1 8.8.8.8
if [ $? -ne 0 ]; then
echo "No está en red"
else
echo "Sí está en red"
fi
```

Nos devuelve un mensaje si el comando ping fue satisfactorio.

Comparaciones numéricas:

number1 -eq number2	Comprueba si number1 es igual a number2.
number1 -ge number2	Comprueba si number1 es más grande o igual number2.
number1 -gt number2	Comprueba si number1 es más grande que number2
number1 -le number2	Comprueba si number1 es más pequeño o igual number2
number1 -lt number2	Comprueba si number1 es más pequeño que number2
number1 -ne number2	Comprueba si number1 no es igual a number2

```
#!/bin/bash
num=11
if [ $num -gt 10]; then
    echo "$num is bigger than 10"
else
    echo "$num is less than 10"
fi
```

Comparaciones de cadenas entre dos valores alfanuméricos:

string1 = string2	Comprueba si string1 es idéntico a string2
string1! = string2	Comprueba si string1 no es idéntico a string2
string1 <string2	Comprueba si string1 es menor que string2
string1> string2	Comprueba si string1 es mayor que string2
-n string1	Comprueba si string1 es más mayor que cero.
-z string1	Comprueba si string1 tiene una longitud de cero

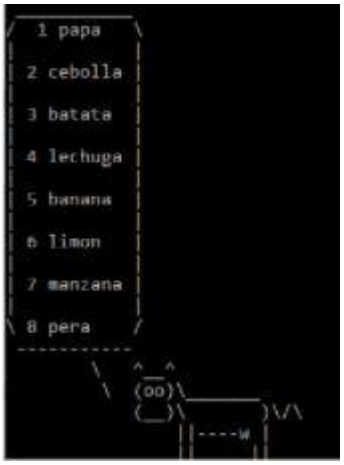
Si estamos logueados cómo root, va por el if, sino por el then.

```
#!/bin/bash
user="root"
if [ $user = $USER ]; then
    echo "The user $user  is the current logged in user"
fi
```

Cálculos matemáticos:
Utilizando la sintaxis $((2+2))

```
#!/bin/bash
var1=$(( 5 + 5 ))
echo $var1
var2=$(( $var1 * 2 ))
echo $var2
```

sudo apt-get update
sudo apt-get install cowsay → instala paquete nuevo llamado cowsay
mkdir carpetascript → creamos carpeta en nuestro equipo
cd carpetascript → nos posicionamos en ella
cat >lista verduras→ creamos el archivo y lo abrió,agregar verduras,finaliza edicion c/ Ctrl D
cat >lista frutas → idem arriba
cat -n lista_verduras lista_frutas → verificamos la creación de ambas listas.
nano listacompras.sh → creamos un nuevo archivo→ será nuestro script. Se abrirá nano con documento en blanco y escribimos:
#!/bin/bash
cat -n lista_verduras lista_frutas | cowsay → Propósito del script → listar los archivos y pasar dicho output a cowsay. Salimos de nano con Ctrl X.
chmod +x listacompras.sh → damos permisos de ejecución a nuestro script.
./listacompras.sh → ejecutamos el script

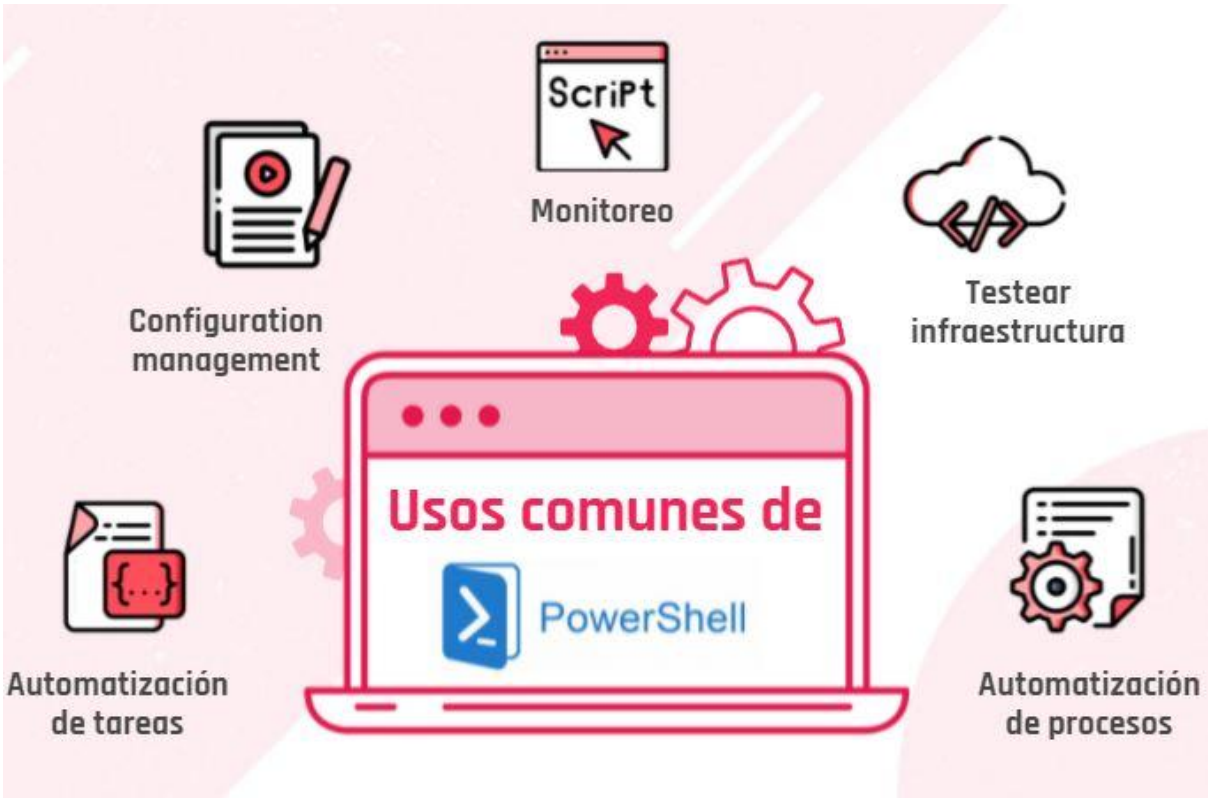


CLASE 8 SEMANA 3

INTRODUCCIÓN A POWERSHELL

Powershell → automatización de tareas formada por una línea de comandos. un lenguaje de scripting y un marco de administración de configuración.
Como lenguaje de scripting → se usa para automatizar la administración de sistemas.

	PowerShell	PowerShell Core
Versión más reciente	5.1	7.1
Plataformas	Solo Windows (cliente y Servidor)	Windows (Cliente y Servidor), Mac OS y Linux
Dependencia	.NET Framework	.Net Core
Uso	Se basa en el runtime de .Net Framework	Se basa en el runtime de .Net Core
Ejecutado como	powershell.exe	pwsh.exe (Windows), pwsh (Mac y Linux).
Verificar el contenido de la propiedad $PSVersionTable.PSEdition	Devuelve: 'Escritorio'	Devuelve: 'Core'
Políticas de actualización	Solo correcciones de errores críticos	Todas las actualizaciones (características, errores)



El pipeline,
Serie de comandos conectados mediante operadores de canalización. Cada operador de pipeline envía los resultados del comando anterior al siguiente comando. En un pipeline los comandos se procesan de izquierda a derecha.

variables,
unidad de memoria en la que se almacenan los valores.Se representan en Powershell mediante cadenas de texto, comienzan con $ Tipos de variables:
-Variables creadas por el usuario → si se crean en la línea de comandos sólo existen mientras la ventana de Powershell está abierta.
-Variables automáticas → las crea Powershell, el usuario no puede cambiar el valor.
-Variables de preferencia → las crea Powershell y los usuarios pueden cambiar los valores.

estructuras de control,
Permiten modificar el flujo de ejecución de las instrucciones de un programa. Se puede:
-If-Then-Else → De acuerdo con una condición, ejecutar un grupo u otro de sentencias.
-Switch-Case →De acuerdo con el valor de una variable, ejecutar un grupo u otro “ “
-Do-While → ejecutar un grupo de sentencias sólo cuando se cumpla una condición
-Do-Until → ejecutar un grupo de sentencias hasta que se cumpla una condición
-For-Next → ejecutar un grupo de sentencias un número determinado de veces
Tienen un único punto de entrada las estructuras de control. Se pueden clasificar en: secuenciales, iterativas y de control avanzadas.

scripts
Archivo de texto sin formato que contiene uno o más comandos de Powershell. Extensión de archivo → .ps1

y funciones.
Conjunto de instrucciones a las que damos un nombre. Podemos llamarlas varias veces sin tener que volver a escribir las instrucciones en cada llamada.

Módulos de Powershell
Paquete que contiene objetos de PowerShell, como cmdlets, proveedores, funciones, flujos de trabajo, variables y alias. Los objetos u acciones de este paquete se pueden implementar en un script de PowerShell, una DLL compilada o una combinación de ambos. Por lo general, estos archivos se agrupan en un solo directorio.
Las personas que escriben comandos pueden usar módulos para organizar sus comandos y compartirlos con otros.

Carga automática de los módulos
A partir de PowerShell 3.0, PowerShell importa módulos automáticamente la primera vez que ejecuta cualquier comando en un módulo instalado. No es necesario administrar los módulos después de instalarlos en su computadora.

Los comandos de un módulo también son más fáciles de encontrar. El cmdlet **Get-Command** obtiene todos los comandos en todos los módulos instalados, incluso si aún no están en la sesión. Pueden encontrar un comando y usarlo sin importar la necesidad de importar el módulo primero. Solo los módulos que se almacenan en la ubicación especificada por la variable de entorno PSModulePath se importan automáticamente.
Los comandos que incluyen un carácter comodín (*) se consideran para descubrimiento, no para uso y no importa ningún módulo.
Los módulos en otras ubicaciones deben importarse ejecutando el Import-Module cmdlet.
Módulos centrales → módulos preinstalados.
Get-Command -Module <module-name>
VER ÚLTIMO PPT.

CLASE 10 SEMANA 4

PYTHON

Python → lenguaje de programación interpretado cuya principal filosofía es que sea legible por cualquier persona con conocimientos básicos de programación.

Características del lenguaje:
- Es totalmente gratuito → lenguaje open source, de código abierto. Se están desarrollando nuevas librerías y aplicaciones.
- Lenguaje multiparadigma → Combina propiedades de diferentes paradigmas de programación, muy flexible y fácil de aprender de manera independiente.
- Apto para todas las plataformas → Podemos ejecutarlo en diferentes S.O. usando el intérprete correspondiente.

El intérprete de Python → Cuando lo instalamos → se añade el comando python al path y se instala el intérprete de Python.
Una vez instalado → abrir consola y ejecutar el comando python3 y lanzará el intérprete.
Apareció en 1991.

¿Por qué es importante saber Python en el contexto de infraestructura?
Principales industrias:

- Inteligencia artificial (AI) → ventajas → de escritura rápida, escalable, robusto y de código abierto. Permite plasmar ideas complejas c/pocas líneas de código.
- Big data → muy extendido en el análisis de datos y la extracción de información útil para empresas.
- Data science → se ocupa de los datos tabulares, matriciales y estadísticos, e incluso los visualiza con bibliotecas populares.
- Frameworks de pruebas → ideal para validar ideas o productos → tiene muchos frameworks integrados que ayudan a depurar el código y ofrecen flujos de trabajo y ejecución rápidos.
- Desarrollo web → permite construir mucho más con menos líneas de código → se crean prototipos de forma más eficiente

Usos comunes →Desarrollo web, videojuegos, análisis de datos e Internet de las cosas(IoT)
Fue usado para desarrollar Instagram, Youtube, Google.

Pros:

+Aprendizaje → Fácil de aprender.
+Académico → Elegido por universidades
+Librerías → extensa cantidad de módulos (escritos en C) como apoyo.

Contras:

-Rendimiento → no es comparable con C/C++
-Desarrollo móvil → no es un área fuerte del lenguaje.
-Errores → Algunos errores se pueden detectar solamente en tiempo de ejecución.

Variables → En Python son “etiquetas” que permiten hacer referencia a los datos (que se guardan en unas “cajas” llamadas objetos) Lenguaje de programación orientado a objetos y su modelo de datos tb.
Cada objeto tiene:
-Un identificador único.
-Un tipo de datos.
-Un valor (el propio dato)
Las variables se crean cuando se definen por primera vez, cuando se les asigna un valor.

Estructuras de control:
-Indentación → recomienda usar bloques de cuatro espacios.
-if → condición principal, elif → condiciones adicionales, else → una vez y al final.
-Condiciones → ==, < , >, not (NO:niega la condición que le sigue), and (Y:junta dos condiciones que tienen que cumplirse las dos) or (O:junta dos condiciones y tiene que cumplirse alguna de las dos)
-Bucles → While → Para repetir una determinada tarea hasta conseguir nuestro objetivo. Tener la precaución de no realizar bucle infinito → vuelta=vuelta+1.

```
while vuelta<10
        print(“Vuelta “+str(vuelta))
        vuelta=vuelta+1
```

→ for → se puede utilizar con cualquier objeto con el que se pueda iterar.

```
for vuelta in range(1,10):
        print("Vuelta " +str(vuelta))
coches = ("Ferrari", "Tesla", "BMW", "Audi" )
for i, coche in enumerate(coches):
        print(str(i) + "-" + coche)
```

CLASE 13 SEMANA 5

CONFIGURATION MANAGEMENT

y change management → Son dos de los procesos fundamentales del conjunto de metodologías conocido cómo ITIL

Change management → ITIL la describe como el proceso de controlar y gestionar un cambio a lo largo de todo su ciclo de vida con el objetivo de minimizar el riesgo.
-Un cambio es la modificación o eliminación de cualquier cosa que pueda afectar directa o indirectamente los servicios → Cualquier cambio en la infraestructura de IT de una organización puede afectar las operaciones de la organización.
-Ejemplos → reemplazo de hardware, instalación de software en un servidor.

