

INFRAESTRUCTURA I

Zoé Agustina Tira

[NOMBRE DE LA EMPRESA] [Dirección de la compañía]

MÓDULO 1 – INMERSIÓN	2
C1A – INTRODUCCIÓN A LA MATERIA & THE BIG PICTURE	2
<i>Introducción a la materia</i>	2
<i>El centro de cómputos</i>	2
MÓDULO 2 – AUTOMATIZACIÓN	6
C2A – AUTOMATIZACIÓN	6
<i>Automatización de la infraestructura</i>	6
<i>Virtualización</i>	8
C4A – SHELL SCRIPTING – PARTE I	9
<i>Introducción a la terminal de Linux</i>	9
<i>Comandos útiles</i>	10
C5A – SHELL SCRIPTING – PARTE II	12
<i>Scripting</i>	12
C7A – POWERSHELL	15
<i>Introducción a PowerShell</i>	15
C8A – PYTHON.....	19
<i>Introducción a Python</i>	19
C10A – CONFIGURATION MANGEMENT	20
<i>Configuration Management</i>	20
C11A – CONFIGURATION MANAGEMENT – ANSIBLE	24
<i>Conociendo Ansible</i>	24
MÓDULO 3 – CONTAINERS.....	27
C13A – DOCKER EN PROFUNDIDAD	27
<i>Docker en profundidad</i>	27
C14A – EL ECOSISTEMA DE DOCKER Y MEJORES PRÁCTICAS	33
<i>Creación del primer repositorio:</i>	33
<i>Docker Networking:</i>	35
<i>Documentación de Docker:</i>	36
<i>Docker Compose</i>	36
MÓDULO 4 – CLOUD COMPUTING	38
C16A – CLOUD COMPUTING, UNA MIRADA HOLÍSTICA E INTEGRADORA	38
<i>¿Qué es el serverless?</i>	38
C17A – COMPUTACIÓN EN LA NUBE	41
<i>Modelo de responsabilidades</i>	41
<i>Máquinas virtuales en la nube:</i>	42
<i>Escalabilidad, tolerancia al fallo y disponibilidad en la nube</i>	44
<i>Proveedores</i>	45
C19A – CLOUD COMPUTING: REDES (VPC + ELB).....	47
<i>Concepto de red de VPC</i>	47
<i>Concepto de elasticidad vs escalabilidad</i>	47
<i>AWS VPC</i>	49
C20A – ARMAMOS UN PEQUEÑO AMBIENTE EN AWS.....	51
<i>¿Qué es Load Balancer y para qué sirve?</i>	51
C22A – ALMACENAMIENTO EN LA NUBE	52
<i>Almacenamiento en la nube</i>	52
C23A – BASES DE DATOS EN LA NUBE	58
<i>Bases de datos como servicio</i>	58
<i>RSD</i>	59

MÓDULO 1 – INMERSIÓN

C1A – INTRODUCCIÓN A LA MATERIA & THE BIG PICTURE

Introducción a la materia

La función del administrador de infraestructura es brindar respuestas a los problemas habituales de hardware y software en cualquier ámbito informatizado. Este servicio debe estar en manos de profesionales que orienten, configuren, prevengan y resuelvan eventualidades, manteniendo sus equipos en un estado óptimo. Debe ser capaz de prestar servicios de administración y soporte de sistemas de base y elementos de infraestructura para el procesamiento de aplicaciones informáticas, tales como servidores, dispositivos de almacenamiento masivo, otros dispositivos de hardware, sistemas operativos,, entre otros. Podrá brindar servicios de administración de la infraestructura tecnológica en la cual opera el software de estas aplicaciones interviniendo en forma puntual para resolver los problemas que experimente esa infraestructura o su eficiencia operativa y realizar un diagnóstico de incidentes que se presenten en la operatoria habitual del sistema.

¿Qué es infraestructura de IT?: La infraestructura de tecnología de la información, o infraestructura de IT, se refiere a los componentes combinados necesarios para el funcionamiento y la gestión de los servicios de IT de la empresa y los entornos de IT.

El centro de cómputos

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.

La implementación cliente-servidor debe tener las siguientes características:

- Transparencia e independencia del hardware y software.
- 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, aumentando su número o su capacidad de cálculo —*escalabilidad vertical*—.

Características de los tres componentes de la arquitectura cliente-servidor:



Servidor: Genéricamente un servidor es un ordenador, pero con prestaciones elevadas. Sin embargo, desde este enfoque, un servidor es un proceso que ofrece recursos y servicios a los clientes que lo solicitan (back end). Según el tipo de servidor implantado, tendremos un tipo de arquitectura cliente-servidor diferente. A su vez, debido a que los programas y datos se encuentran centralizados, se facilita la integridad y el mantenimiento.

Cliente: Es una computadora, normalmente con prestaciones ajustadas, sin embargo, en entornos cliente-servidor, se utiliza el término front end, ya que es un proceso que solicita los servicios del servidor a través de una petición del usuario. Un proceso cliente se encarga de interactuar con el usuario, por lo que estará construido con alguna herramienta que permita implementar interfaces gráficas (GUI).

Middleware: Es la parte del software del sistema que se encarga del transporte de los mensajes entre el cliente y el servidor y facilita la interconexión de sistemas heterogéneos sin utilizar tecnologías propietarias. Por lo cual, se ejecuta en ambos lados de la estructura. Permite independizar a los clientes y a los servidores, ofrece más control sobre el negocio, debido a que permite obtener información desde diferentes orígenes —uniendo tecnologías y arquitecturas distintas— y ofrecer de manera conjunta. Otra característica es que los sistemas están débilmente acoplados ya que interactúan mediante el envío de mensajes.

La importancia del centro de datos:

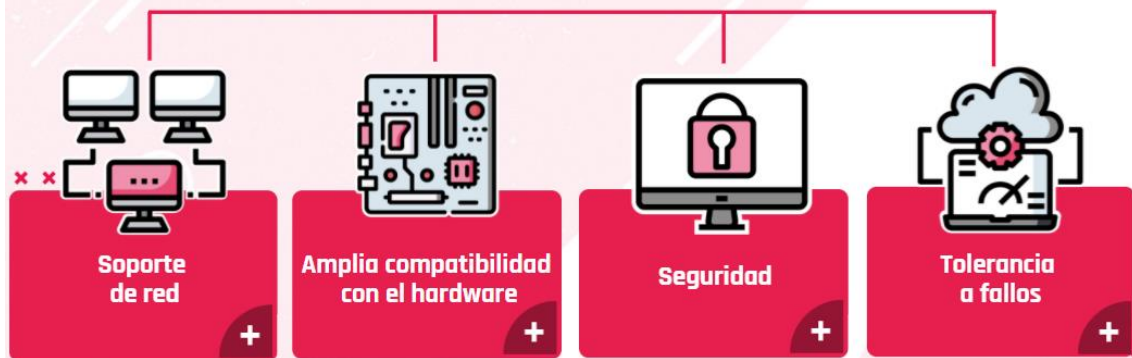
Las peticiones o request son emitidas por un cliente, éstas definen que tipo de información se va a requerir y a donde. El cliente y el servidor no hablan el mismo lenguaje, por lo que aparece el middleware, que es una capa de software que se va a encargar de traducir nuestro pedido para que sea comprendido por el servidor. Dentro de los middleware se encuentran los DBMS, un sistema de administración de base de datos que permiten crear, recuperar, actualizar y administrar datos. También se encuentran las APIs, interfaz de programación de aplicaciones que brindan un conjunto de subrutinas, funciones y procedimientos para ser utilizados en otro software.

Una vez recibida la petición, el servidor la procesa y prepara una respuesta llamada response, ella pasara por el middleware para ser traducida y finalmente el cliente la interpreta y la muestra.

Datacenter: Es un lugar específicamente encargado de almacenar sistemas de información y comunicación, los cuales están debidamente protegidos contra incendios, accesos indebidos e interrupción de energía.

Sistema Operativo (S.O): Conjunto de programas capaces de administrar recursos físicos del servidor así como los protocolos de ejecución.

Características fundamentales de un sistema operativo



Soporte de red: Es indispensable que tengan un soporte completo para poder brindar conectividad.

Amplia compatibilidad con el hardware: Un punto fundamental para el aprovechamiento pleno de las características del servidor es que el S.O. sea capaz de exprimir al máximo las características técnicas del hardware en donde se ejecuta, es por ello que se debe priorizar el uso de S.O. actualizados y con un soporte importante de controladores.

Seguridad: Es de vital importancia que el S.O. instalado sea seguro; eso implica no solo que este actualizado con todos los parches/actualizaciones, sino que además debe tener aplicadas políticas estrictas de acceso para prevenir accesos no autorizados o ataques. Adicionalmente esa seguridad debe reforzarse con la instalación de Firewalls y antivirus. En este ítem también debemos incluir el respaldo de la información, ya sea por medio de herramientas que el propio S.O. nos ofrezca o instalando software externo, con el propósito de tener la menor pérdida de datos posible en caso de fallos fatales.

Dispositivos físicos de protección: Debemos priorizar S.O. que en su arquitectura permita una tolerancia a fallos, ya sea 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.

Diferencias entre S.O comunes y S.O para servidores:

Manejo diferente del hardware: Debido al diferente propósito, los S.O. de estaciones de trabajo no pueden aprovechar todo el hardware disponible, por ejemplo, el manejo de memoria RAM —teniendo el caso de Windows 10 64-bit que puede manejar 6TB de RAM, mientras que Windows Server 2019 alcanza los 24TB—.

Características soportadas: Hay funcionalidades que nativamente un S.O. de estación de trabajo no es capaz de brindar, ya que en su versión de kernel están limitadas o deshabilitadas.

Soporte: Algo muy importante a tener en cuenta es, cuando nuestro negocio o aplicación depende de un S.O., es el soporte por parte del fabricante/desarrollador. En el caso de los S.O. de estación de trabajo, el soporte/cobertura que tenemos es para un uso específico, si sobre esa base quisiéramos desplegar una arquitectura, por ejemplo, de servidor web, si bien es probable que nos funcione, vamos a carecer de soporte técnico ya que el fabricante nos indicará que para ese propósito esta la versión “Server” del producto.

Opciones disponibles para un S.O. de servidor:

En el grupo de los UNIX (o UNIX-like) tenemos a:

- GNU Linux
- FreeBSD
- macOSServer
-

En el grupo de Windows, integrado por toda la familia de Windows Server, tenemos las versiones que van desde la versión 2003 a 2019.

MÓDULO 2 – AUTOMATIZACIÓN

C2A – AUTOMATIZACIÓN

Automatización de la infraestructura

La Infraestructura IT es el conjunto de dispositivos y aplicaciones de software necesarios para que cualquier empresa opere. Ésta se compone de elementos como: software, hardware, redes, instalaciones y todo lo que se requiera para desarrollar, controlar, monitorear y dar soporte a los servicios que ofrece el departamento de IT.

Automatización: Consiste en usar la tecnología para realizar tareas casi sin necesidad de las personas. Se puede implementar en cualquier sector en el que se lleven a cabo tareas repetitivas.

Automatización de la IT: —también denominada automatización de la infraestructura— consiste en el uso de sistemas de software para crear instrucciones y procesos repetibles a fin de reemplazar o reducir la interacción humana con los sistemas de IT.

¿Por qué automatizar?: Automatizar tareas permite ganar tiempo y maximizar la productividad de nuestra infraestructura IT.

Beneficios de automatizar:

- Elevar la productividad empresarial.
- Reducir costos operativos.
- Disminuir los riesgos de fallas.
- Elevar la seguridad de la información.
- Tener una mejor capacidad de respuesta.
- Facilidad de adaptación.
- Alojamiento una mayor cantidad de datos.
- Elevar la competitividad del negocio.

5 tareas más comunes para automatizar en TI:

1. *Aprovisionamiento de las áreas de TI:* Automatiza el proceso para habilitar máquinas a través de un levantamiento de servicio para el personal nuevo de nuestra organización
2. *Gestión de configuración:* Ahorra tiempo y trabajo al automatizar la configuración de las máquinas que se habilitan de acuerdo con los objetivos que tengan el área de TI.
3. *Seguridad y cumplimiento:* Establece acciones de políticas de seguridad y cumplimiento automatizables en la gestión de las máquinas.
4. *Organización de la nube:* Asegura la información y mantiene la disponibilidad de ella a través de la automatización para generar mayor eficiencia.
5. *Implementar aplicaciones:* Algunas aplicaciones requieren instalarse o configurarse, esta tarea puede ser automatizada en las áreas de TI.

Configuración y mantenimiento del sistema:

Infraestructura y servicios: Primeramente tenemos que analizar los dispositivos que tenemos en nuestra red de IT. Si contamos con servidores legacy o servicios contratados por un cloud providers. De las dos opciones anteriores, lo que tenemos que tener en cuenta es el sistema operativo con el que trabajan (MAC, Linux, Windows).

De los cloud providers más conocidos contamos con:

- AWS
- Google Cloud
- Microsoft Azure

Manejo del código: Podemos automatizar el deploy de nuestro código con CI/CD (refiere a las prácticas combinadas de integración continua y entrega continua). Algunas herramientas para hacerlo son:

- Jenkins
- GitActions
- JetBrains

Contenedores: Los contenedores se están convirtiendo en el modelo de empaquetado de software del producto que desarrollamos. Permiten la virtualización de ambientes de trabajo compatibles transportables, totalmente configurados para que nuestro código funcione en todos los equipos. El más popular es Docker. Cuando manejamos muchos contenedores, tenemos que migrar a un clúster de contenedores, dirigidos por los orquestadores de contenedores. Por ejemplo: Kubernetes o Docker Swarm.

Ambientes de trabajo: Hay que tener en cuenta en qué ambiente está nuestro clúster.



Los ambientes más populares son:

- Ansible
- Chef
- Puppet
- Terraform

Monitores de red: Es muy importante tener la supervisión de los dispositivos que hay en nuestra red y de los servicios, y programar alertas en caso de que suceda algún cambio. Para esto podemos utilizar: Nagios, Prometheus, Icinga2 o DataDog.

Lenguajes de scripting: Necesitamos trabajar en estrecha colaboración con los desarrolladores y el administrador del sistema para automatizar tareas de los operadores y desarrolladores (como pueden ser backups, cron jobs, system monitoring). Según el sistema operativo, podemos usar: Bash o PowerShell. Pero también existen lenguajes de scripting independientes del S.O, como Python, Ruby o Go.

Conclusión: Siempre hay que tomar la decisión basándose en la necesidad, recursos disponibles y experiencia que cuenta la empresa.

Virtualización

Introducción a la virtualización: La virtualización permite mejorar la agilidad, la flexibilidad y la escalabilidad de la infraestructura de IT, al mismo tiempo que proporciona un importante ahorro de costos.

Ventajas de la virtualización son:

- Mayor movilidad de las cargas de trabajo.
- Aumento del rendimiento.
- Menos esfuerzo en los upgrades/updates del sistema.
- Mejor disponibilidad de los recursos o la automatización de las operaciones:
 - Simplifican la gestión de la infraestructura de IT.
 - Permiten reducir los costos de propiedad y operativos.

Componentes de virtualización:



Máquinas virtuales: Se pueden crear sistemas operativos guest del tipo Microsoft y Linux en casi todas sus versiones, se deben tener en cuenta las versiones de cada sistema operativo y la compatibilidad con el sistema host.

Administrador de máquinas virtuales: Desde la herramienta de management se administran todos los recursos físicos y virtuales de los guest que son las instancias virtuales que se crean para usos específicos. Desde el mismo management podremos establecer clustering con otros virtual machines manager para tener alta disponibilidad y tolerancia a fallos. Además, administra todos los recursos virtuales de nuestras virtual machines.

S.O base: Es el sistema operativo encargado de administrar los dispositivos físicos (hardware) y proveer una capa de abstracción a los entornos virtuales.

Hardware (servidores físicos): Los microprocesadores, tanto los de Intel como los de AMD, tienen una característica llamada virtualización de CPU. Se aplica a servidores o máquinas de escritorio.

C4A – SHELL SCRIPTING – PARTE I

Introducción a la terminal de Linux

La consola: La interfaz de línea de comandos, o CLI — command-line interface—, es un método de comunicación entre usuario y máquina que acepta instrucciones del usuario a través de líneas de texto. La herramienta que posibilita la función de interfaz de usuario se la denomina shell.

Diferentes tipos de Shell: En Linux tenemos una multitud de shells. El más conocido de todos probablemente es Bash, debido a que es el que suele venir por defecto en la gran mayoría de distribuciones GNU/Linux, pero también destacan otros como Bourne Shell (sh), Korn Shell (ksh) o C Shell (csh), los cuales vamos a conocer.

- *Bourne Shell – sh:* Primer shell utilizada para el sistema operativo Unix. Todas las versiones de Linux Unix permiten a los usuarios cambiar a la original Bourne Shell si así lo desean. Sin embargo, hay que tener en cuenta que al hacerlo, se renuncia a funcionalidades como el completado de nombres de archivo y el historial de comandos que los depósitos posteriores han añadido.
- *C/TC Shell – csh tcsh:* Está pensado en facilitar el control del sistema al programador en lenguaje C, ya que su sintaxis es muy similar a la de este lenguaje. Conocido popularmente también como csh, está presente en otros SO, por ejemplo, en Mac OS. Posee una evolución, conocida como tcsh que incorpora funcionalidades avanzadas y mayores atajos de teclado.
- *Korn Shell:* Intenta combinar las características de la C Shell, TC Shell y Bourne Shell en un solo paquete. También incluye la capacidad para crear nuevos comandos de shell para los desarrolladores cuando surja la necesidad. Posee funciones avanzadas para manejar archivos de comandos que la colocan a la par de lenguajes de programación especializados, como AWK y Perl.
- *Bourne-Again Shell (BASH):* Es una versión actualizada de la Bourne Shell original. Su sintaxis es similar a la utilizada por la Bourne Shell, incorporando funcionalidades más avanzadas que se encuentran en las shells C, TC y Korn. Entre las funcionalidades está la capacidad para completar nombres de archivos pulsando la tecla TAB, la capacidad de recordar un historial de comandos recientes y la capacidad de ejecutar múltiples programas en segundo plano a la vez.