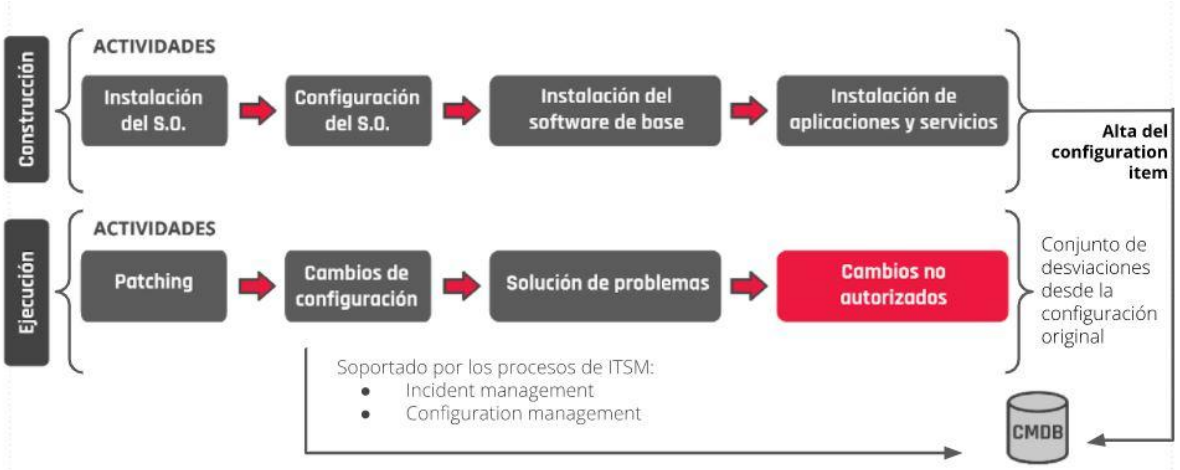
Configuration management → Proceso que permite gestionar los cambios de configuración de nuestros activos informáticos, permitiendo a la organización mantener un registro histórico y a su vez aplicar controles.
Cada uno de los activos informáticos en el contexto de este proceso se los conoce cómo **configuration item(CI)** y se almacenan en lo que es llamado CMDB (configuration management database)

Ambos procesos se complementan, → **change management** aporta la gobernabilidad de los cambios que se efectúan en el parque tecnológico de una organización, y **configuration management** nos permite gestionar una base de datos (CMDB) con la información de nuestros activos (CIs) y un historial de los cambios realizados para cada uno de ellos.

De ocurrir una falla en uno de los activos que nos fuercen a restaurarlo (falla crítica en un servidor) el proceso de configuration management nos aporta una vista de todos los cambios sucedidos para ese activo desde su creación → permite reproducir cada uno de los cambios y devolverlo al estado anterior a la falla.

Configuration management tradicional
Ejemplo ciclo de vida de un servidor:

- Construcción → conjunto de pasos que pueden ser automatizados, documentados, que permiten la puesta en marcha y registrarlo en la CMDB como un nuevo CI.
- Ejecución → ciclo de vida, todo lo que suceda en relación al activo y cambios que puedan suceder en él.



Etapa de construcción de un servidor

Instalación del S.O.	Configuración del S.O.	Instalación del software de base	Instalación de aplicación y servicios
El proceso de instalación del sistema operativo en el hardware o máquina virtual.	El conjunto de configuraciones definido por la organización como requeridas para el que el servidor pueda formar parte del ecosistema.	Es probable que la organización de IT haya determinado que debe instalarse software adicional para que el servidor pueda operar correctamente dentro del ecosistema. Estos pueden incluir: antivirus, software de IDS / IPS, agentes de backup, etcétera.	Finalmente, durante el proceso de construcción del servidor se debe instalar y configurar el software o servicio que dará sentido a su existencia. Por ejemplo, si el servidor está destinado a ser un servidor web, no solo se trata de instalar el servidor web (IIS, NGINX, Apache, etc.), sino que también se trata de desplegar el sitio web que lo alojará.

Etapa de ejecución de un servidor

Patching	Cambios de configuración	Solución de problemas	Cambios no autorizados
El proceso de instalar actualizacion del sistema operativo, dependiendo del sistema operativo es la cadencia con la que este proceso se ejecutará. En términos generales las organizaciones eligen ejecutar instalaciones mensuales de actualizaciones. Este proceso debería verse reflejado mediante el proceso de change management.	Cualquier cambio de configuración del sistema operativo o hardware que responda a una necesidad. Estos deberían verse reflejados como parte del proceso de change management.	Cualquier incidencia que haya producido una falla en el sistema reduciendo o anulando su funcionamiento. Requiriendo la intervención de un administrador para restaurar el nivel de servicio. Estos deberían verse reflejadas como parte del proceso de incident management.	**¡Alerta!** Cualquier cambio o actividad realizado en el servidor y que no queda documentado en un ticket, ya sea como un cambio o un incidente.

Configuration management + Change management , ¿es practicable?
¿Es posible reconstruir un activo en el mismo estado que se encontraba previo a la falla que nos forzó a restaurarlo?P/ realizar esto de manera exitosa depende de muchas variables:

- Tener disciplina reflejando c/ uno de los cambios realizados al activo en ciclo de vida
- El administrador se encarga de elegir qué cambios son necesarios reproducir y cuales ser obviados.
- Error humano, podemos equivocarnos al intentar reconstruir el activo. + antiguo el activo → + cantidad de cambios a aplicar → + posibilidad de cometer un error.
- La existencia de **cambios no autorizados** → que es implementado con éxito es más peligroso que uno que no lo es. → produce una configuración efectiva sin dejar evidencia de la misma.

Configuration management

- tradicional → cada servidor es administrado de forma individual → mascota → con un trato personalizado.
- moderno → los servidores se administran en conjunto, con una mirada industrial y abarcativa del parque tecnológico. Enfoque que nos permite implementar soluciones rápidas y reemplazar componentes fallidos en lugar de invertir tiempo en intentar resolver problemas complejos. Ganado se administra como un todo.

¡***Configuration as Code*** al rescate! Moderno

Definir la configuración de un servidor cómo código (CaC) es fundamental.

- Gestionar el parque tecnológico como ganado no sería posible sin la utilización de un sistema de configuration management.
- Cualquier cosa definida como código puede ser automatizada.
- El código puede ser sometido a pruebas.
- CaC es un paso anterior a habilitar Self-Healing y Self-Remediation (2 prácticas del mundo moderno de infraestructura que permiten implementar procesos de autorreparación)
- Es compatible con el proceso de change management de ITIL → las modificaciones no suceden en los activos, sino en un repositorio que soporta versionado. De modo que los cambios pueden ser testeados y desplegados en ambientes bajos para luego aplicarlos en producción.
- No reemplaza al proceso de configuration management de ITIL.

Desafíos de adoptar CaC: Transicionar de los procesos tradicionales a procesos modernos.

- Entender si realmente tiene sentido transicionar a estos procesos.
- Comprender qué tan lejos queremos ir con la implementación de Cac, ¿vamos a adaptar los sistemas existentes?¿o adoptamos la práctica solo para los nuevos?
- Abandonar las viejas formas de trabajar puede presentar resistencia.
- Puede que necesitemos repensar la forma en la que se construyen los servidores en la organización.

Video - Proceso de configuración management moderno → Configuración as código.
Todo comienza con un requerimiento, que puede ser desde configurar un aspecto de un S.O, un antivirus, instalar un Servidor web o servidor de bases de datos.

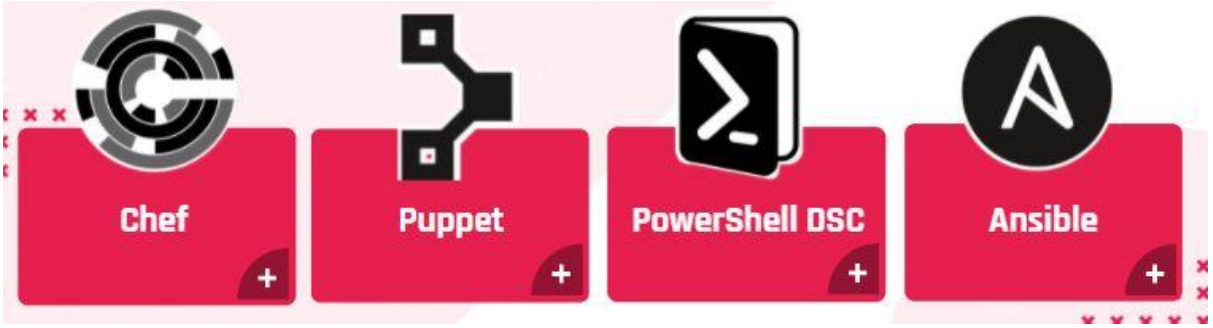
En un escenario moderno, el administrador escribe un archivo o manifiesto de dicha configuración que luego será aplicado a los servidores correspondientes.
Procesos tradicionales → imperativos , donde configuramos un servidor.
Procesos modernos → declarativos, ya que manifestamos la configuración que será aplicado por un sistema de configuration management en el servidor.}
Una vez que la configuración está codificada pueden suceder dos cosas:
1- Aplicar la configuración manualmente al servidor.
2- Guardar los cambios en un sistema de control de versiones. Un repositorio Git.
Pipeline → utilizado en el proceso de despliegue de software. 2 instancias muy distintas:

- Build → testear la configuración y compilar.
- Release → Enviar al sistema de configuration management el resultado del proceso de build.

Los servidores que forman parte de nuestro ambiente están gobernados mediante un sistema de configuration management, tienen instalado un agente con dos funcionalidades:

1) Registra el servidor contra el sistema de configuration management.
2) Opera de representante para el sistema de configuration management en el contexto del servidor a configurar.

Herramientas de configuration management



-Chef → arquitectura cliente-servidor, orientada a **Linux.** Las configuraciones son definidas y asignadas en el servidor y es mediante el cliente de Chef que se hacen efectivas en aquellos activos que deseamos configurar. La presencia de un cliente hace que Chef pueda detectar las desviaciones que se producen a causa de posibles intervenciones manuales y así corregirlas para devolverla al estado deseado. El cliente de Chef funciona en **modo "pull"**, lo que significa que este verifica periódicamente contra el servidor si hay nuevas versiones de la configuración. Las configuraciones de Chef se escriben en **Ruby.**

-Puppet→ arquitectura cliente-servidor, orientada a **Linux**. Las configuraciones son definidas y asignadas en el servidor y es mediante el cliente de Puppet que se hacen efectivas en aquellos activos que deseamos configurar. La presencia de un cliente hace que Puppet pueda detectar las desviaciones que se producen a causa de posibles intervenciones manuales y así corregirlas. El cliente también funciona en modo **"Pull"**. Las configuraciones de Puppet se escriben en un **DSL** (domain specific language) creado por Puppet labs. La gran diferencia entre Puppet y Chef está dada por las capacidades de reporting que tiene Puppet.

-PowerShell DSC → DSC (del inglés, desired state configuration) es una tecnología que forma parte de PowerShell, originalmente desarrollada para mantener la configuración de hosts que **Windows** Server, es multiplataforma y puede también configurar hosts **Linux**.

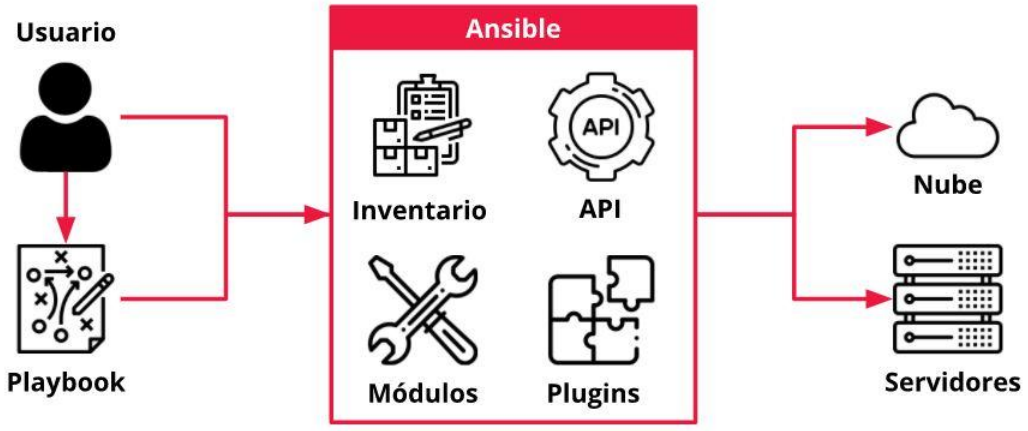
Puede funcionar tanto en una arquitectura cliente-servidor (**modo "pull'**) o en una modalidad standalone (**modo "push"**). Cuando DSC funciona en modo "push" las configuraciones son definidas localmente y el cliente (llamado LCM, del inglés local configuration manager) es el encargado de aplicarlas sin buscar en un servidor. Tanto en modo "push" como en modo "pull", tiene la capacidad de determinar si han habido desviaciones respecto de la configuración definida y corregirlas.
Las configuraciones para DSC son escritas en el lenguaje de scripting **PowerShell**.

-Ansible → desarrollada en **Python**. Ansible **no requiere** de la instalación **de un agente** en aquellos servidores que se configuraran usando Ansible, sino que la herramienta establece una conexión vía SSH y de esa manera despliega las configuraciones. Su versatilidad ha hecho que su uso evolucione más allá de ser utilizada como herramienta de configuration management para configurar hosts. Sino que también se utiliza **como orquestador**, para controlar despliegues en la nube y hasta desplegar aplicaciones como parte de un proceso de liberación de software. Las configuraciones en Ansible se escriben en archivos con formato **YAML** (un superset del formato JSON).

Ansible
-Herramienta open source de configuration management y de aprovisionamiento, similar a Chef, Puppet o Salt.
-Usa SSH para conectarse a los servidores y ejecutar las tareas de configuración. Nos permite controlar y configurar nodos desde un servidor central.
-Lo que lo hace diferente de otras herramientas de configuration management es que Ansible utiliza la infraestructura de SSH. Fue comprado por Red Hat en 2015.

¿Por qué usar Ansible?
-No usa agentes.
-Idempotente → solo se harán configuraciones si son necesarias y que se podrán aplicar de manera repetible sin provocar efectos secundarios.
-Declarativo → nos deja escribir una descripción del estado que deseamos para un servidor y luego Ansible se encarga de aplicarla de forma idempotente.
-Fácil de aprender.

Arquitectura



*Inventario :
→ Es una lista de los nodos que pueden ser accedidos por Ansible. Está soportado por un archivo de configuration, cuya ubicación es /etc/ansible/hosts. Los nodos pueden estar listados por nombre o IP.
→ Cada nodo es asignado a un grupo (“web servers”, “db servers”,etc)
→ Debe estar escrito en algún formato (YAML, INI,etc)

YAML
```
mail.example.com

[webservers]
foo.example.com
bar.example.com

[dbservers]
one.example.com
two.example.com
three.example.com
```

*Playbooks :
→ archivos escritos en formato YAML.
→ Son la descripción del estado deseado de los sistemas que vamos a configurar. Ansible hace todo el trabajo para llevar los servidores al estado que nosotros hayamos especificado sin importar el estado en el que estén cuando la configuración se aplique.
→ hacen que las nuevas instalaciones, actualizaciones y la administración del día a día sea repetible, predecible y confiable.
→ son simples de escribir y mantener. Se escriben en un lenguaje natural → sencillos de evolucionar y editar.
→ contienen Plays (jugadas)
→ Las jugadas contienen tareas (tasks)
→ Las tareas invocan módulos.

YAML
```
---
- hosts: webservers
  remote_user: root

  tasks:
  - name: Asegurarse que la ultima versiond de Apache este instalada
    yum:
      name: httpd
      state: latest

  - name: Asegurarse que Apache este correindo
    service:
      name: httpd
      state: started
      enabled: yes
```

*Módulos :
→ + 1000 incluidos con Ansible para automatizar las diferentes partes de nuestro ambiente. Se pueden pensar como los plugins que hacen el trabajo real de configuración. Cuando se

ejecutan las tareas escritas en un playbook, lo que se está ejecutando es en realidad un módulo.