Ejecución de la consola: Si bien cada distribución de Linux tiene su manera particular de acceder a la consola, cuando el SO se inicia en los niveles 1, 2, 3 y 4 nos llevará por defecto a la consola. Si nuestro S.O inicia en nivel 5, inicia con interfaz gráfica (GUI). Estas varían de acuerdo a la distribución instalada. En el caso de Ubuntu, tenemos dos opciones:

- La primera de ellas es lanzando un TTY, o espacio de trabajo sin entorno gráfico. Podemos ejecutar 7 terminales al mismo tiempo de esta forma. De la 1 a la 6, ninguna tiene interfaz gráfica. Para cambiar de TTY en Linux debemos usar el atajo de teclado Control+Alt más la tecla —de F1 al F7— del TTY que queramos ejecutar.
- La segunda opción es encontrar una app dedicada que se ejecuta en una ventana, dentro del panel de aplicaciones de nuestra distro. En el caso de Ubuntu, por ejemplo, podemos encontrar esta terminal dentro del cajón de programas del entorno gráfico GNOME.

Elevación de privilegios: Los sistemas operativos contemplan el uso de solo un usuario, el cual tiene permisos de administrador. En Linux se separa la cuenta de usuario común de la de superusuario y es eso lo que conocemos como root. Esta cuenta *posee todos los privilegios y permisos para realizar acciones sobre el sistema*.

Para la ejecución de algunos comandos debemos ingresar dicho acceso (clave de root). El uso de instrucciones con privilegios de superusuario pueden ser sumamente útiles, pero totalmente devastadoras si desconocemos las consecuencias de su uso en el sistema. (Se usa la palabra sudo).

Comandos útiles

Comandos para el manejo de archivos:

ls: Lista los diferentes archivos y directorios de la carpeta de trabajo en la que te encuentres. El comando acepta multitud de opciones, algunas de las cuales veremos a continuación.

ls -a: Mostrará —en forma de lista— todo el contenido que se encuentre dentro del directorio de trabajo, incluyendo archivos y carpetas ocultos.

ls -l: Muestra el contenido en forma de lista e incluye información referente a cada elemento.

mkdir: Crea un directorio con el nombre y la ruta que especifiques. Si no se le indica ninguna ruta, por defecto, creará la carpeta dentro del directorio de trabajo en el que te encuentres.

rmdir: Elimina un directorio, el mismo debe estar vacío.

rm: Elimina archivos sueltos y directorios que ni se encuentren vacíos.

rm -r: Elimina el directorio y recursivamente todo su contenido.

cp: Copia archivos y directorios, también los ubica en otras rutas, definiendo origen primero y luego el destino.

mv: Mueve archivos, se debe especificar la ubicación de inicio y la de destino. También se usa para renombrar archivos, utilizando las rutas originales.

Comandos para leer archivos de texto:

cat: Crea un archivo e imprime por pantalla su contenido. Invocando el comando con > se abre un archivo para editarlo, con CTRL+D se termina la edición y se guarda el contenido. Sin el piquito, se lee el archivo.

cat -n: Numera las líneas.

cat -b: No muestra las líneas en blanco.

more: Imprime por pantalla el contenido de un archivo de texto. Mejor para leer archivos largos.

nano: Edita textos.

- CTRL+R: Le indica a un archivo de texto a Nano que lo abra y muestre su contenido por la consola.
- CTRL+V: estando dentro de Nano y con el archivo abierto en la consola, esta combinación sirve para avanzar a la página siguiente.
- CTRL+Y: sirve para retroceder a la página anterior.
- CTRL+W: sirve para introducir un carácter o grupo de caracteres y buscar en el texto cualquier letra o palabra que coincida con el parámetro de búsqueda.
- CTRL+X: para cerrar el archivo una vez que lo hayas terminado de visualizar en la consola. Eso cerrará el editor de texto Nano y volverá a aparecer el prompt de Bash por consola.

grep: Busca un patrón que definamos en un archivo. Su primer parámetro es la cadena de texto a buscar, luego el archivo, -r es para recorrerlo recursivamente.

tee: Lee una entrada estándar y la escribe en la salida estándar y en uno o más archivos.

cURL: Es una abreviatura de “Client URL”. Están diseñados para funcionar como una forma de verificar la conectividad a las URL y como una gran herramienta para transferir datos.

El comando tiene una amplia compatibilidad con los protocolos más usados, FTP, HTTP, POP / SMTP / IMAP, SCP.

curl + URL: Muestra la página de la URL que se le indica.

curl + URL -o: Escribimos el contenido de la página en un archivo en nuestro equipo.

curl + URL -v: Verifica la conectividad a un sitio.

curl + URL -I: Muestra los encabezados de la solicitud, tales como la ruta por defecto, publicador web, etc.

Comando jq: JSON es un formato de datos estructurados que se utiliza normalmente en la mayoría de las API y servicios de datos modernos. Es particularmente popular en aplicaciones web debido a su naturaleza liviana y compatibilidad con JavaScript.

Shells como Bash no pueden interpretar y trabajar con JSON directamente. Esto significa que trabajar con JSON a través de la línea de comando puede ser engorroso e implica la manipulación de texto utilizando una combinación de herramientas como sed y grep.

jq se basa en el concepto de filtros que funcionan sobre un flujo de JSON. Cada filtro toma una entrada y emite JSON a la salida estándar.

Tomando el archivo JSON obtenido con cURL, una ejecución sencilla de jq nos devuelve todo el contenido del JSON.

Para poder acceder a una propiedad específica es necesario indicarla luego del punto, con el nombre de la misma. Si queremos acceder a varias propiedades, las separamos por coma.

C5A – SHELL SCRIPTING – PARTE II

Scripting

Bash: Interfaz que interpreta las ordenes que el usuario le hace al sistema. También se pueden leer y ejecutar ordenes desde un archivo llamado script. Es un lenguaje de scripting, por lo que es muy buena para la administración de sistemas y automatización de tareas.

La primera línea define que Shell usaremos.

```
1  #!/bin/bash
2  # This is a comment
3  pwd
4  whoami
```

Luego de guardarlo, configuramos el archivo para que sea ejecutable, con `chmod +x`. Luego lo ejecutamos con `./nombreArchivo`.

Tipos de variables

- *Globales o de entorno*: Son en mayúsculas. Para ver las variables que estamos usando y que están cargadas en nuestra sesión, escribimos `printenv` o `env`. Para declararla se hace con `export NOMBREVARIABLE=valor`. Para acceder a la misma `$NOMBREVARIABLE`.
- *De usuario o locales*: Pueden ser accedidas solo por el usuario y la sesión en la que fueron creadas. Se declara como `nombrevariable=valor`. Para acceder `$nombrevariable`.

Estructuras de control

Sentencia if-then: Los scripts de Bash necesitarán condicionales. La estructura más básica de la sentencia if-then es así:

```
if command; then
    hacer algo
fi
```

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

Sentencia if-then:

```
if comand; then
    hacer algo
else
    hacer otra cosa
fi
```

```
#!/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
```

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.