→ Cada módulo es independiente y se puede escribir en cualquiera de los lenguajes de scripting standard de mercado(python,Perl,Ruby,Bash) Principio de diseño → idempotencia.
→ + Populares → Service, file, copy, iptables, entre otros.
→ Se incluyen en los playbooks para componer configuraciones complejas o abarcativas.
→ También es posible invocar módulos individualmente desde la CLI una única vez.Probarlo o realizar una tarea específica. Ejemplo, vemos dos comandos:

```Bash
ansible 127.0.0.1 -m service -a "name=httpd state=started"
ansible localhost -m ping
```

1) Reproduce una de las tareas que ya vimos en un playbook que nos ayuda a asegurarnos que el servicio de Apache está corriendo.
2) Invoca el módulo ping para hacer un ping localmente contra “localhost”.

La herencia de Python :
→ El lenguaje es templating (Jinja2)
→ Operador ternario: se puede utilizar dentro de los templates de Jinja para alterar algunos comportamientos de nuestros playbooks en función de ciertas condiciones.
→ Errores → muchas veces los que encontremos van a estar en un formato que nos resultará sencillo de leer.

Casos de uso:

- Aprovisionamiento: Utilizar Ansible para instanciar servidores o máquinas virtuales “configurando el sistema de virtualización”
- Configuration management: gestionar y mantener las configuraciones de nuestros servidores.
- App deployment: distribuir aplicaciones.
- Continuous delivery: Utilizarlo como componente de un proceso de CI/CD para automatizar el despliegue de una aplicación luego de su proceso de compilación.
- Seguridad y compliance: La naturaleza idempotente de Ansible hace que podamos utilizarlo para distribuir configuraciones asociadas con la seguridad sin importar la configuración actual.
- Orchestration: Puede ser utilizado también para orquestar operaciones en la nube o “configurar” la nube.

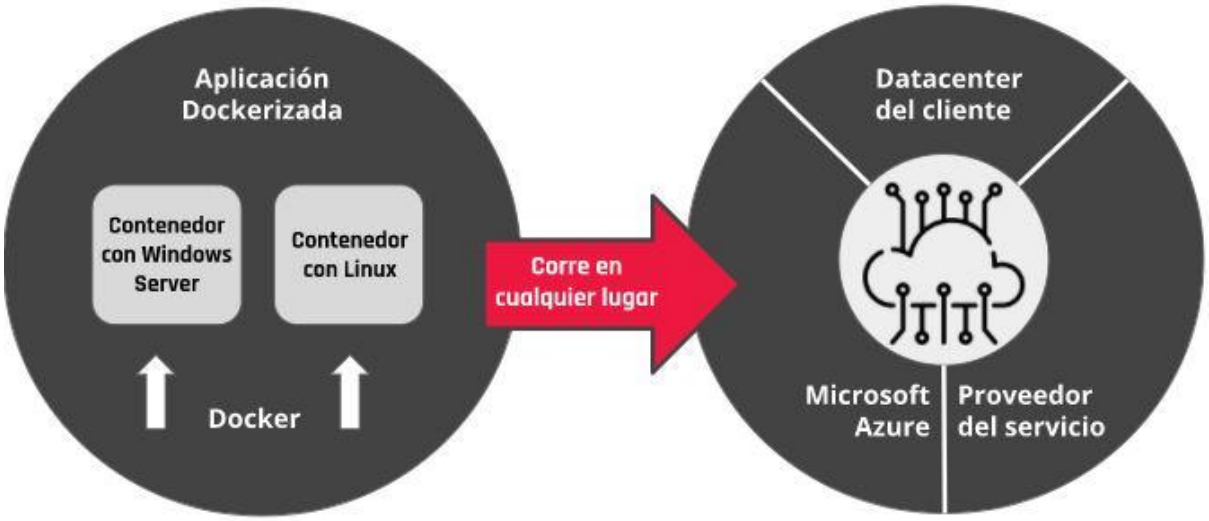
Configuration management con Ansible:
Herramienta más simple para implementar una estrategia de Config Manag. Seguro, consistente y confiable.

CLASE 14 SEMANA 5

DOCKER EN PROFUNDIDAD

- Pueden crear cualquier aplicación en cualquier idioma usando cualquier stack(S.O.)
- Las apps dockerizadas pueden ejecutarse en cualquier lugar y sobre cualquier cosa.
- No más al dicho “Funciona en mi máquina”

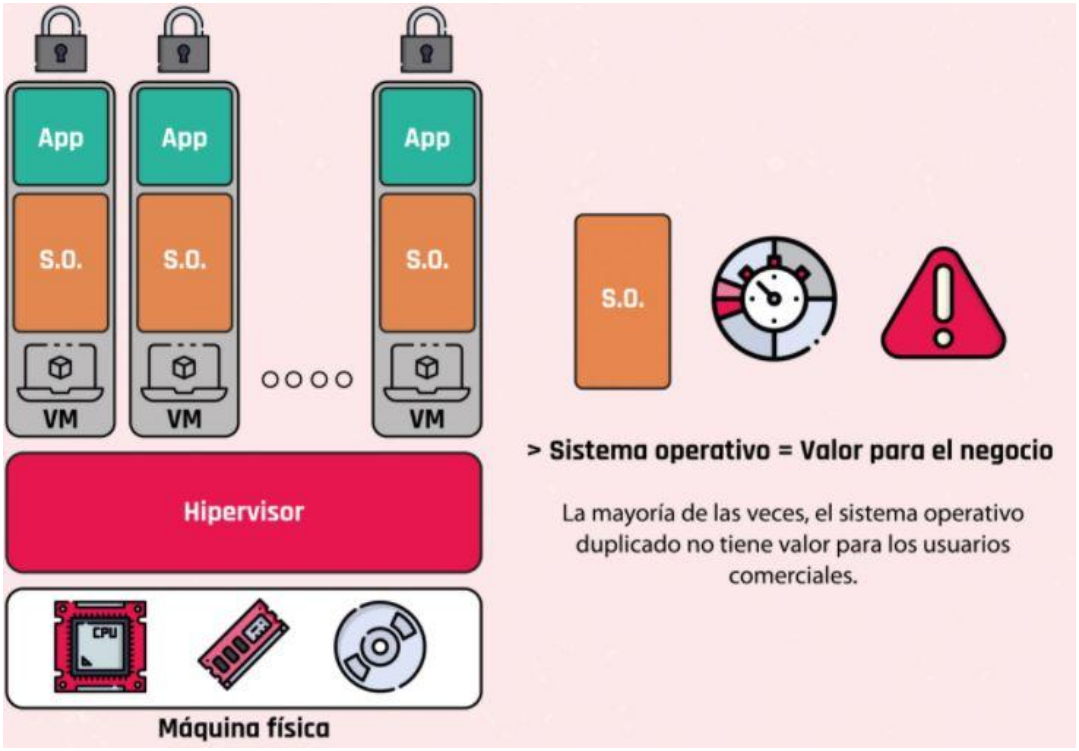
- Encapsulando dependencias los desarrolladores y los administradores de sistemas trabajan mejor en conjunto.



Vocabularios de Docker:

- **Host:** Una máquina virtual que ejecuta Docker daemon para alojar una colección de contenedores Docker.
- **Cliente:** aquí se ejecutan los comandos q están siendo ejecutados(cliente-servidor)
- **Imagen:** Una colección ordenada de sistemas de archivos (capas) que se utilizarán al crear una instancia de un contenedor.
- **Contenedor:** Una instancia en tiempo de ejecución de una imagen.
- **Registro:** Una colección de imágenes de Docker.

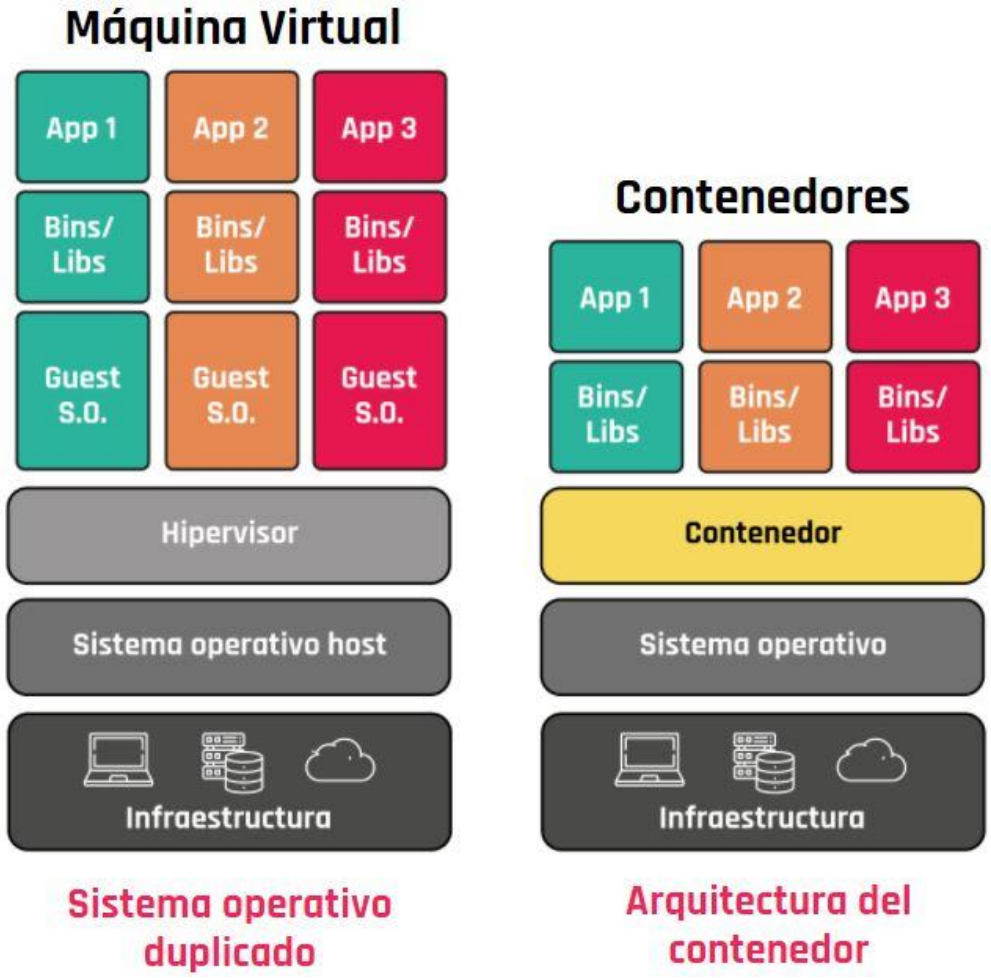
Desafíos con la virtualización:



La creación de una MV con el propósito específico de ejecutar una aplicación puede superar algunos de los inconvenientes de ejecutar aplicaciones directamente en el S.O. host. Aunque una MV está en el host, se ejecuta como un S.O. separado, que incluye su propio kernel, sistema de archivos, interfaces de red, entre otros. Esto hace que sea fácil mantener casi todo dentro del S.O. separado del host.

Comparativa entre máquinas virtuales y containers:

-Las máquinas virtuales deberán contener el sistema operativo y todos los componentes de soporte.
-Los contenedores ofrecen una alternativa a la ejecución de aplicaciones directamente en el host o en una máquina virtual que puede hacer que las aplicaciones sean más rápidas, portátiles y escalables. La flexibilidad proviene de que el contenedor puede llevar todos los archivos que necesita. Al igual que la aplicación que se ejecuta en una máquina virtual, puede tener sus propios archivos de configuración y bibliotecas dependientes, además de tener sus propias interfaces de red distintas de las configuradas en el host. Entonces, nuevamente, al igual que con la VM, una aplicación en contenedores debería poder moverse más fácilmente que sus contrapartes instaladas directamente y no tener que competir por los mismos números de puerto porque cada contenedor en el que se ejecutan tiene interfaces de red separadas.

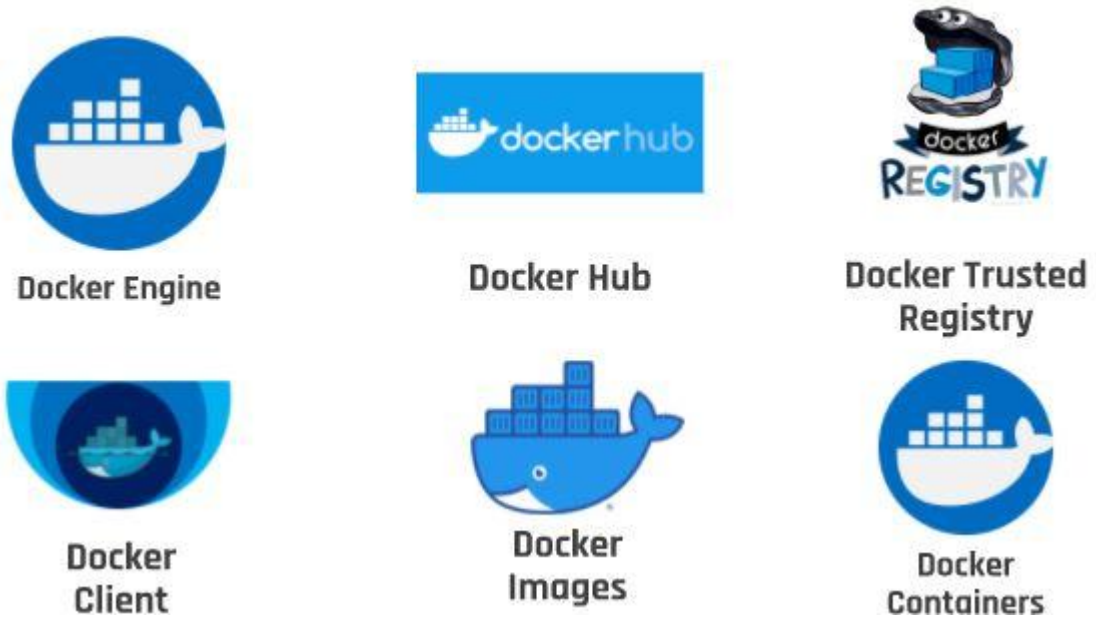


Sistema operativo duplicado

Arquitectura del contenedor

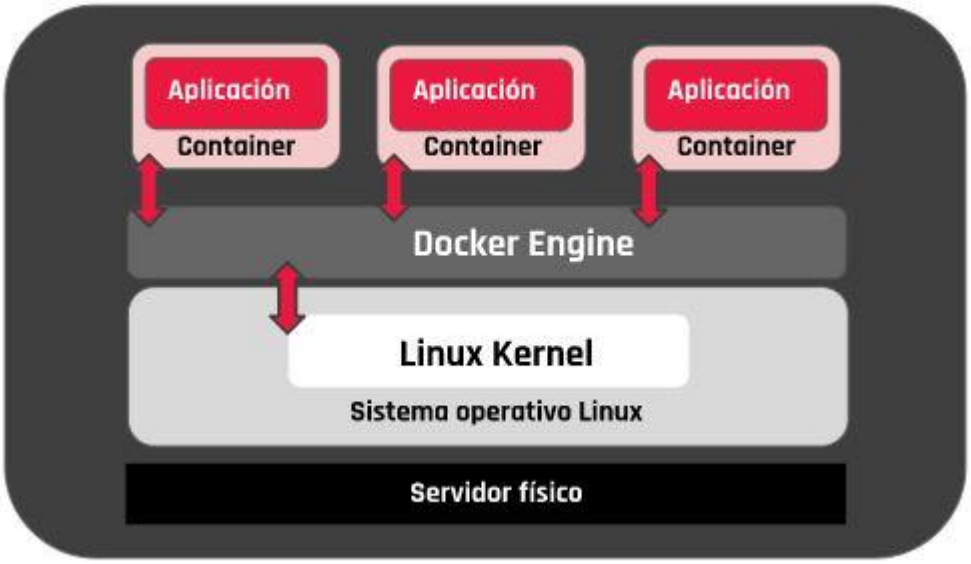
Containers:
-Entornos físicos
→ Aplicaciones construidas e implementadas tradicionalmente en sistemas físicos con relación 1:1.
→ Las nuevas aplicaciones a menudo requieren nuevos sistemas físicos para el aislamiento de recursos.
-Entornos virtuales:
→ Mejor utilización e implementación de aplicaciones, más rápidas que en un entorno físico tradicional.
→ Las aplicaciones implementadas en máquinas virtuales son muy compatibles.
-Entornos físicos/virtuales:
→ Aceleran aún más la implementación de la aplicación.
→ Reducen el esfuerzo para implementar aplicaciones.
→ Optimizan el desarrollo y las pruebas.
→ Menores costos asociados con la implementación de aplicaciones.
→ Incrementan la consolidación de servidores.

Plataforma de Docker:

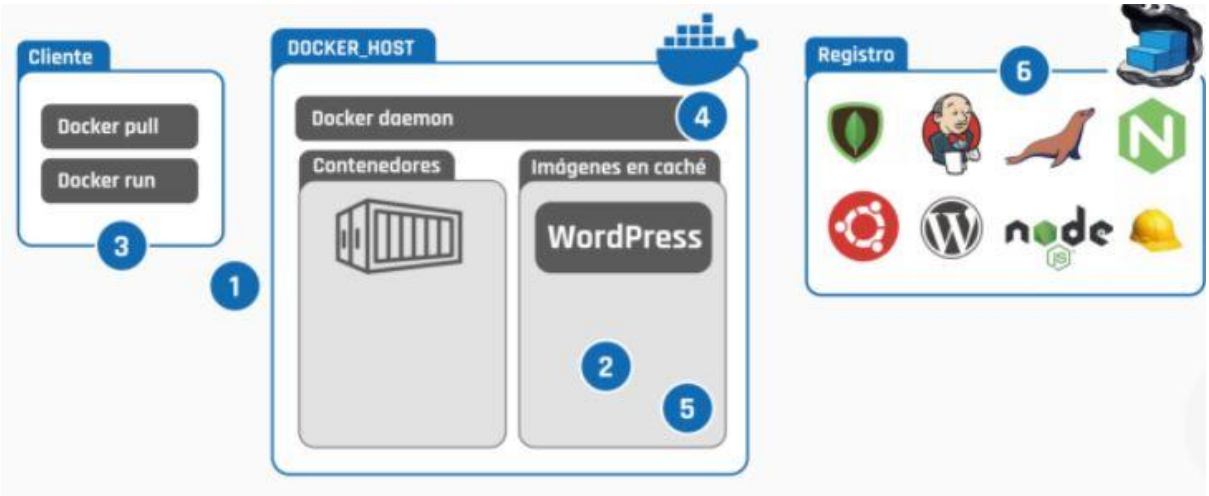


- Docker Engine (Docker daemon) → Es el programa que permite construir, enviar y ejecutar contenedores. Utiliza espacios de nombres y grupos de control del kernel de Linux para proporcionar un entorno de tiempo de ejecución aislado para cada app.
- Docker Hub → Es un registro en línea de imágenes de Docker.
- Docker Trusted Registry → es un registro privado en el sitio p/ imágenes de Docker.
- Docker Client → Es el que toma las entradas del usuario y las envía al daemon. El cliente y el daemon pueden ejecutarse en el mismo host o en diferentes hosts.
- Docker Images → es una plantilla de solo lectura utilizada para crear contenedores. Contiene un conjunto de instrucciones para crear los contenedores.
- Docker Containers → Es una plataforma de aplicación aislada basada en una o más imágenes que contiene todo lo necesario para ejecutar una aplicación.

Docker Trusted Registry>



Docker en acción:

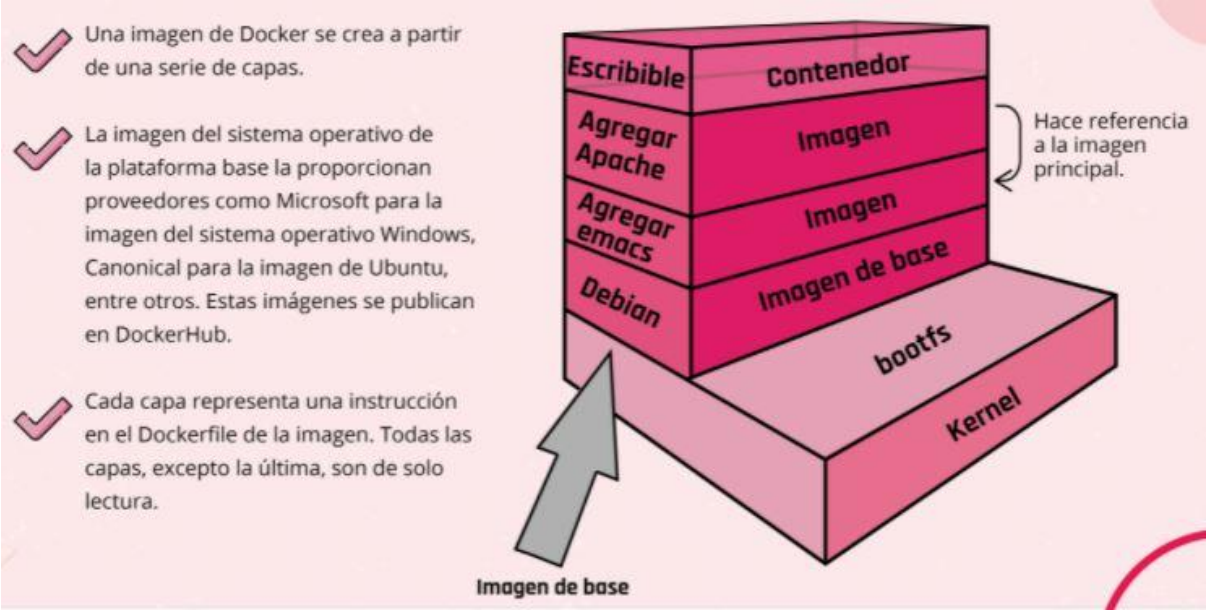


1) Localmente teníamos Docker CLI (cliente) + Docker host (el blog se ejecutó localmente en nuestra máquina)
2) De forma remota, accedimos a un repo de imagen con la imagen de Wordpress.
3) Docker CLI → para interactuar con contenedores en su máquina.
4) Docker daemon para ejecutarlos en su máquina.
5) Imagen en caché para una extracción rápida.
6) Imágenes públicas disponibles en el registro de Docker.

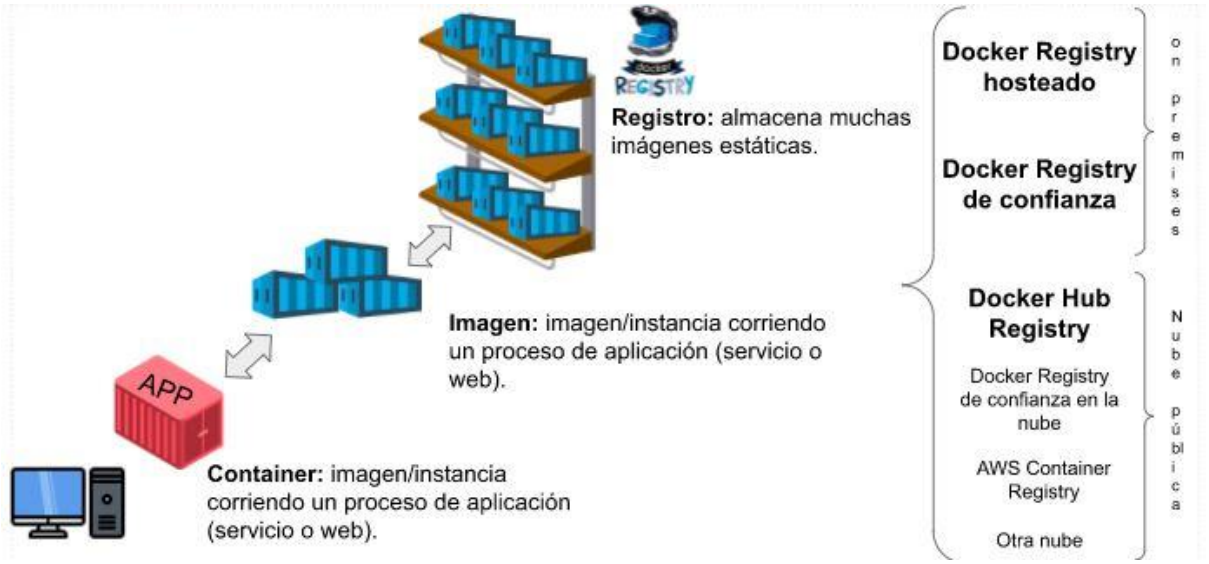
¿Qué es un registro Docker?
→ Sirven para almacenar las diversas imágenes Docker que utilicemos en nuestro sistema. De esta forma podremos subir imágenes nuevas a los registros o descargarlas cuando las necesitemos en alguna máquina Docker.
→ Es como una estantería donde las imágenes se almacenan y están disponibles para extraerlas con el fin de compilar contenedores que ejecuten servicios o apps web.
→ Hay privados a nivel local y en la nube pública. Docker Hub es un registro público mantenido por Docker.

→ Colocar imágenes en un registro permite almacenar fragmentos de la app que son estáticos e inmutables, incluidas todas sus dependencias a nivel de marco. Después esas imágenes se pueden versionar e implementar en varios entornos y, por lo tanto, proporcionar una unidad de implementación coherente.

Cómo se compone una imagen.



Taxonomía básica de Docker



Dockerfile:
Es un archivo de texto simple con un conjunto de comandos o instrucciones. Estos comandos/instrucciones se ejecutan sucesivamente para realizar acciones en la imagen base para crear una nueva imagen de la ventana acopable.

Instrucciones básicas de Dockerfile:
-DE → Define la imagen base para usar e iniciar el proceso de construcción.
-CORRER → Toma el comando y sus argumentos para ejecutarlo desde la imagen.
-CMD → Función similar a un comando run, pero se ejecuta solo después de que se crea una instancia del contenedor.
-PUNTO DE ENTRADA → Se dirige a su aplicación predeterminada en la imagen cuando se crea el contenedor.
-AÑADIR → Copia los archivos de origen a destino (dentro del contenedor)
-ENV → Establece variables de entorno.

Etiquetas de imagen:
Sirven p/ identificar las versiones de las imágenes, a la hora de listar las imágenes se listan con su tag o etiqueta asociado. Pueden agrupar sus imágenes usando nombres y etiquetas (si no proporcionan ninguna etiqueta, se asume el valor predeterminado de la última)

CLASE 16 SEMANA 6

EL ECOSISTEMA DE DOCKER Y MEJORES PRÁCTICAS

Docker Hub:
Servicio de registro de repositorios proporcionado por Docker Inc. Compartir y colaborar son sus premisas.

¿Para qué sirve Docker Hub?
→ extraer y enviar imágenes de la ventana acoplable hacia y desde Docker Hub. Como un Github.
→ repo en línea basado en la nube que almacena ambos tipos de repo (público y privado).
→ repo públicos son accesibles para todos, pero el privado es accesible para el propietario.
→ costo asociado si almacenamos más de un cierto número de repositorios como privado.

Características de Docker Hub:
1) Repositorios de imágenes : Ayuda a encontrar y extraer imágenes de contenedores.
2) Equipo y organizaciones: Crear grupos de trabajo e impulsar los repo privados, que están disponibles para su uso únicamente dentro de nuestra organización.
3) Integración de GitHub y Bitbucket: Permite la integración con repo de código fuente.

4) Construcciones automatizadas: Si se ha enviado un cambio en el código fuente a los repositorios de código fuente, automáticamente detecta y crea imágenes de contenedor desde GitHub o BitBucket y las envía a Docker Hub.
5) Webhooks: Una vez que hemos enviado nuestras imágenes con éxito, con la ayuda de un webhook, desencadena una acción p/ integrar Docker Hub con otros servicios.
6) Imágenes oficiales y del editor: Las imágenes de alta calidad proporcionadas por los dockers se consideran imágenes oficiales y se pueden extraer y utilizar. Del mismo modo, las imágenes de alta calidad proporcionadas por proveedores externos son imágenes del editor, también llamadas imágenes certificadas, que brindan soporte y garantía de compatibilidad con Docker Enterprise.

Creación del primer repositorio:
Solo nos ofrece un repositorio privado de forma gratuita. Con la herramienta terminal de Docker Desktop, descargada e instalada, podremos iniciar sesión en Docker Hub mediante un comando. → docker login

Explorando las imágenes:
Hay dos formas de buscar imágenes y repositorios públicos desde Docker Hub. En el sitio web o usar la herramienta de línea de comandos y ejecutar el siguiente comando:
docker search mysql

Descargar una imágen:
-Usando el comando pull:
docker pull mysql
-Si ya tenemos mysql image en nuestra máquina, el comando anterior actualizará automáticamente la imagen a la última versión.

Crear una imagen:
-Requiere un Dockerfile → manual de instrucciones que le dice a Docker qué ensamblar.
-Docker lee las instrucciones de un Dockerfile y crea imágenes automáticamente. La imagen de Docker es un sistema de archivos en capas y consta de varias capas de solo lectura y cada capa de una imagen de Docker representa las instrucciones de un Dockerfile.
sudo vim Dockerfile

```
FROM ubuntu:16.04
MAINTAINER someuser@somedomain.com
RUN apt-get update
RUN apt-get install -y mysql
CMD echo "My first image created."
```

Podemos utilizar # símbolos para agregar un comentario en un Dockerfile.
-FROM → define la imagen base que se utilizará.
-MAINTAINER → persona que va a mantener esa imagen.
-RUN → Se utiliza p/ejecutar la instrucción dada p/la imagen(1° actualice,2° instale MySQL)
-CMD → Se utiliza para ejecutar un comando una vez que se ha lanzado el contenedor.