Ej:

```
#!/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 cadena

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 mayor que cero.
-z string1	Comprueba si string1 tiene una longitud de cero.



Ej:

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

Cálculos matemáticos

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

C7A – POWERSHELL

Introducción a PowerShell

PowerShell (PS): Es una interfaz con líneas de comando (CLI) que tiene la posibilidad de ejecutar scripts y facilita la configuración, administración y automatización de tareas multiplataforma. Tiene una salida basada en objetos, por lo que acepta y devuelve objetos. La posibilidad de poder tratar con objetos es la principal diferencia de PS respecto a otras CLI. Los comandos que usa, llamados cmdlet, devuelve una instancia a un objeto que da lugar a una información de salida mucho más completa. Permiten un conjunto de comandos extensibles y brinda la opción de crear nuevos cmdlet a partir de un script. Otra característica es que trabaja con alias de comandos.

Se utiliza para:


- Automatizar la administración de sistemas.
- Compilar, probar e implementar soluciones a menudo en entorno de CIC.

Características:

- Extensible mediante funciones, clases, scripts y módulos.
- Sistema de formato extensible para una salida fácil.
- Sistema de tipo extensibles para crear tipos dinámicos.
- Compatibilidad integrada con formatos de datos comunes. CSV, JSON, XML

Diferencias entre PowerShell y PowerShell Core:

PowerShell Core: Nueva versión de PowerShell, un shell de línea de comandos y lenguaje de scripts que se incluye con Microsoft Windows. Está disponible como una aplicación multiplataforma, es decir, las secuencias de comandos que se escriban se ejecutaran en cualquier sistema operativo compatible. PowerShell sólo está disponible para Windows.

	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)

Usos comunes de PowerShell

- **Monitoreo:** Para escribir scripts que pueden ser utilizados por un software de monitoreo.
- **Testear infraestructura:** Las pruebas de infraestructura en Pester son código de PS que ejecuta el módulo Pester PowerShell y se crea de una manera específica, conocida como lenguaje específico de dominio (DSL). Este DSL describe el estado deseado y tiene el código necesario para verificar ese estado y comparar el resultado.
- **Automatización de procesos:** Automatizar tareas en un proceso de liberación de software (CI/CD).
- **Automatización de tareas:** PS sirve para facilitar a los administradores de sistemas las tareas de automatización, administración y configuración de sistema.
- **Configuration management:** Al utilizar PowerShell DSC.

Kit de supervivencia de PowerShell:



Pipeline, variables, estructuras de control, scripts y funciones.

Pipeline: Serie de comandos conectados mediante operadores de canalización ("|", ASCII 124). Cada operador del pipeline envía los resultados del comando anterior al siguiente comando.

Por ejemplo: `Command-1 | Command-2 | Command-3`

Los comandos se procesan en orden de izquierda a derecha. El procesamiento se controla como una operación única y el resultado se muestra a medida que se genera.

Variable: Unidad de memoria en la que se almacenan los valores. En PowerShell, las variables se representan mediante cadenas de texto que comienzan por un signo de dólar (\$), como \$a, \$process o \$my_var. Los nombres no distinguen mayúsculas de minúsculas y pueden incluir espacios y caracteres especiales.

Tipos de variables en PowerShell

- **Variables creadas por el usuario:** el usuario crea y mantiene las variables. Éstas solo existen mientras la ventana de PowerShell está abierta. Cuando se cierra la ventana de PowerShell, se eliminan las variables. Para guardar una variable, deberán agregarla a su perfil de PowerShell.
- **Variables automáticas:** Almacenan el estado de PowerShell. Las crea PowerShell y cambia sus valores según sea necesario para mantener su precisión. Los usuarios no pueden cambiar el valor de estas variables.
- **Variables de preferencia:** Almacenan las preferencias de usuario para PowerShell. Estas variables las crea PowerShell y se rellenan con valores predeterminados. Los usuarios pueden cambiar sus valores.

Estructuras de control: Permiten modificar el flujo de ejecución de las instrucciones de un programa. Con ellas se puede:

- De acuerdo con una condición, ejecutar un grupo u otro de sentencias (If-Then-Else).
- De acuerdo con el valor de una variable, ejecutar un grupo u otro de sentencias (Switch-Case).
- Ejecutar un grupo de sentencias solo cuando se cumpla una condición (Do-While).
- Ejecutar un grupo de sentencias hasta que se cumpla una condición (Do-Until).
- Ejecutar un grupo de sentencias un número determinado de veces (For-Next).

Se pueden clasificar en:

- Secuenciales.
- Iterativas.
- Control avanzadas.

Script: Es un archivo de texto sin formato que contiene uno o más comandos de PS. Los scripts de PS tienen una extensión de archivo .ps1. Lo más importante es que permite ejecutar los comandos simplemente escribiendo la ruta de acceso del script y el nombre de archivo. Los scripts pueden ser tan simples como un solo comando en un archivo o tan complejos como un programa completo.

Módulos de PowerShell:

Módulo: Paquete que contiene objetos de PS, como cmdlets, proveedores, funciones, flujos de trabajo, variables y alias. Los objetos y 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.

Get-Command obtiene todos los comandos en todos los módulos instalados, incluso si aún no están en la sesión. Se puede encontrar un comando y usarlo sin tener la necesidad de importar el módulo primero. Los comandos que incluyen un carácter comodín (*) se consideran para descubrimiento, no para uso y no importan ningún módulo. Los módulos en otras ubicaciones deben importarse ejecutando el Import-Module cmdlet.

¿Cómo utilizar e instalar un módulo de PowerShell?: La mayoría de los módulos se instalan automáticamente. PowerShell viene con varios módulos preinstalados, a veces llamados módulos centrales. En equipos con Windows, si las funciones que se incluyen con el sistema operativo tienen cmdlets para administrarlos, esos módulos están preinstalados.

Install-Module: obtiene uno o más módulos que cumplen los criterios especificados de un repositorio en línea. Este verifica que los resultados de la búsqueda sean módulos válidos y copia las carpetas del módulo en la ubicación de instalación.

Si el módulo que se está instalando tiene el mismo nombre o versión —o contiene comandos en un módulo existente—, se muestran mensajes de advertencia.

¿Cómo encontrar módulos instalados?: Para encontrar módulos que están instalados en una ubicación de módulo predeterminada, pero que aún no se han importado a tu sesión, debes escribir:

```
Get-Module -ListAvailable
```

Para encontrar los módulos que ya se han importado a tu sesión, escríblos en tu sesión de PS:

```
Get-Module
```

Find-Module: busca módulos en un repositorio que coinciden con los criterios especificados. Devuelve un objeto PSRepositoryItemInfo para cada módulo que encuentra. Los objetos se pueden enviar por el pipeline a cmdlets como Install-Module.

¿Cómo buscar los comandos en un módulo?: Usamos el comando Get-Command cmdlet para buscar todos los comandos disponibles. Se pueden usar los parámetros del Get-Command cmdlet para filtrar comandos por módulo, nombre y sustantivo.

```
Get-Command -Module <module-name>
```

¿Cómo obtener ayuda para los comandos de un módulo?: Si el módulo contiene archivos de ayuda para los comandos que exporta, el Get-Help cmdlet mostrará los temas de ayuda. Usar el mismo Get-Help formato de comando que usarías para obtener ayuda para cualquier comando de PS. Para obtener ayuda para un comando de un módulo y obtener ayuda en línea para el comando en un módulo:

```
GetHelp <command-name>
```

Para descargar e instalar los archivos de ayuda de los comandos de un módulo:

```
Update-Help -Module <module-name>
```

¿Cómo importar y quitar un módulo?: La importación es necesaria cuando un módulo no está instalado en las ubicaciones especificadas por la variable de entorno PSModulePath, \$env:PSModulePath; o el módulo consta de un archivo, como un archivo .dll o .psm1, en lugar de un módulo típico que se entrega como una carpeta.

Para importar módulos, puedes usar el Import-Module cmdlet. Para importar módulos en una ubicación PSModulePath en la sesión actual, utiliza el siguiente formato de comando:

```
Import-Module <module-name>
```

Al quitar un módulo, se eliminan de la sesión los comandos que el módulo agregó. Para hacerlo tendrás que utilizar el siguiente formato de comando:

```
Remove-Module <module-name>
```

C8A – PYTHON

Introducción a Python

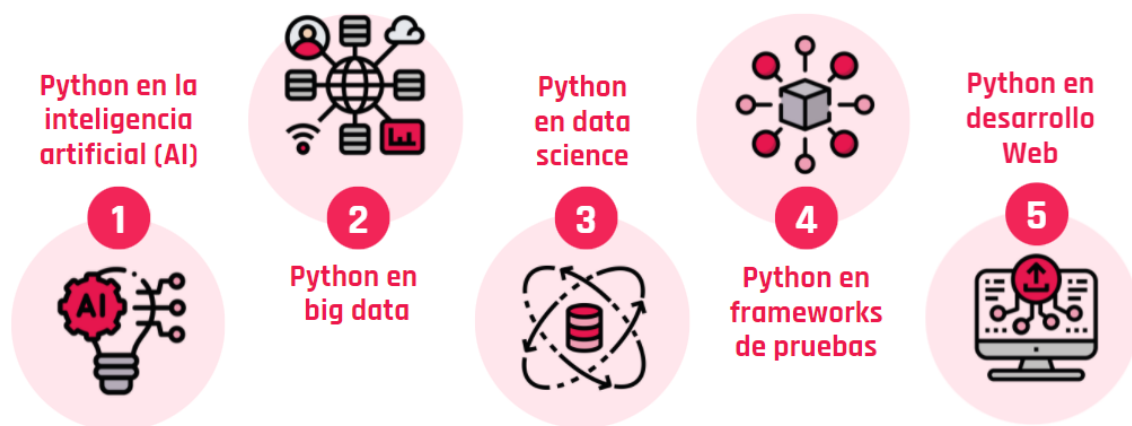
Python: “Lenguaje de programación versátil, multiplataforma y multiparadigma que se destaca por su código legible y limpio”. Es un lenguaje interpretado de escritura rápida, escalable, robusto y de código abierto. Es multiparadigma, lo que permite que sea flexible y fácil de aprender de manera independiente. Es apto para todas las plataformas, solo hay que utilizar el intérprete correcto (ya que es un lenguaje interpretado).

¿Qué es y para qué sirve Python?: Su principal objetivo es la automatización de procesos, lo que hará de las tareas algo mucho más simple.

Usos de Python: Uno de sus principales aliados es la inteligencia artificial, pero también se destaca en aplicaciones web y big data. También se utiliza para obtener información de otros sitios web, lo que comúnmente se denomina “scraping”. Otras áreas como astronomía, robótica, vehículos autónomos, negocios, meteorología y desarrollo de interfaces gráficas de usuario también se benefician con el uso de Python.

¿Por qué es importante saber Python en infraestructura?

Principales industrias que usan este lenguaje de programación:



Variables en Python: Son "etiquetas" que permiten hacer referencia a los datos (que se guardan en unas "cajas" llamadas objetos). Python es un lenguaje de programación orientado a objetos y su modelo de datos también está basado en objetos. Para cada dato que aparece en un programa, Python crea un objeto que lo contiene.

Cada objeto tiene:

- Un **identificador único** (un número entero, distinto para cada objeto). El identificador permite a Python referirse al objeto sin ambigüedades.
- Un **tipo de dato** (entero, decimal, cadena de caracteres, etc.). El tipo de datos permite saber a Python qué operaciones pueden hacerse con el dato.
- Un **valor** (el propio dato).

C10A – CONFIGURATION MANGEMENT

Configuration Management

Configuration management y change management son dos de los procesos fundamentales del conjunto de metodologías conocido como ITIL.

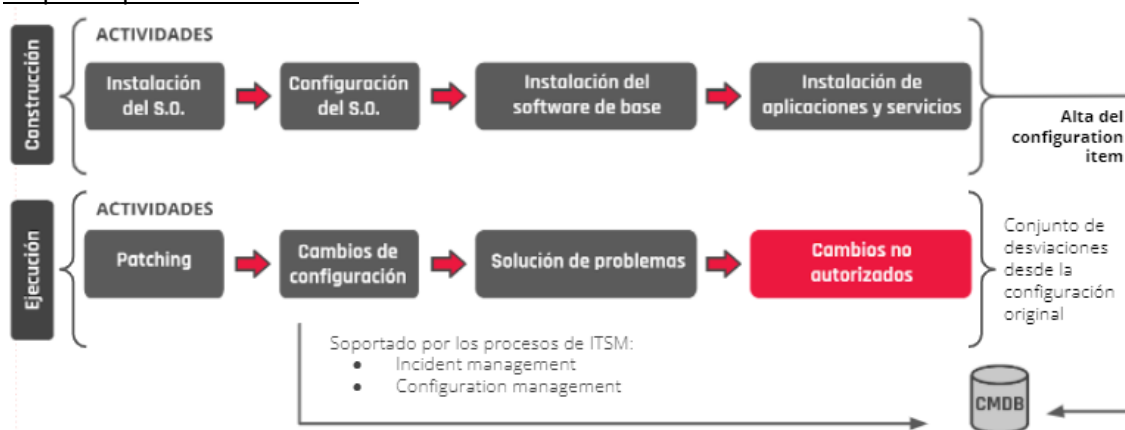
ITIL describe la gestión de cambios como el proceso de controlar y gestionar un cambio a lo largo de todo su ciclo de vida con el objetivo de minimizar el riesgo.

¿Qué es un cambio?: Modificación o eliminación de cualquier cosa que pueda afectar directa o indirectamente los servicios. Ejemplos de un cambio: reemplazo de hardware, instalación de software en un servidor o la modificación de una configuración de un sistema que alterará su comportamiento.

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 conocen como configuration items (CI) y se almacenan en lo que es llamado CMDB (configuration management database). (Es una documentación de los cambios realizados).

Change management + Configuration management: Ambos procesos se complementan, ya que change management aporta a 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.

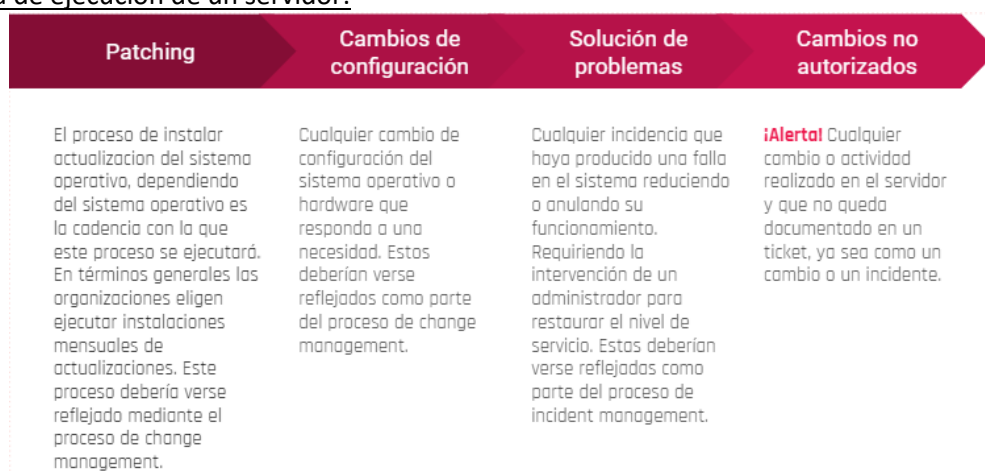
Mapa del proceso tradicional:



Etapas de construcción de un servidor:



Etapas de ejecución de un servidor:



Proceso de configuración management moderno: Todo comienza con un requerimiento, que puede ser configurar un aspecto específico de un sistema operativo como un antivirus hasta instalar un servidor web o un servidor de base de datos. El administrador escribe un manifiesto de dicha configuración que luego será aplicado al servidor correspondiente.

Se puede decir que los procesos tradicionales son imperativos, mientras que los procesos modernos son declarativos ya que manifestamos la configuración que será aplicada por un sistema de configuration management en el servidor. Luego de que la configuración esta codificada, pueden suceder dos cosas:

1. El administrador del sistema puede aplicar la configuración manualmente en el servidor.
2. El administrador del sistema opta por guardar los cambios en un sistema de control de versiones. Este sistema de control de versiones, será la “fuente de la verdad” para que el sistema de configuration management observe los cambios y aplique la configuración en los servidores.

Este último enfoque suele ser parte de las prácticas que se utilizan en el modelo GitOps. Habiendo enviado el código al repositorio git comienza el proceso automático. Un sistema externo tomara la configuración de git para luego plasmarla en los servidores adecuados. Por lo general, quien se ocupa de conectar el repositorio git con el sistema de configuration management es el pipeline. Éstos, son utilizados mayormente en los procesos de despliegue de software y tiene dos instancias, la etapa de build y la etapa de release. Build tiende a utilizarse para testear y compilar la aplicación. En release, se envía al sistema el resultado del proceso de build.

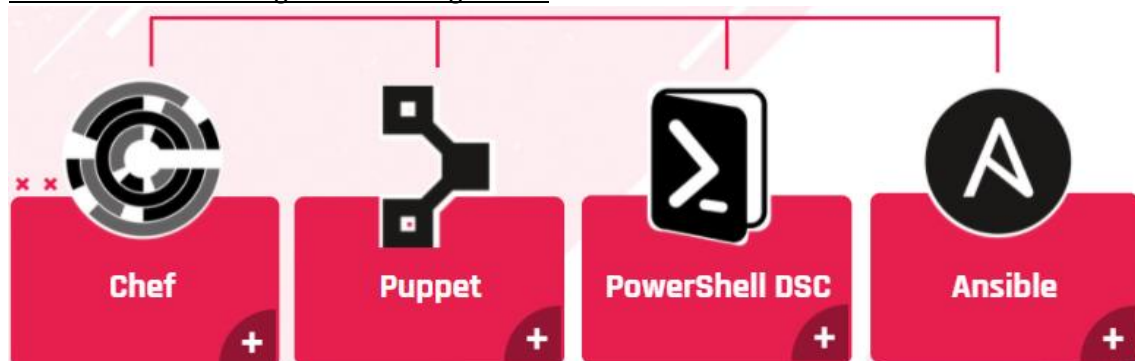
¿Cómo hacemos para que los servidores sepan que hay una nueva configuración?: Los servidores que forman parte de nuestro ambiente y que están gobernados por el sistema de configuration management tienen instalados un agente que cumple con dos funciones:

1. Registra el servidor contra el sistema de configuration management, permitiendo que el sistema conozca la existencia del servidor.
2. Opera de representante para el sistema de configuration management en el contexto del servidor a configurar.

Durante el proceso de registro, los servidores envían información sobre ellos mismos al sistema de configuration management, produciendo un “inventario vivo”, o sea, que cualquier cambio que se produzca en el servidor se puede ver reflejado en el sistema de forma casi inmediata. Luego, el administrador de sistemas asigna una configuración a un servidor dependiendo de distintos parámetros, por ejemplo el nombre del servidor, el rol y el segmento de la red.

Por último, si la configuración especificara que se debe instalar software en el servidor, el mismo debe estar disponible en una ubicación que sea accesible para el servidor.

Herramientas de configuration management:



Chef: Plataforma de configuration management con una arquitectura cliente-servidor, principalmente, orientada a Linux. Lo que significa que 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: Plataforma de configuration management con una arquitectura cliente-servidor, principalmente, orientada a Linux. Lo que significa que 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. Como en Chef, 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 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 ha habido desviaciones respecto de la configuración definida y corregirlas.
Las configuraciones para DSC son escritas en el lenguaje de scripting PowerShell.

Ansible: Es una tecnología de configuration management 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).

C11A – CONFIGURATION MANAGEMENT – ANSIBLE

Conociendo Ansible

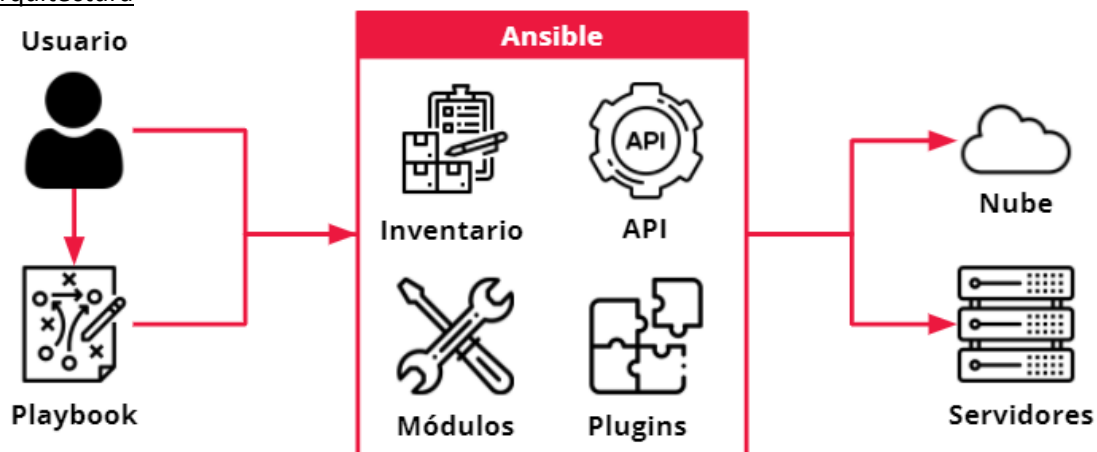
¿Qué es Ansible y para qué sirve? Se trata de un software de gestión de la configuración automática y remota, que nos permite centralizar la configuración de numerosos servidores, dispositivos de red y cloud providers de una forma sencilla y automatizada.

Usa SSH para conectarse a los servidores y ejecutar las tareas de configuración. Ansible nos permite controlar y configurar nodos desde un servidor central.

¿Por qué elegir Ansible?

- *No usa agentes:* Mientras que al equipo que queramos configurar se pueda acceder vía SSH y correr Python, se podrá configurar usando Ansible.
- *Idempotente:* Toda la arquitectura de Ansible está estructurada alrededor del concepto de idempotencia. La idea es que solo se harán configuraciones si son necesarias y que se podrán aplicar de manera repetible sin provocar efectos secundarios.
- *Declarativo:* A diferencia de un script, en donde debemos escribir la lógica necesaria para efectuar una configuración, Ansible trabaja por nosotros, dejándonos escribir una descripción del estado que deseamos para un servidor o conjunto de servidores. Es luego Ansible el que se encarga de aplicar dicha descripción de forma idempotente.
- *Fácil de aprender*

Arquitectura



Inventario: Lista de los nodos que pueden ser accedidos por Ansible. Por defecto, está soportado por un archivo de configuración, cuya ubicación es `/etc/ansible/hosts`. Los nodos pueden estar listados por nombre o IP.

Cada nodo es asignado a un grupo, como pueden ser “web servers”, “db servers”, entre otros. El inventario debe estar escrito en uno de muchos formatos, estos pueden ser YAML, INI, etcétera. YAML es el formato más utilizado en la industria.

Playbooks: Archivos también escritos en formato YAML. Son la descripción del estado deseado de los sistemas que vamos a configurar. Ansible es el que 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. Los playbooks 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 por lo que son muy sencillos de evolucionar y editar, contienen Plays (jugadas), las jugadas (contienen tareas (en inglés, tasks), las tareas invocan módulos.

Ejemplo de un playbook: Es la versión más reciente de Apache y se asegura de que este corriendo en aquellos servidores que estén bajo el grupo “webservers”.