-COPY → Se utiliza para copiar un archivo de nuestro S.O. host al contenedor de la ventana acoplable.
-EXPOSE→ Se utiliza para especificar el número de puerto en el que el contenedor ejecutará su proceso.

Empujar una imagen:
Una vez creada nuestra imagen correctamente y se está ejecutando, podemos enviarla a Docker Hub mediante el comando push.
docker push asadali08527/first-repo

¿Qué son las imágenes certificadas por Docker?
Imágenes oficiales impulsadas por proveedores o contribuyentes. Solo puede ser certificado por Docker Hub si su contenido cumple con las reglas, estándares y leyes proporcionadas por Docker Hub. Debe pasar ciertas pruebas de referencia.
Docker Hub proporciona InspectDockerImage → herramienta a través de la cual un proveedor puede autocertificar las imágenes y los complementos.

Imágenes populares en Docker Hub:
Antes de extraer una imagen hay que considerar algunos factores clave:

- Buscar una versión específica utilizando etiquetas.
- Optar por el que tenga máxima cantidad de descargas y estrellas.
- Buscar la fecha de su última actualización
- Si es posible, verificar su tipo, ya sea del editor verificado u oficial(Docker Certified)

Docker Networking:

Nos enfocamos en el driver llamado bridge. Todos los otros drivers están orientados a ambientes productivos, pero ese ámbito hoy está dominado por Kubernetes, que tienen otro modelo de Networking completamente diferente.

- Las redes de tipo “bridge” son locales y exclusivas del host en donde fueron creadas.
- tipo de red por defecto en Docker.
- Simula la creación de switches o hubs, de nivel 2(en el modelo OSI)
- Por defecto se crea una red de tipo bridge llamada “bridge” (mejor que no sea usada) → crear nuevas redes para usos específicos.

Resolución de nombres:
Las IPs son efímeras y + en los containers → necesitamos una forma para la comunicación entre los containers sin conocer la IP que les fue asignada.
P/todas las redes que creemos (no las por defecto) + los containers a los que se le asigne un nombre de manera deliberada (--name en docker create) → Docker nos provee con un servicio de resolución de nombres dentro del ámbito de la red misma utilizando el nombre del container para identificarlo por medio del servicio DNS.

<u>docker network</u>

- create: nos permite crear una red
  --internal: la red creada será de tipo ‘privada’.
- connect: nos permite conectar un container existente a una red.

docker run
--network <networkName> : nombre de la red a la cual conectaremos al container.
--name : nombre del container, necesario para que funcione la resolución de nombres.
-p <hostPort>:<containerPort> : nos permite publicar ciertos puertos de un container en el host para poder acceder al servicio dentro del container
-p: nos permite publicar todos los puertos definidos en la especificación del container (Dockerfile) en puertos aleatorios del host.

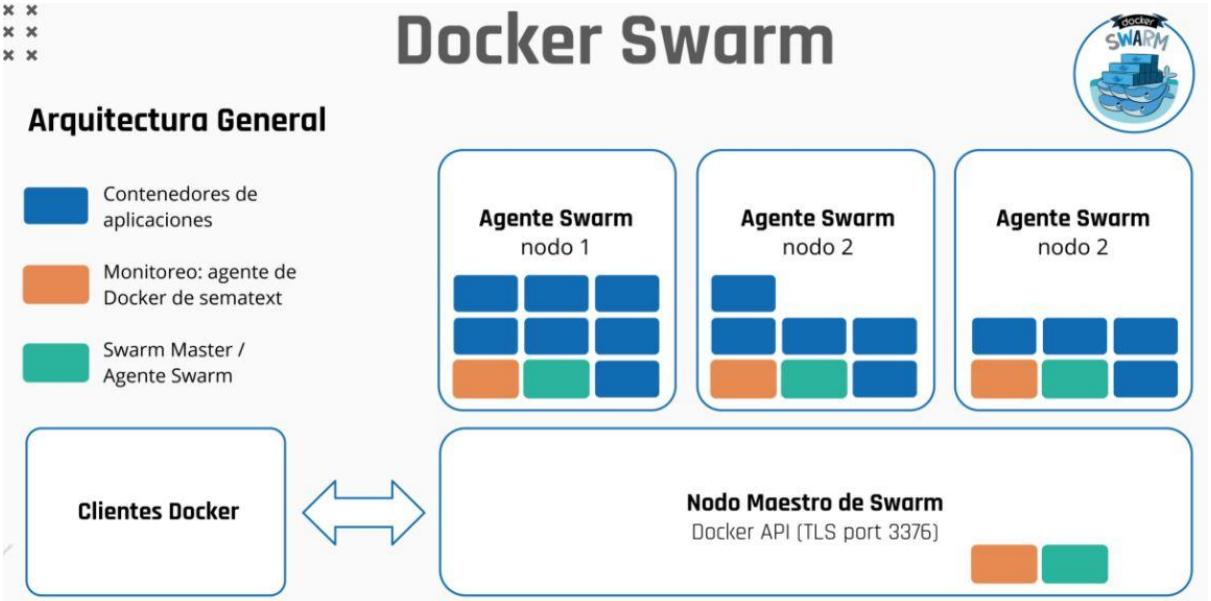
Docker Compose

→ Herramienta que permite simplificar el uso de Docker. A partir de archivos YAML es + sencillo crear contenedores, conectarlos, habilitar puertos, volúmenes, etc.
→ Podemos crear diferentes contenedores y al mismo tiempo, en c/contenedor, diferentes servicios, unirlos a un volumen común, iniciarlos y apagarlos. Es un componente fundamental para poder construir aplicaciones y microservicios.
→ Facilidad para dar una serie de instrucciones y luego repetirlas en diferentes ambientes.
→ La idea es conseguir optimizar al máximo nuestro entorno de desarrollo y enfocarnos en el principal objetivo que es escribir código de calidad.

¿Por qué utilizar Docker con Docker Compose?

+ No necesitamos instalar ni mantener software adicional en nuestro equipo.
+ Podemos tener todo nuestro entorno de desarrollo en un único repositorio (back end, el front end y las configuraciones de la base de datos)→ facilita a los desarrolladores el poder colaborar de mejor manera en el proyecto.
+ Levantar todo el entorno de desarrollo se limita a un solo comando docker-compose up.
- Está enfocada al rendimiento (los contenedores están enfocados a ser eficientes) pero siguen consumiendo recursos de la máquina anfitriona (procesador+memoria) si la cantidad de contenedores que están corriendo al mismo tiempo es grande o si los contenedores son pesados al crearse y levantarse, podrían llevar a cuelgues del sistema y cosas similares.

Docker Swarm:



Swarm es una herramienta integrada en Docker (aplicación para la creación de contenedores) que permite agrupar una serie de hosts de Docker en un clúster y gestionarlos de forma centralizada, así cómo orquestar contenedores.

Mejores prácticas en Docker para desarrolladores:

→ Utilizar imágenes oficiales: En Docker Hub hay 2 tipos de imágenes. Las oficiales, publicadas por organizaciones (por ejemplo, Ubuntu) o las publicadas por usuarios.
¿Cómo diferenciarlas?
Las oficiales están etiquetadas como Docker Official Image. Las imágenes no oficiales incluyen el nombre del usuario que las publicó, mientras que las oficiales no.
Veamos la diferencia a la hora de ejecutar 'docker pull':
-Imagen oficial → docker pull ubuntu:latest
-Imagen no oficial → docker pull usuario/nginx:latest

→ Utilizar el comando COPY en lugar de ADD:
Ambos agregan o copian un archivo de una fuente externa a una imagen de Docker.
-ADD fue el primero en ser creado para los Dockerfiles. La diferencia reside en que ADD soporta copiar desde distintos orígenes y desempaquetar un archivo .tar (tarball, un formato de empaquetado nativo de Unix / Linux). ADD puede generar confusión, incluso algunos inconvenientes técnicos y hasta generar problemas de performance.
-COPY fue introducido para cumplir la función específica de copiar un archivo desde un filesystem local a la imagen. Facilita la lectura del dockerfile.
En términos generales se recomienda:
● Usar COPY para copiar archivos locales a la imagen en tiempo de compilación.
● Usar RUN para ejecutar curl y encadenar con otros comandos para bajar un archivo de una fuente web y copiarlo, instalarlo o descomprimirlo.
● Usar ADD cuando se tiene un archivo .tar y se lo quiere desempaquetar dentro de la imagen que se está construyendo.

→ Usar multi-stage builds:
La misma nos permite generar imágenes más pequeñas incluyendo explícitamente sólo aquellos componentes necesarios para ejecutar una aplicación.
En los multi-stage builds los Dockerfiles tienen dos o más 'FROM' → produce varias imágenes etéreas durante el proceso de 'build' y nos permite copiar elementos entre ellas hasta generar una imagen definitiva con solo aquello estrictamente necesario para ejecutar la aplicación.
Se usa una 1era imagen de base especificada en el 1er FROM→ en la que copiamos el código fuente de nuestra aplicación y con las herramientas necesarias para compilarla.
Dentro del mismo Dockerfile indicamos un 2do FROM→ solo lo necesario para ejecutar la aplicación. Y copiamos el artefacto generado x la compilación en el 1er container, al 2do.

→ No generar dependencias externas:
Los contenedores que construyamos tienen que ser trasladables. No importa en donde corran, deben contar con todo lo necesario para ejecutar las aplicaciones que viven en ellos.
En los ambientes de desarrollo no hay que utilizar volúmenes de tipo bind.

→ Concatenar comandos:

Muchas veces nos vamos a encontrar en Dockerfiles con líneas como esta:

- RUN apt-get update && apt-get upgrade -y

En lugar de algo como:

- RUN apt-get update
- RUN apt-get upgrade -y

¿Por qué? Cada línea en un Dockerfile produce una nueva capa en nuestra imagen, de modo que ejecutar estos 2 comandos por separado, produciría 2 capas.
En este caso, ambos comandos sirven al mismo propósito, actualizar los paquetes instalados en el S.O. Tiene sentido concatenarlos ya que están estrechamente relacionados y ejecutar uno sin ejecutar el otro no tendría sentido. De esta manera optimizamos el uso de recursos y tiempo de compilación.

CLASE 17 SEMANA 6

INTRODUCCIÓN AL CLOUD COMPUTING:

-La computación en la nube es la tecnología que permite tener todos nuestros archivos e información en la nube de manera que podamos acceder a ellos desde múltiples dispositivos y sin necesidad de preocuparnos por poseer la capacidad suficiente de almacenamiento.
-Se ofrece como un servicio web por demanda → significa que la aplicación software corre en una infraestructura o servidor proveedor para dar servicio a múltiples clientes.

A finales de la década ‘90 aparecen las 1eras empresas (Salesforce.com, Amazon Web Services) que ofrecen servicios en la nube. Años ‘70 surge la idea de una red de computadoras universal.
El impulso definitivo al cloud computing se produjo gracias a la inmersión en esta tecnología de 2 gigantes de la informática: Google y Microsoft.

Definición
La informática en la nube es la entrega bajo demanda de potencia de cómputo, bases de datos, almacenamiento, aplicaciones y otros recursos de IT, a través de Internet con un sistema de precios de pago por uso. Estos recursos se ejecutan en equipos de servidores ubicados en grandes centros de datos en diferentes partes del mundo. Se deja de considerar la infraestructura como hardware y en cambio, se empieza a verla (y usarla) como software.

¿Cómo funciona?
Se sustenta en 3 pilares fundamentales, esenciales y diferentes que cubren distintas áreas de producción y servicios, ya sea para una persona o empresa.

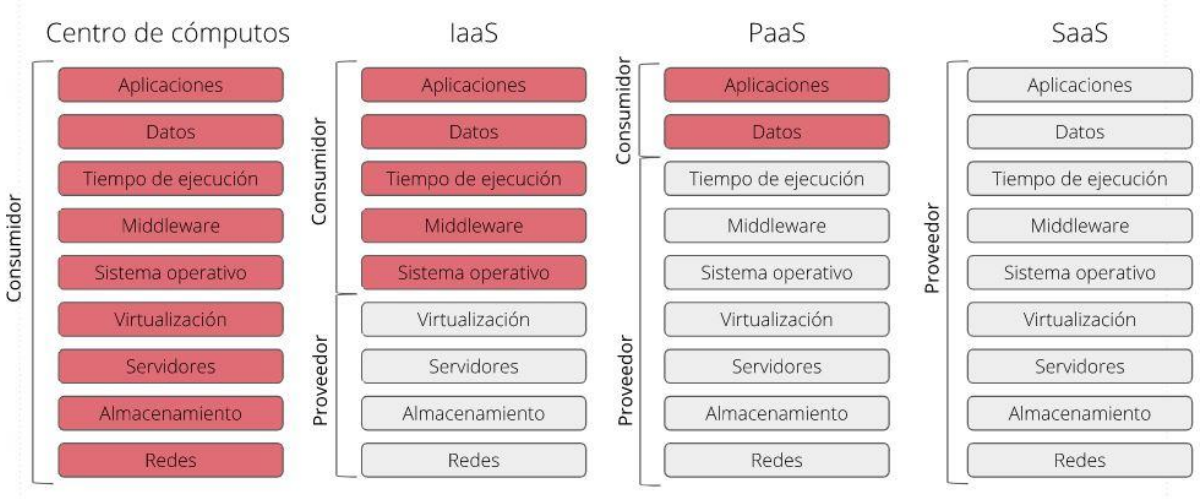
- Software como servicio
- Plataforma como servicio
- Infraestructura como servicio

Tipos de nubes → En función de su privacidad existen:

- Nubes privadas → Accesibles únicamente desde una determinada organización. Gestionadas por la propia organización o por un tercero.
- Nubes públicas → Abiertas al público y son propiedad de un proveedor de cloud computing que se encarga de gestionarlas. Todas las garantías de privacidad, seguridad y disponibilidad, así como las penalizaciones por incumplimiento, deben estar expresadas en el contrato de servicio.
- Nubes híbridas → Mezcla de los dos anteriores y tienen la capacidad de portabilidad de aplicaciones y datos como características principal.

Modelo de responsabilidades:
Cloud computing → utilizar recursos de un tercero → inquilinos de aquel que es propietario de los activos informáticos.
MdR → es un estándar que nos permite definir diferentes tipos de contratación de los servicios al dueño de la nube, qué tareas quedan delegadas a la nube y cuáles serán nuestra responsabilidad.

<u>El modelo:</u>



- Columnas → centro de cómputos, IaaS. PaaS y Saas.
- Roles → consumidor y proveedor.
- Cajas que definen un conjunto de tareas, tecnologías y/o disciplinas con las que ya estamos familiarizados, como pueden ser: redes, virtualización o middleware.
- Colores que tiñen cada una de ellas.

<u>Los roles: consumidor y proveedor</u>
Para definir quién es responsable.
cloud computing → contratar recursos de manera flexible y con interfaces bien definidas -APIs- para la administración de los mismos a un tercero → dueño de la nube o de los centros de cómputos en los que decidamos correr nuestras aplicaciones → proveedor.