```
- 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: Hay más de 1000 módulos incluidos con Ansible para automatizar las diferentes partes de nuestro ambiente. Se puede pensar en los módulos 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 lo puede escribir en cualquiera de los lenguajes de scripting standard de mercado. Uno de los principios de diseño de los módulos es la idempotencia. Dentro de los módulos más populares podemos encontrar: *Service, file, copy, iptables*, entre otros.

Ejemplo de la invocación de un módulo: El primer nos ayuda a asegurarnos que el servicio de Apache está corriendo. El segundo invoca el módulo ‘ping’ para hacer un ‘ping’ localmente contra ‘localhost’:

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

La herencia de Python: Como Ansible está desarrollado en Python, hay algunas cuestiones que es importante conocer:

- El lenguaje de templating (Jinja2)
- Operador ternario: El operador ternario de Python 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 errores que encontremos van a estar en un formato que, de estar familiarizados con Python, nos resultará más sencillo de leer.

Casos de uso:

- *Aprovisionamiento:* Utilizar Ansible para instanciar servidores o máquinas virtuales “configurando el sistemas 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: Ansible es la herramienta más simple para implementar una estrategia de configuration management. Está diseñado para ser minimalista, consistente, seguro y altamente confiable. Cualquier desarrollador, tester o administrador de infraestructura puede fácilmente utilizarlo para configurar un conjunto de nodos. Además, cualquier persona en el departamento de IT podría escribir un playbook sin mayores dificultades.

Las configuraciones descritas en Ansible (playbooks) son sencillamente una descripción de la infraestructura de modo que cualquier persona en el área de IT podría entender el significado de cada tarea en un playbook.

Ansible solo requiere el password o la clave privada del usuario que se utilizará para acceder desde Ansible a los sistemas que se configuraron.

¿Sabías que?



MÓDULO 3 – CONTAINERS

C13A – DOCKER EN PROFUNDIDAD

Docker en profundidad

Introducción: Los contenedores son la vía que nos coloca Docker para tener el mismo uso que con las máquinas virtuales creadas de la forma tradicional. Docker utiliza estos contenedores para aislar uno o más procesos. Estos procesos en el host necesitan memoria, CPU, acceso a la red y espacio en disco.

Es un ambiente perfecto para que las aplicaciones funcionen de forma correcta.

Docker: Es un proyecto de código abierto que automatiza el despliegue dentro de los contenedores. De esta manera, proporciona una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos. Básicamente es una forma simplificada de empaquetar y ejecutar aplicaciones. Teniendo la facilidad de construir, ejecutar, detener, iniciar, investigar, modificar y manipular contenedores. Implementa una API de alto nivel para proporcionar contenedores que ejecutan procesos de manera aislada.

Mediante el uso de contenedores, los recursos pueden ser aislados, los servicios restringidos y los procesos tienen una visión casi privada del S.O.

¿Porque Docker?: Podemos contener cualquier aplicación en cualquier lenguaje con cualquier S.O. Simplifica la creación de tareas de carga y el funcionamiento de las mismas.

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, también conocido como Docker daemon:

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 aplicación.



Docker Engine



Docker Hub: Es un registro en línea de imágenes de Docker.

Docker Hub

Docker Trusted Registry: Es un registro privado en el sitio para imágenes de Docker.



Docker Trusted Registry



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 Client

Docker Images: Es una plantilla de solo lectura utilizada para crear contenedores. Contiene un conjunto de instrucciones para crear los contenedores.



Docker Images



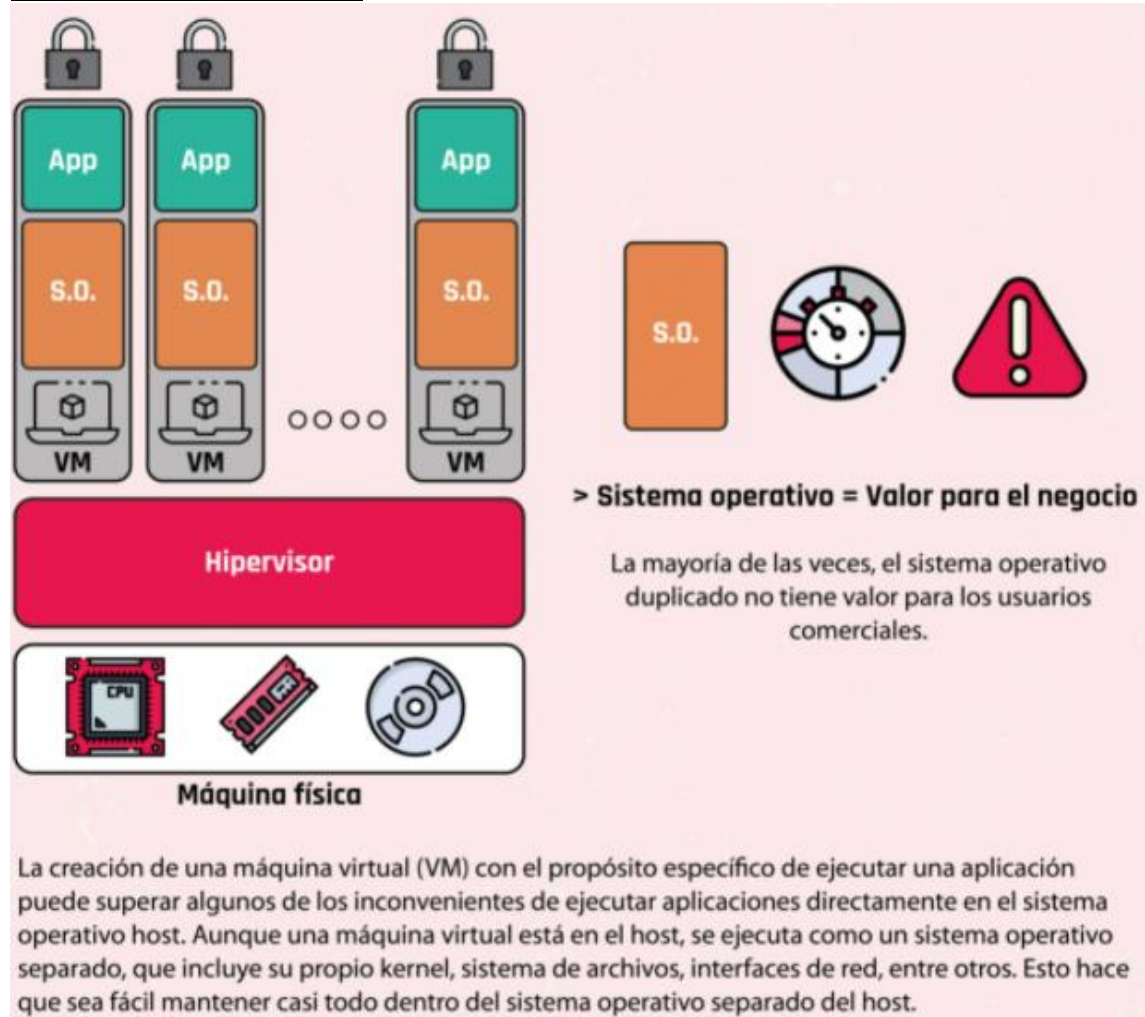
Docker Containers: Es una plataforma de aplicación aislada basada en una o más imágenes que contiene todo lo necesario para ejecutar una aplicación.