Quienes deciden contratar los servicios p/ ejecutar las apps en sus centros → consumidor.

<u>Las columnas:</u>

- <u>Centro de cómputos</u> → definir de qué tareas nosotros seríamos responsables si somos quienes alojamos en su totalidad los servicios de tecnologías → no

contratamos los servicios de ningún proveedor p/ que administre nuestros sistemas.
*Pros: Total control sobre los activos, poder decidir cómo, para qué, dónde y cuándo los utilizamos.
*Contras: Responsabilidad total por la salud, utilización y mantenimiento del parque informático. No puedes escalar de manera flexible.

- IaaS → Infrastructure as a Service → proveedor nos da posibilidad de instanciar MV en su centro de cómputos sin nosotros tener la necesidad de administrar la infraestructura subyacente.
  *Pros: Control sobre el S.O., ambiente ideal para instalar apps legacy. Permite escalar la infraestructura de forma más dinámica.
  *Cons: Responsables por la administración y salud del S.O. y todos los servicios o app que se ejecuten encima de este
- PaaS → Platform as a Service → el proveedor ofrece una tecnología específica expuesta por medio de interfaces definidas (APIs) y nos abastece de los recursos subyacentes. Ejemplo: Amazon DynamoDB.
  *Pros: Abstraemos 100% de la administración de los recursos de las capas inferiores. Delegamos la gestión, monitoreo y escalabilidad de la tecnología.
  *Contras: Opción costosa. Las configuraciones disponibles están limitadas a aquellas que el proveedor haya decidido exponer o disponibilizar.
- SaaS → Software as a Service → el proveedor nos ofrece una aplicación. Ejemplos: Trello, Salesforce o GMail.
  *Pros: Con una tarjeta de crédito accedemos a una app lista p/utilizar y resolver una necesidad. Sin tener que cubrir el costo de la infraestructura y operación de la misma.
  *Contras: Poco a nulo control sobre la configuración del sistema contratado.

Algunos proveedores de cloud computing

Microsoft Azure → Está en el primer puesto gracias a cuatro factores principales:
+ su oferta de servicios en las tres capas de la nube (IaaS, PaaS y SaaS);
+ su compromiso para desarrollar y ayudar a los clientes a implementar inteligencia artificial,
+ machine learning y Blockchain en entornos de producción innovadores y
+ sus ingresos en la nube líderes en el mercado.

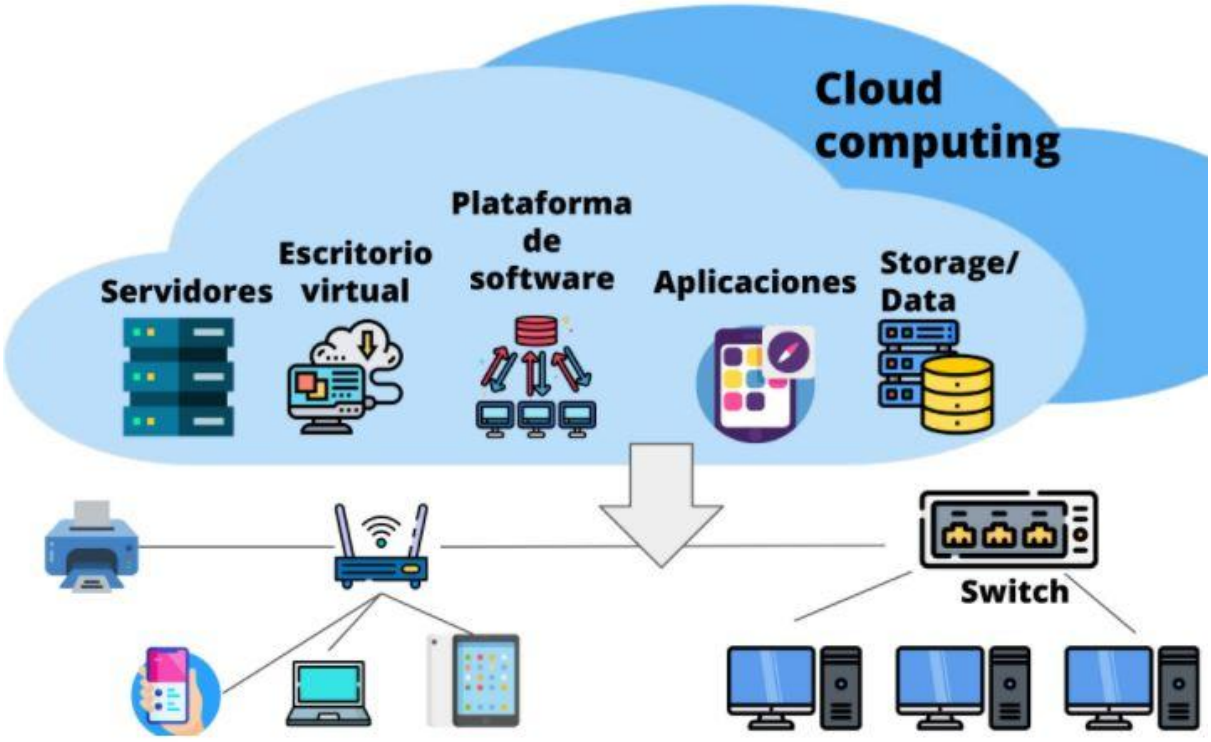
Amazon → Ofrece una experiencia de usuario sencilla y brinda una gran variedad de productos y servicios. La propuesta de valor es similar a la de Azure ofreciendo servicios en los tres modelos de responsabilidad: IaaS, PaaS y SaaS.

Google Cloud Platform (GCP) → Ofrece un software libre y entornos multinube que permite usar los datos y ejecutar las aplicaciones en cualquier tipo de entorno o nube, ya sea públicas o privadas. A su vez, para proteger los datos, proporciona encriptado de datos.

IBM → Aunque no esté dentro de las tres mejores, IBM se encuentra ganando posiciones gracias a la transformación en su tecnología de software desde el entorno real hasta la nube.

CLASE 19 SEMANA 7

COMPUTACIÓN EN LA NUBE



¿Por qué usar una máquina virtual en la nube?

Por ejemplo, podríamos necesitar un servidor conectado las 24 horas a internet, donde se alojan páginas web y servicios online de venta o distribución de algún producto.

En este caso solo tienen dos opciones.

-La primera sería comprar un costoso equipo para montar un servidor en nuestro negocio, además de tener que comprar dispositivos para conectarlo a internet y un plan de datos de alta velocidad, configurarlo, instalar sistemas de seguridad y contra incendios para protegerlo, comprar e instalar sistemas de respaldo de datos, instalar sistemas de energía por si hay un apagón o, en fin, atender cualquier problema físico o de configuración característicos de la administración de un servidor.

-La segunda opción sería contratar un servicio para crear una máquina virtual en la nube, que podría actuar como servidor y a la cual solo tendrían que subir sus bases de datos y programas usados para operar sus negocios. Microsoft Azure, Amazon Web Services y Google Platform son de los servicios existentes p/ crear sistemas virtualizados en la nube. Además, esta máquina sería accesible en poco tiempo a partir de la contratación; y su mantenimiento quedaría a cargo de la prestataria del servicio.

Casos de uso

¿Cómo puedo crear una máquina virtual en la nube?

Actualmente hay dos grandes sistemas que ofrecen servicios de máquinas virtuales en la nube, operados por las gigantes Amazon y Microsoft.

-El sistema de Amazon es parte de su plataforma Amazon Web Services (AWS) y se denomina Amazon Elastic Compute Cloud (EC2). Este servicio crea y ejecuta máquinas virtuales en la nube a las cuales denomina “instancias”.
Además, ofrece plantillas para que el cliente configure y cree en línea las máquinas que desee, cada una basada en un S.O. como Linux SuSE, Linux Amazon, Linux Ubuntu y Red Hat Linux, así como Windows Server 2012. Estas plantillas también incluyen aplicaciones y configuraciones predefinidas de memoria RAM, número de procesadores virtuales y otros detalles “físicos” de cada máquina a crear.
- Microsoft ofrece su plataforma Azure —que es parte de los servicios de su tecnología IaaS (Infraestructure as a Service)—. Estos servicios incluyen desde la instalación de un servidor web hasta bases de datos y la creación de máquinas virtuales de cualquier tipo y potencia.

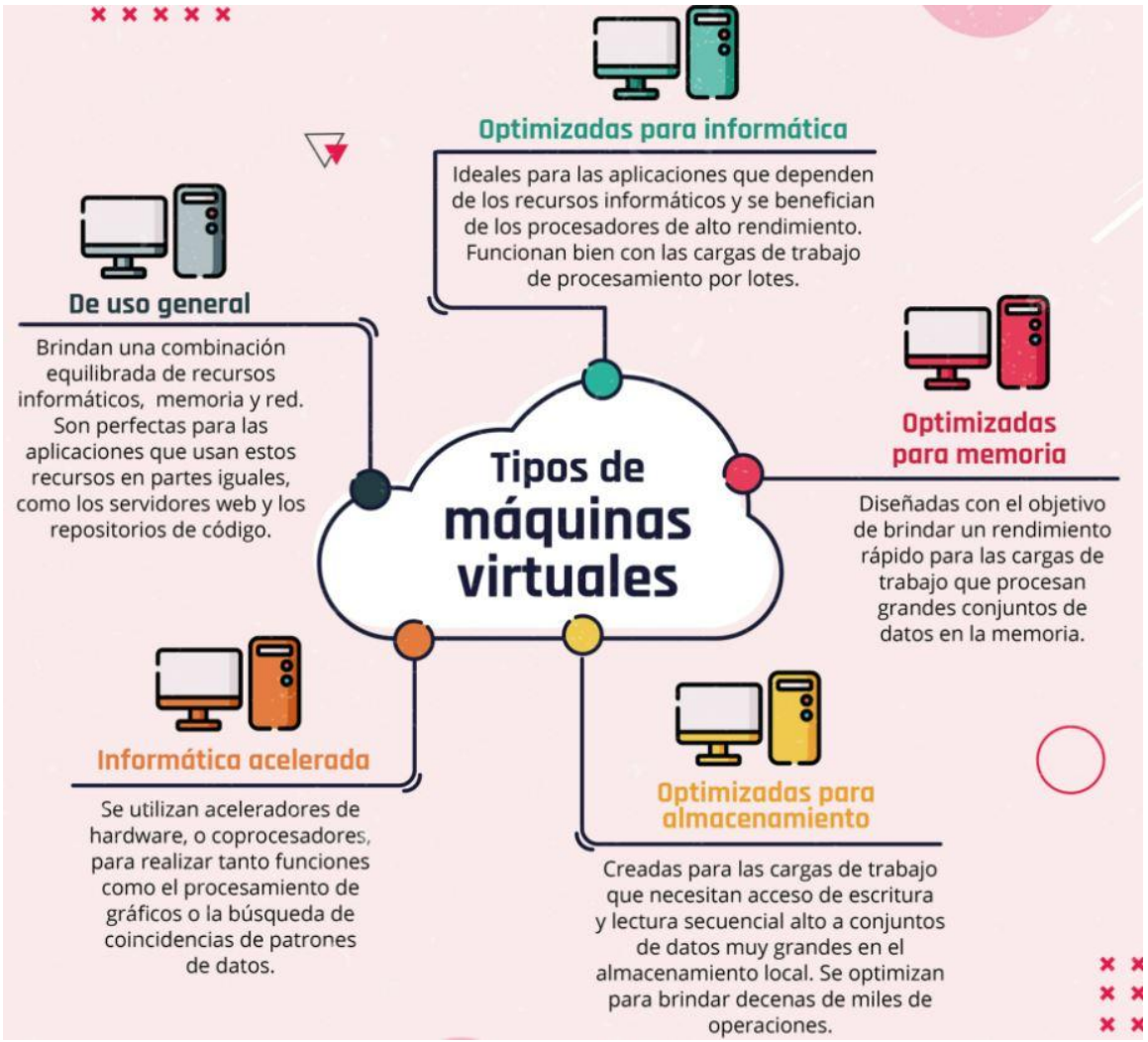
Ventajas de las MV en la nube

La principal ventaja es la económica, ya que solo tenemos que pagar el servicio a un proveedor y generalmente este ofrecerá atractivos paquetes diseñados en función de cuántas máquinas queremos tener, cantidad de procesadores, memoria RAM, espacio en disco duro, entre otras características.
+ Tendremos menos gasto en hardware: Con una MVI en la nube es el proveedor del servicio quien asume el costo del mantenimiento de los servidores y demás equipos reales requeridos, así como también del tiempo y costo de mantener el “hardware virtual” que se configuró según los requerimientos.
+ Software actualizado constantemente: El proveedor tendrá siempre actualizado el software usado para configurar y mantener las MV de los clientes.
+ Solo pagamos por el uso: Al contratar una MV en la nube solo pagaremos por el uso que le dan a la misma y cuando no la necesiten, pueden desconectarse para reducir costos.
+ Accesibilidad desde cualquier lugar: No importa dónde se encuentren, siempre podemos acceder y usar sus MV en la nube desde un equipo que tenga conexión a internet.
+ Mayor rapidez de implementación: El costo y tiempo necesario desde que contratan una MV hasta que pueden usarla es mucho menor que lo que tardaríamos en armar y configurar un servicio similar en un equipo físico en nuestra casa u oficina.
+ Menores costos operativos: Contratar una MV en la nube nos evita tener que pagar facturas de electricidad, sistemas antirrobo, aire acondicionado o calefacción y alquiler del espacio físico o infraestructura donde se instalarán equipos servidores reales.
+ Fiabilidad: La mayoría de las empresas que ofrecen máquinas virtualizadas en la nube operan en centros de datos muy modernos y equipados con lo mejor en cuanto a conectividad a internet, abastecimiento eléctrico las 24 horas, seguridad y, sobre todo, poseen equipos de muy alto nivel y con gran capacidad de almacenamiento de datos.
+ Potencia escalable: Si lo requieren, pueden aumentar o disminuir la potencia de su conjunto de MV o modificar los parámetros de los servicios que estas requieren. Entre estos cambios tenemos el aumento de la capacidad de procesamiento de datos, la necesidad de cambiar a un software más adecuado para nuestras necesidades, pero más exigente en recursos, un mayor o menor almacenamiento en disco o incluso la asignación de una MV más potente para una tarea específica durante un período de tiempo.

Desventajas de usar MV ubicadas en la nube:

- Menor rendimiento: La transferencia de grandes volúmenes de datos nunca será tan rápida por internet como lo sería a través del disco duro de una máquina anfitriona física y una MV instalada directamente en ella.

- Su tiempo de respuesta o latencia es mayor: Al depender de internet, como acceso a la MV, se enfrentan al problema del lag o retraso en la transmisión de datos y respuesta a cada comando durante las horas del día en que hay mayor tráfico de datos en las redes. Eso implica que será imposible tener una respuesta instantánea de parte de la MV cuando el flujo de internet esté muy saturado o si los servidores de la empresa que maneja la nube tienen mucha carga de trabajo.