Docker Containers

Vocabulario de Docker:

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

Desafíos con la virtualización:



Máquina virtual vs contenedores:

Máquina Virtual



Sistema operativo duplicado



Contenedores



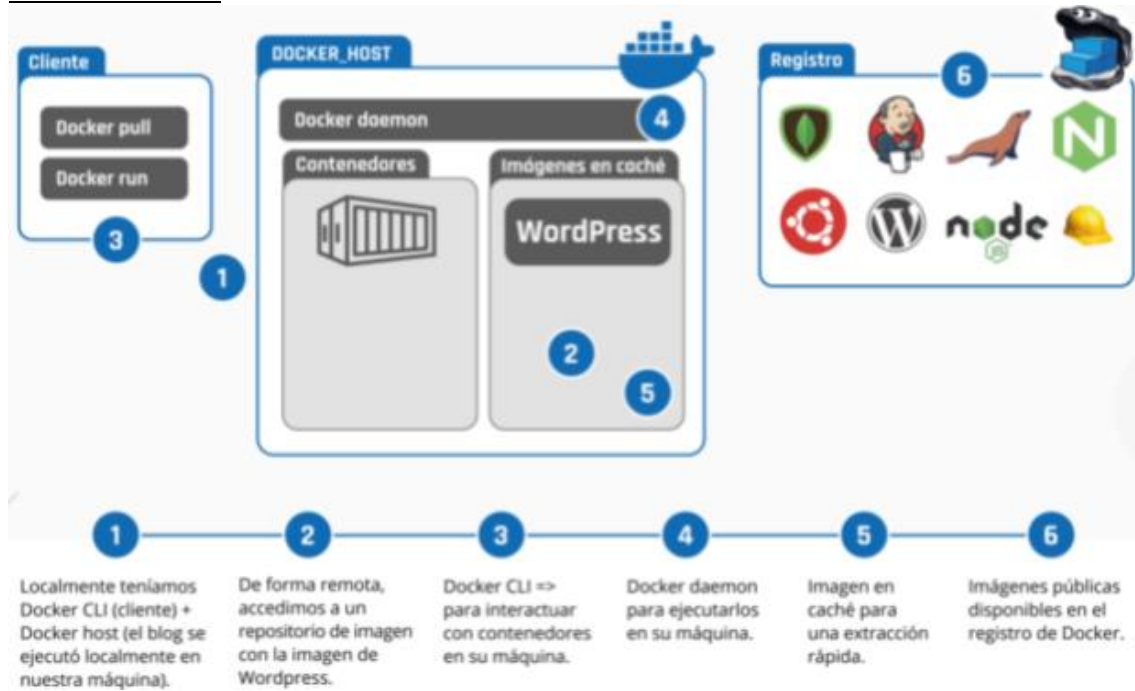
Arquitectura del contenedor



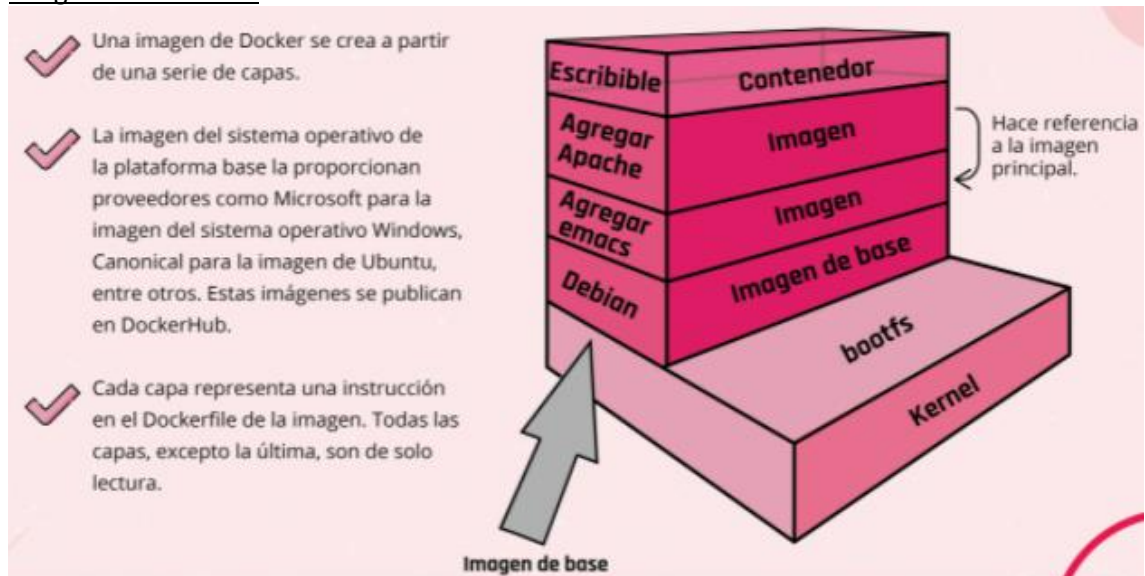
Sistema operativo duplicado: Las máquinas virtuales deberán contener el sistema operativo y todos los componentes de soporte.

Arquitectura del contenedor: 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. 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.

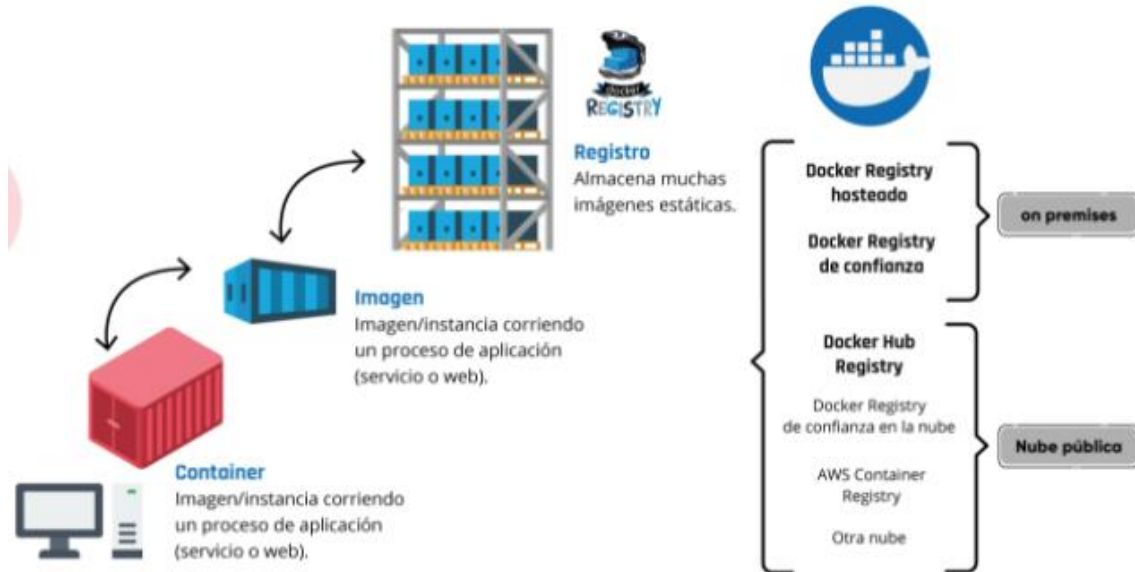
Docker en acción:



Imágenes de Docker:



Taxonomía básica en Docker:



Dockerfile: Archivo de texto simple con un conjunto de comandos o instrucciones. Éstos se ejecutan sucesivamente para realizar acciones en la imagen base para crear una nueva imagen de la ventana acoplable.



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: O tags sirven para 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).