- Riesgo a quedar desconectados de nuestras MV: Si vivimos en una zona o país con conexión a internet lenta o irregular, y los datos alojados en ellas durante los momentos en que el servicio sea más lento o se interrumpa. En muchas ocasiones, esto se resuelve contratando 2 o + conexiones con proveedores distintos, pero a su vez esto implica un mayor gasto económico.
- Datos del usuario puedan ser vulnerados: Al depender del acceso a la nube a través de terceros, se corre el riesgo de ser vulnerados por un atacante que logre acceso al sistema del proveedor del servicio de virtualización. En este caso su seguridad dependerá de la capacidad tecnológica del proveedor, por lo que deberán confiar que aplique medidas como instalación de cortafuegos físicos en sus líneas de datos, servicios automatizados de respaldo de datos, instalación de sus servidores en espacios protegidos contra robos y fuego, entre otras.



Escalabilidad, tolerancia al fallo y disponibilidad en la nube

¿Qué es la escalabilidad de la nube?
Se refiere a la capacidad de crecer en capacidad en demanda (habilidad para incrementar su capacidad). Pero no solamente crecer, sino también aumentar su capacidad para ahorrar recursos. A este concepto de crecer y reducir su tamaño se le llama elasticidad, a una infraestructura que permite este comportamiento se la llama elástica. Cuando se utilizan servicios elásticos, la nube debe ser capaz de realizar cobros en base al uso de estos, de forma que no se realicen cargos por recursos que no se encuentran en uso.

¿Qué es la tolerancia al fallo de la nube?
El servicio es capaz de responder a cierto grado de errores sin dejar de ser funcional → tiene la capacidad de solucionar los problemas que existen o reemplazar elementos que no funcionan de forma correcta por otros que sí lo hacen. Un ejemplo puede ser una red local conectada a un switch de muchos puertos alámbricos. Si uno de estos puertos falla, la red puede seguir operando con el resto de los puertos existentes.

¿Qué es la disponibilidad de la nube?
Cuando tenemos múltiples recursos y uno de estos falla, pero hay otros que hagan su trabajo, podemos considerar que existe alta disponibilidad. Si por el contrario, este recurso es único y no puede ser reemplazado, se dice que existe una baja disponibilidad.

Proveedores

- Amazon EC2 → Servicio web que proporciona capacidad informática en la nube segura y de tamaño modificable. Está diseñado para simplificar el uso de la informática en la nube a escala web para los desarrolladores. La sencilla interfaz de servicios web de Amazon EC2 permite obtener y configurar capacidad con una fricción mínima. Proporciona un control completo sobre los recursos informáticos y puede ejecutarse en el entorno informático acreditado de Amazon.
  A su vez ofrece la plataforma informática más amplia y profunda con elección de procesador, almacenamiento, red, S.O. y modelo de compra. Cuentan con las instancias de GPU + poderosas p/ la capacitación de machine learning y las cargas de trabajo gráficas, así cómo las instancias de costo por inferencia + bajas de la nube. En AWS se ejecutan + cargas de trabajo de SAP, HPC machine learning y Windows que en cualquier otra nube.

- Microsoft Azure → Las MV en M.A. tienen soporte p/ S.O. Windows y Linux, e incluso soporte a Windows Server 2003. Podemos configurar, a través de diversas opciones, la memoria RAM asignada y la CPU que va a tener asociada.
  Componentes de la MV:
  **Disco Virtual* → incluye el S.O. instalado. Gracias a este disco virtual podemos iniciar el equipo y guardar información en forma persistente.
  **Placa de red virtual* → al igual que en un equipo físico, es la que me facilitará la conexión con 1 o + redes.
  **Direcciones IP* → debido a ellas podré conectarme al equipo virtual (pueden ser privadas o públicas)

*_El S.O._ → lo que ejecutará como core el equipo virtual. No se puede modificar salvo operaciones avanzadas sobre la VM.
*_Grupos de seguridad de red_ → que nos ayudarán a definir desde qué orígenes nos podemos conectar y hacia qué destinos podemos acceder, teniendo en cuenta protocolos, puertos, etc. Los network security groups son una manera ágil de gestionar los permisos de red, para una o más máquinas. Todos estos componentes son definidos por software. No somos los responsables de mantener los componentes de bajo nivel que permiten su funcionamiento (hardware).
En Microsoft Azure podremos parametrizar dos elementos fundamentales:
*El nombre de la MV: una vez elegido, no puede cambiarse.
*El tamaño: brindan opciones de tamaños preconfiguradas para la CPU, memoria, cantidad de discos soportados y calidad de componentes. Se puede cambiar siempre → "cambio vertical" en el hardware virtual. A veces el equipo → reiniciarse.

- Google Cloud → Una instancia es una MV alojada en la infraestructura de Google. Podemos crear una instancia si utilizamos Google Cloud Console, la herramienta de línea de comando de gcloud o la API de Compute Engine.
  Las instancias de Compute Engine pueden ejecutar las imágenes públicas de Linux y Window Server que proporciona Google, así cómo las imágenes personalizadas privadas que podemos crear o importar desde sus sistemas existentes. También tenemos la posibilidad de implementar contenedores de Docker, que se inician de forma automática en instancias que ejecutan la imagen pública de Container-Optimized OS.
  Si usamos un conjunto de tipos predefinidos de máquinas o creamos nuestros propios tipos personalizados de máquinas, podemos elegir las propiedades de máquina de nuestras instancias, como la cantidad de CPU virtuales y de memoria.
  Instancias y proyectos:
  Cada instancia pertenece a un proyecto de Google Cloud Console y cada proyecto puede tener una o más instancias. Cuando creamos una instancia en un proyecto, especificamos la zona, el sistema operativo y el tipo de máquina de esa instancia. Cuando la borramos, se quita del proyecto.
  Instancias y opciones de almacenamiento
  De forma predeterminada, cada instancia de Compute Engine tiene un pequeño disco de arranque persistente que contiene el sistema operativo. Cuando las aplicaciones que se ejecutan en sus instancias requieren más espacio de almacenamiento, pueden agregar opciones de almacenamiento adicionales a sus instancias.
  Herramientas para gestionar instancias
  Como desarrollamos anteriormente, podemos usar diversas herramientas para crear y administrar instancias, como Google Cloud Console, la herramienta de línea de comandos de gcloud y la API de REST. Si queremos configurar aplicaciones en las instancias, conectémonos a la instancia con Secure Shell (SSH) para instancias de Linux o Remote Desktop Protocol (RDP) para instancias de Windows Server.

CLASE 20 SEMANA 7

ALMACENAMIENTO EN LA NUBE

El almacenamiento en la nube —del inglés cloud storage— permite guardar cualquier tipo de archivos en una infraestructura digital con la comodidad de la administración y backup remotos.
Cada día, más personas y empresas almacenan archivos en servicios como Microsoft OneDrive, Google Docs o Amazon. Reemplaza archivos físicos con la posibilidad de acceder a sus versiones digitales. La información estará siempre almacenada y disponible desde cualquier lugar.

Ventajas:

1) Tiene más espacio físico → Cuando los servidores físicos eran la norma, el principal problema surgía cuando no había espacio suficiente. Almacenar documentos en la nube es perfectamente ajustable a las necesidades de cada empresa, por lo tanto, fácilmente escalable.
2) Compartir archivos con mayor facilidad → Administrar archivos en la nube es más práctico y menos complicado que en un hardware. Los documentos digitales hacen los procesos más ágiles porque todas las etapas son realizadas desde un único sistema en línea, y pueden ser firmados electrónicamente con seguridad, integridad y conformidad.
3) Acceso a la información desde cualquier dispositivo → Si incorporamos documentos, videos, fotos o cualquier tipo de archivos en la nube, tendremos acceso a ellos desde casi cualquier dispositivo y, además, la mayoría de los proveedores nos permite subir, descargar y modificarlos en tiempo real.
   Esto dependerá del sistema operativo y entre algunos de ellos encontramos: Windows; OS X; iOS; Google Chrome OS; Android.
   La tecnología de la nube protegerá los archivos en caso de que falle nuestro dispositivo, pudiendo acceder a los documentos con nuestro usuario y contraseña de forma segura.
4) Diversidad de archivos → Cada empresa tiene necesidades de almacenamiento distintas, ya sea porque maneja archivos muy pesados por la naturaleza del trabajo o porque tiene operaciones que requieren mucho orden y estructura.
   Normalmente las plataformas ofrecen una cantidad de gigabytes gratuitos, por lo que no es necesario que todos los usuarios tengan que pagar una renta para disfrutar de un espacio acorde a sus requerimientos.
5) Optimización automática → Ventaja más importante. Ante la enorme competencia dentro de este sector, es común encontrar cada vez más aspectos que los diferencien entre ellos para atraer mayor cantidad de usuarios y afianzar la fidelidad de quienes ya lo utilizan.

Algunos Ejemplos:

- Dropbox: Es muy sencillo añadir atajos a las carpetas para que se parezca a nuestro escritorio, pudiendo acceder a los documentos de la nube de forma rápida y sencilla.
- iCloud: este servicio se encuentra preinstalado en todos los dispositivos de Apple, haciendo un backup de la mayoría de tu información de forma automática, respaldando tus docs en la nube para brindarnos mayor accesibilidad y seguridad.
- Google Drive: ofrece todo el pack de Google Docs, Slide, Sheets, fotos, entre otros.

¿Qué puedo guardar en la nube?

¿Qué es un Blob?

Los BLOB (Binary Large OBject) son elementos utilizados en las bases de datos para almacenar datos de gran tamaño que cambian de forma dinámica. No todos los sistemas de gestión de bases de datos son compatibles con los BLOB. Generalmente, estos datos son imágenes, archivos de sonido y otros objetos multimedia; a veces se almacenan como códigos de binarios BLOB. Un objeto BLOB representa un objeto tipo fichero de datos planos inmutables. Los BLOBS representan datos que no necesariamente se encuentran en un formato nativo.

¿Qué es un file share?

Los file share son archivos para compartir. El intercambio de archivos es el acto de distribuir o proveer acceso a información almacenada digitalmente, como programas informáticos, obras multimedia (audio, video), documentos o libros electrónicos. Puede ser implementado con distintos tipos de almacenamiento, transmisión y modelos de distribución. Algunos de los métodos más comunes son la distribución manual mediante el uso de medios extraíbles (CD, DVD, disquetes, cintas magnéticas, memorias flash), las instalaciones centralizadas de servidores de archivos en redes informáticas, los documentos enlazados de la World Wide Web y el uso de redes peer-to-peer (P2P) distribuidas.

¿Qué es una tabla en la nube?

Una tabla en la nube es similar a la de una base de datos, se refiere al tipo de modelado de datos donde se guardan los datos recogidos por un programa. Su estructura general se asemeja a la vista general de un programa de tablas. Las tablas se componen de dos estructuras:

Campo: Corresponde al nombre de la columna. Debe ser único y además de tener un tipo de dato asociado.

Registro: Corresponde a cada fila que compone la tabla. Ahí se componen los datos y los registros. Eventualmente pueden ser nulos en su almacenamiento.

Cada tabla creada debe tener un nombre único en cada base de datos, haciéndola accesible mediante su nombre o su seudónimo (alias), dependiendo del tipo de base de datos elegida.

Soluciones de almacenamiento en la nube.

Si nos hemos decidido por el uso del cloud computing para nuestro negocio, hay diferentes factores a tener en cuenta para decidirnos por un tipo de herramienta.

1- Almacenamiento público:

Los datos se guardan en una nube pública, abierta al uso a todas las personas que lo deseen. Este servicio puede ser gratuito o de pago, pudiendo adquirir diferentes planes en función de la capacidad de almacenamiento que necesitemos, el ancho de banda y las diferentes facilidades que nos pueda aportar el proveedor del servicio. Este tipo de almacenamiento lo ofrece Amazon, Azure de Microsoft y Google Engine.

2- Almacenamiento privado:

Cuando la empresa es muy grande o compleja, lo más recomendable es decidirse por una nube privada. Los datos se almacenan en la nube dentro de un entorno local de difícil acceso a todos aquellos que no sean de la empresa, por razones de seguridad. Uno de los sistemas de almacenamiento privado más destacable es Openstack, solución open source.

3- Almacenamiento híbrido:
Este tipo de nube combina las dos anteriores, las empresas pueden aprovecharse de las características de la nube privada, pero recurrir al uso de una nube pública → cuando contrata servicios a 3eros q no están dentro de la propia empresa. Este tipo de soluciones tiene mucho potencial, ya que permite hacer crecer el sistema de almacenamiento de la empresa en base a las necesidades, sin dejar de centralizar los recursos de la misma.

Principales proveedores
Los principales proveedores de servicios en la nube son:

- Microsoft
- Amazon
- IBM
- Salesforce
- SAP

Veremos las soluciones de los dos principales cloud providers: Microsoft y Amazon.

1- Microsoft
Windows Azure Storage
Es la solución de almacenamiento en la nube de Microsoft. Proporciona almacenamiento para objetos de datos, que presenta una alta disponibilidad, es seguro, durable y redundante y se puede escalar de forma masiva.
La plataforma de Azure Storage incluye los servicios de datos siguientes:
● Blobs de Azure: un almacén de objetos que se puede escalar de forma masiva para datos de texto y binarios. También incluye compatibilidad con el análisis de macrodatos a través de Data Lake Storage Gen2.
● Azure Files: recursos compartidos de archivos administrados para implementaciones locales y en la nube.
● Colas de Azure: un almacén de mensajería para mensajería confiable entre componentes de aplicación.
● Tablas de Azure: un almacén NoSQL para el almacenamiento sin esquema de datos estructurados.
● Azure Disks: volúmenes de almacenamiento en el nivel de bloque para máquinas virtuales de Azure.

Características	Descripción	Cuándo se usa
Archivos de Azure	Ofrece recursos compartidos de archivos en la nube totalmente administrados a los que se puede acceder desde cualquier lugar a través del protocolo de bloque de mensajes del servidor (SMB) estándar del sector. Los recursos compartidos de archivos de Azure se pueden montar desde implementaciones de Windows, Linux y macOS en la nube o locales.	Cuando se desee migrar mediante lift-and-shift, una aplicación a la nube que ya usa las API del sistema de archivos nativo para compartir datos entre ella y otras aplicaciones que se ejecutan en Azure. Cuando se quiera reemplazar o complementar los servidores de archivos locales o los dispositivos NAS. Cuando se desee almacenar herramientas de desarrollo y depuración a las que es necesario acceder desde muchas máquinas virtuales.
Azure Blobs	Permite que los datos no estructurados se almacenen y accedan a una escala masiva en Blobs en bloques. También admite Azure Data Lake Storage Gen2 para soluciones de análisis de macrodatos empresariales.	Cuando se desea que la aplicación admita escenarios de streaming y de acceso aleatorio. Cuando se desea poder tener acceso a datos de la aplicación desde cualquier lugar. Cuando se desea crear una instancia empresarial de Data Lake en Azure y realizar análisis de macrodatos.
Azure Disks	Permite que los datos se almacenen y se acceda a ellos desde un disco duro virtual conectado de manera persistente.	Cuando se desea migrar mediante lift-and-shift aplicaciones que usan las API del sistema de archivos nativo para leer y escribir datos en discos persistentes. Cuando se desea almacenar datos a los que no se necesita acceder desde fuera de la máquina virtual a la que está conectado el disco.
Colas de Azure	Permite la puesta en cola de mensajes asincrónicos entre los componentes de la aplicación.	Cuando se quiere desacoplar los componentes de la aplicación y usar la mensajería asincrónica para comunicarse entre ellos.
Tablas de Azure	Permite almacenar datos NoSQL estructurados en la nube, lo que proporciona un almacén de claves y atributos con un diseño sin esquema.	Cuando se quiere almacenar conjuntos de datos flexibles, como datos de usuarios para aplicaciones web, libretas de direcciones, información de dispositivos u otros tipos de metadatos que el servicio requiera.

2- Amazon

Servicios de almacenamiento de AWS

- Almacenamiento de objetos:

● Almacenamiento de archivos:

● Almacenamiento en bloque:

● Transferencias de datos:

● Almacenamiento e informática de borde:

Conclusión

Actualmente tanto Azure Storage como Amazon Storage soportan sitios web estáticos (un sitio web estático se compone de HTML, CSS, JavaScript y otros archivos estáticos, como imágenes o fuentes). Un sitio estático suele ser una aplicación de página única (SPA) escrita con diversos marcos de JavaScript, como Angular, React o Vue. Independientemente de cómo se diseñe la aplicación, puede servir los archivos directamente desde el storage en lugar de usar un servidor web. El hospedaje en el almacenamiento es más sencillo y mucho más económico que el mantenimiento de un servidor web; por lo general, el hospedaje estático cuesta solo unos céntimos al mes. Si necesita procesamiento en el lado servidor, a menudo puede satisfacer esas necesidades mediante funciones sin servidor como las admitidas por cada proveedor.

Amazon S3

Amazon Simple Storage Service (Amazon S3) es un servicio de almacenamiento para Internet. Está diseñado para facilitar la informática de escalada web. Tiene una interfaz de servicios web simple que podemos utilizar para almacenar y recuperar cualquier cantidad de datos, en cualquier momento y desde cualquier parte de la web. Ofrece a cualquier desarrollador acceso a la misma infraestructura de almacenamiento de datos económica, fácilmente escalable, fiable, segura y rápida que utiliza Amazon para mantener su propia red global de sitios web. Este servicio tiene como fin maximizar los beneficios de escalar.

Ventajas:

Se ha desarrollado con un conjunto de características que se centran en la simplicidad y robustez.

- Creación de buckets: podemos crear y nombrar un bucket que almacena datos. Los buckets son los contenedores fundamentales en Amazon S3 para el almacenamiento de datos.
- Almacenamiento de datos en buckets: tenemos la posibilidad de almacenar una cantidad ilimitada de datos en un bucket. Cargamos la cantidad de objetos que deseamos en un bucket de Amazon S3. Cada objeto puede contener hasta 5 TB de datos. Cada objeto se almacena y recupera con una clave única asignada por el desarrollador.
- Descarga de datos: permite descargar nuestros datos, o que otros lo hagan.
- Permisos: brindemos o denegamos acceso a otras personas que desean cargar o descargar datos en nuestro bucket de Amazon S3. Concedemos permisos para cargar y descargar a tres tipos de usuarios. Los mecanismos de autenticación pueden ayudar a proteger los datos del acceso no autorizado.

- Interfaces estándar: utilicemos las interfaces REST y SOAP basadas en estándares diseñados para trabajar con cualquier conjunto de herramientas de desarrollo de Internet.

Conceptos principales:

- S3 Bucket → Un bucket es un contenedor para objetos almacenados en Amazon S3, es decir, cada objeto está almacenado en un bucket. Por ejemplo, si el objeto denominado photos/puppy.jpg se almacena en el bucket awsexamplebucket1 en la región EE. UU. Oeste (Oregón), es direccionable mediante la URL: https://awsexamplebucket1.s3.us-west-2.amazonaws.com/photos/puppy.jpg.
  Los buckets sirven a diversos fines:
  -Organizan el espacio de nombres de Amazon S3 al más alto nivel.
  -Identifican la cuenta responsable para los cargos de almacenamiento y transferencia de datos.
  -Juegan un papel en el control de acceso.
  -Sirven como la unidad de agregación para informes de uso.

- Objetos → Los objetos son las entidades fundamentales almacenadas en Amazon S3, se componen de datos de objetos y metadatos. Los metadatos son conjuntos de pares nombre-valor que describen el objeto. Los objetos incluyen algunos metadatos predeterminados, como la fecha de la última modificación y los metadatos HTTP estándar, como Content-Type. También podemos especificar metadatos personalizados en el momento en que se almacena el objeto.
  Un objeto se identifica de forma exclusiva dentro de un bucket con una clave (nombre) y un ID de versión.

- Claves → Una clave es el identificador único de un objeto dentro de un bucket. Cada objeto de un bucket tiene una clave. La combinación de un bucket, clave e ID de versión identifican de forma única cada objeto. Por lo tanto, se puede pensar en Amazon S3 como una asignación de datos básica entre "bucket + clave + versión" y el objeto en sí. Se puede acceder a cada objeto de Amazon S3 de forma exclusiva a través de la combinación de punto de enlace de servicio web, nombre del bucket, clave y, de forma opcional, una versión. Por ejemplo, lo podemos ver en la siguiente URL: https://doc.s3.amazonaws.com/2021-03-01/AmazonS3.wsdl, donde "doc" es el nombre del bucket y "2021-03-01/AmazonS3.wsdl" es la clave.

- Regiones → Podemos elegir la región geográfica de AWS donde Amazon S3 almacenará los buckets que creamos. Esta elección puede optimizar la latencia, minimizar los costos o cumplir con requisitos legales. Los objetos almacenados en una región nunca la abandonan, a menos que se transfieran expresamente a otra región. Por ejemplo, los objetos almacenados en la región UE (Irlanda) nunca salen de ella.

- Identity and Access Management → Utilicemos AWS Identity and Access Management (IAM) para administrar el acceso a nuestros recursos de Amazon S3. Por ejemplo, podemos usar IAM con Amazon S3 para controlar el tipo de acceso que tiene un usuario o un grupo de usuarios a partes concretas de un bucket de

Amazon S3 que es propiedad de nuestra cuenta de AWS.

- Operaciones → A continuación, les compartimos las operaciones más comunes que ejecutaremos a través de la API.
  • Crear un bucket: además, podemos nombrarlo para almacenar los objetos.
  • Escribir un objeto: crear o sobreescribirlo para almacenar datos. Cuando escribimos un objeto, debemos especificar una clave única en el espacio de nombres del bucket. En esa instancia es cuando debemos especificar cualquier control de acceso que deseamos aplicar en el objeto.
  • Leer un objeto: releamos los datos, se pueden descargar a través de HTTP.
  • Eliminar un objeto: podemos eliminar algunos de sus datos.
  • Enumerar claves: indiquemos las claves incluidas en uno de los buckets. Podemos filtrar la lista de claves en función de un prefijo.

CLASE 22 SEMANA 8

CLOUD COMPUTING, UNA MIRADA HOLÍSTICA E INTEGRADORA

¿Qué otros servicios se ofrecen más allá de compute y storage?

¿Qué es la informática sin servidor?

Serverless Cloud Computing → Se trata de un modelo de desarrollo nativo de la nube que permite que los desarrolladores diseñen y ejecuten aplicaciones sin tener que gestionar servidores. Si bien se utilizan servidores, se encuentran aislados de la etapa de desarrollo de las aplicaciones. El proveedor de nube se encarga de las tareas rutinarias de preparación, mantenimiento y adaptación de la infraestructura de los servidores. Los desarrolladores solo deben empaquetar el código en contenedores para implementarlo.

Una vez que se instalan las aplicaciones sin servidor, responden a la demanda y se amplían o reducen automáticamente en función de las necesidades. Por lo general, los proveedores de nube pública ofrecen estas tecnologías a un costo que se mide a través de un modelo de ejecución basado en eventos, según se soliciten. Por eso, cuando una función permanece inactiva, no se genera ningún cargo

Visión general de la arquitectura sin servidor.
Lo que distingue a la informática sin servidor de los demás modelos de cloud computing es que el proveedor de nube se encarga de gestionar tanto la infraestructura de nube como la expansión de las aplicaciones. Aquellas sin servidor se implementan en contenedores que se inician automáticamente cuando se solicitan.

En el caso de los modelos estándar de cloud computing de **infraestructura como servicio** (**IaaS**), los usuarios adquieren cierta capacidad por adelantado, lo cual significa que pagan a un proveedor de nube pública por ciertos elementos del servidor que funcionan de forma permanente para ejecutar sus aplicaciones. El usuario debe encargarse de ampliar la capacidad durante los períodos de mayor demanda y reducirla cuando ya no haga falta. La infraestructura de nube requerida para ejecutar las aplicaciones permanece activa incluso cuando no se utilizan.

En cambio, la **arquitectura sin servidor** permite que se inicien solo cuando sea necesario. Cuando un evento activa el funcionamiento del código de las aplicaciones, el proveedor de nube pública le asigna los recursos correspondientes de forma dinámica, y el usuario deja de pagar cuando termina de ejecutarse. La informática sin servidor no solo ofrece beneficios en cuanto a los costos y la eficiencia, sino que también permite que los desarrolladores ya no deban encargarse de las tareas rutinarias y poco relevantes.

¿Qué función desempeña el proveedor de nube en la informática sin servidor?
Los proveedores de nube ejecutan los servidores físicos y asignan los recursos de forma dinámica para los usuarios que implementan el código directamente en la producción.

Normalmente, las ofertas de informática sin servidor se clasifican en dos grupos:
Back end como servicio (BaaS) y Función como servicio (FaaS).
El primero otorga a los desarrolladores acceso a diferentes aplicaciones y servicios externos. Por ejemplo, un proveedor de nube puede ofrecer servicios de autenticación, cifrado adicional, bases de datos a las que se puede acceder desde la nube e información muy confiable sobre el uso. Las funciones sin servidor se suelen solicitar mediante llamadas a las interfaces de programación de aplicaciones (API).

Ventajas	Desventajas
• Reduce el costo operativo y aumenta la productividad del desarrollador. • Se reducen los costos operativos, ya que puede pagar por el tiempo de procesamiento en la nube según sea necesario, en lugar de ejecutar y gestionar sus propios servidores todo el tiempo.	• Los proveedores de nube tienen restricciones estrictas en cuanto a la interacción con sus elementos, lo cual a su vez determina el grado de flexibilidad y personalización de sus propios sistemas. • Probablemente, la decisión de cambiar de proveedor también implique un costo de actualización de los sistemas, para cumplir con las especificaciones de la nueva empresa.

¿Qué es una nube híbrida?
Es una arquitectura de IT que incorpora cierto grado de gestión, organización y portabilidad de las cargas de trabajo en dos o más entornos. Según a quién le consulte, es posible que esos entornos deban incluir lo siguiente:

- Al menos una nube privada y una pública.
- Dos o más nubes privadas.
- Dos o más nubes públicas.
- Un entorno virtual o sin S.O. conectado a al menos una nube, ya sea pública o privada.

Funciones de las nubes híbridas:
Todas las nubes híbridas deben poder realizar lo siguiente:

- Conectar varias computadoras a través de una red.
- Consolidar los recursos de IT.
- Escalar horizontalmente e implementar los recursos nuevos con rapidez.
- Poder trasladar las cargas de trabajo entre los entornos.
- Incorporar una sola herramienta de gestión unificada.
- Organizar los procesos con la ayuda de la automatización.

¿Cómo funcionan las nubes híbridas?
La forma en que las nubes públicas y privadas funcionan como parte de una nube híbrida es similar a cómo lo hacen de forma independiente:

- Una red de área local (LAN), una red de área amplia (WAN), una red privada virtual (VPN) y las interfaces de programación de aplicaciones(API) conectan varias computadoras entre sí.
- La virtualización, los contenedores o el almacenamiento definido por software extraen los recursos, que pueden agruparse en lagos de datos.
- El sistema de software de gestión asigna esos recursos a entornos donde las aplicaciones pueden ejecutarse, los cuales luego se implementan, según se solicite, con la ayuda de un servicio de autenticación.

Las nubes independientes se vuelven híbridas cuando esos entornos se conectan de la forma más sencilla posible. Esa interconectividad es lo único que permite que las nubes híbridas funcionen, y es por eso que estas nubes son la base del edge computing. Además,

determina la forma en que se trasladan las cargas de trabajo, se unifica la gestión y se organizan los procesos. La calidad de las conexiones tiene un efecto directo sobre el funcionamiento de su nube híbrida.

Arquitectura moderna de la nube híbrida

Actualmente, las nubes híbridas ya no necesitan una red amplia de API para trasladar las cargas de trabajo de una nube a otra. Para diseñar nubes híbridas, los equipos modernos de IT :

- Ejecutan el mismo sistema operativo en todos los entornos de IT;
- Desarrollan e implementan aplicaciones como grupos de servicios pequeños, independientes y sin conexión directa; y
- Gestionan todo con una PaaS unificada.

¿Las nubes híbridas son seguras?

Una nube híbrida diseñada, integrada y gestionada de forma correcta puede ser tan segura como una infraestructura de IT local. Aunque hay algunos desafíos exclusivos de la seguridad de la nube híbrida (como la migración de datos, el aumento de la complejidad y una mayor superficie de ataque), la presencia de varios entornos puede constituir una de las defensas más sólidas contra los riesgos de seguridad. Gracias a todos esos entornos interconectados, las empresas pueden elegir dónde colocar los datos confidenciales en función de los requisitos, y los equipos de seguridad pueden adoptar de manera uniforme un sistema de almacenamiento en la nube que sea redundante y pueda aumentar los esfuerzos de recuperación ante desastres.

¿Cómo conecto mi centro de cómputos a la nube?

Point-to-Site

Una VPN Point-to-Site nos permite crear una conexión segura a la red virtual desde un equipo cliente individual. Se establece una conexión de punto a sitio al iniciarla desde el equipo cliente. Esta solución resulta útil para los teletrabajadores que deseen conectarse a redes virtuales desde una ubicación remota, por ejemplo, desde una casa. La conexión VPN de punto a sitio también es una solución útil en comparación con la conexión VPN de sitio a sitio cuando solo necesitan conectarse a la red virtual algunos clientes.

Site-to-Site

Una VPN Site-to-Site (de sitio a sitio) se utiliza principalmente en las empresas. Las organizaciones con instalaciones en diferentes ubicaciones geográficas utilizan esta VPN para conectar la red de una instalación a la red en otra.