C14A – EL ECOSISTEMA DE DOCKER Y MEJORES PRÁCTICAS

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

¿Para qué sirve Docker Hub?: Nos permite extraer y enviar imágenes de la ventana acoplable hacia y desde Docker Hub. Podemos tratar esto como un GitHub, donde obtenemos y enviamos nuestro código fuente, pero en el caso de Docker Hub descargamos o publicamos nuestras imágenes de contenedor.

Es un repositorio en línea basado en la nube que almacena ambos tipos de repositorios públicos y privados. Los repositorios públicos son accesibles para todos, pero el privado es accesible para el propietario interesado de los repositorios. También hay un 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 de Docker Hub.
2. *Equipo y organizaciones*: Permite crear grupos de trabajo e impulsar los repositorios privados, que están disponibles para su uso únicamente dentro de nuestra organización. De esta forma, hemos gestionado el acceso a nuestros repositorios privados de imágenes de contenedores.
3. *Integración de GitHub y Bitbucket*: Permite la integración con repositorios de código fuente como GitHub y Bitbucket.
4. *Construcciones automatizadas*: Si se ha enviado algún cambio en el código fuente a los repositorios, 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 para 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:

1. Iniciar sesión en Docker Hub. Si no tienes una cuenta, puedes crearla desde la página web.
2. Hacer clic en “Crear repositorio” en la página de bienvenida. Luego, te pedirá un nombre y le dará un nombre a tu repositorio.
3. Seleccionar una visibilidad pública o privada. También se puede integrar tus repositorios de código fuente como GitHub y Bitbucket a través de la configuración de compilación, pero es opcional.
4. Hacer clic en “Crear”.
5. Con la herramienta terminal de Docker Desktop, descargada e instalada, podremos iniciar sesión en Docker Hub mediante un comando:

```
docker login
```

Explorar las imágenes: Hay dos formas de buscar imágenes y repositorios públicos desde Docker Hub.

1. Buscarlo en el sitio web de Docker Hub
2. Usar la herramienta de línea de comandos y ejecutar el siguiente comando, suponiendo que queremos buscar en la imagen del repositorio de MySQL.

```
docker search mysql
```

Descargar una imagen: Podemos descargar una imagen de Docker Hub 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. Una cosa a tener en cuenta aquí es que si notamos la salida del comando de búsqueda de la ventana acoplable, hay muchas imágenes de MySQL en Docker Hub, y eso se debe a que cualquiera puede enviar una imagen. Depende de nosotros saber cuál usar en función de nuestro caso de uso y asegurarnos de que sea el apropiado. Por ejemplo, si queremos extraer una imagen bitnami/mysql:

```
# docker pull bitnami/mysql
```

Crear una imagen: Este proceso requiere un Dockerfile, un archivo de configuración que sigue ensamblando instrucciones.

¿Cómo funciona?: 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. Cada capa de una imagen de Docker representa las instrucciones de un Dockerfile.

A continuación, sigamos los pasos para crear una imagen usando un Dockerfile que especifica la configuración de nuestra aplicación.

```
# sudo vim Dockerfile
FROM ubuntu:16.04
ENV DEBIAN_FRONTEND noninteractive
MAINTAINER someuser@somedomain.com
RUN apt-get update
RUN apt-get install mysql-server -y
CMD echo "My first image created."
```

- *FROM (de):* define la imagen base que se utilizará.
- *MAINTAINER (mantenedores):* persona que va a mantener esa imagen.
- *RUN (correr):* se utiliza para ejecutar la instrucción dada para la imagen. En nuestro caso, primero actualiza el sistema y luego instala MySQL.
- *CMD:* se utiliza para ejecutar un comando una vez que se ha lanzado el contenedor.
- *COPY (copiar):* se utiliza para copiar un archivo de nuestro sistema operativo host al contenedor de la ventana acoplable.
- *EXPOSE (exponer):* se utiliza para especificar el número de puerto en el que el contenedor ejecutará su proceso.

Ya creado el Dockerfile debemos ejecutar docker build para “armar” nuestra imagen localmente, para luego enviarla a Docker Hub. Este comando debemos ejecutarlo dentro de la carpeta donde se encuentra el Dockerfile.

```
docker build ./ -t asadali08527/first-repo
```

Podemos verificar que la imagen está creada con la siguiente línea de código:

```
docker image ls
```

Empujar una imagen: Una vez que nuestra imagen se ha creado correctamente y se está ejecutando, podemos enviarla a Docker Hub mediante el comando push.

```
docker push asadali08527/first-repo
```

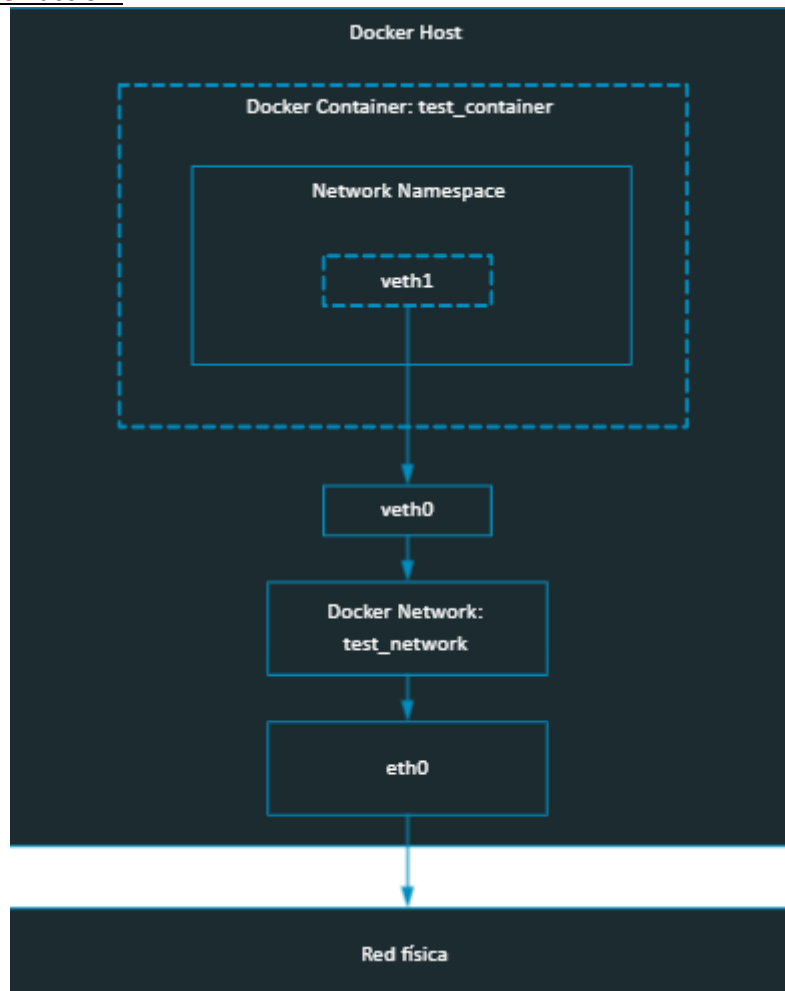
¿Qué son las imágenes certificadas por Docker?: Son las imágenes oficiales impulsadas por proveedores o contribuyentes. Una imagen solo puede ser certificada por Docker Hub si su contenido cumple con las reglas, estándares y leyes proporcionadas por Docker Hub. Docker Hub proporciona inspectDockerImage, herramienta a través de la cual un proveedor puede auto-certificar las imágenes y los complementos (por lo general, el proveedor o contribuyente publica sus complementos para registrar volúmenes y redes).

Docker Networking:

Introducción a las redes de Docker: Nos enfocaremos en el driver llamado bridge.

- Las redes de tipo “bridge” son locales y exclusivas del host en donde fueron creadas.
- Son el tipo de red por defecto en Docker.
- Simula la creación de switches o hubs, de nivel 2 (en el modelo OSI). Podemos utilizar herramientas como ‘brctl’ en Linux para ver el funcionamiento interno.
- Luego de la instalación de Docker se crea por defecto una red de tipo bridge llamada “bridge”, las buenas prácticas indican que esta no debe ser utilizada y en su lugar se deben crear nuevas redes para usos específicos.

Bridge Driver en acción:



Resolución de nombres: Las IPs son efímeras, por lo que para todas las redes que creemos y para todos los containers a los que se le asigne un nombre de manera deliberada —usando `--name` en `docker create`— conectados a estas, 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.

docker network

create: nos permite crear una red.

--internal: agrando esta opción la red creada será del tipo 'privada'.

- *connect:* nos permite conectar un container existente a una red.

docker run

--network <networkName>: nombre de la red a la cual conectaremos el 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.

Documentación de Docker:

- https://docs.docker.com/engine/reference/commandline/network_create/#specify-advanced-options
- <https://docs.docker.com/network/>
- <https://docs.docker.com/network/network-tutorial-standalone/>

Docker Compose

Docker Compose: Es una herramienta que permite simplificar el uso de Docker. A partir de archivos YAML es más sencillo crear contenedores, conectarlos, habilitar puertos, volúmenes, etc. Con Compose podemos crear diferentes contenedores y al mismo tiempo, en cada contenedor, diferentes servicios, unirlos a un volumen común, iniciarlos y apagarlos, etc. Es un componente fundamental para poder construir aplicaciones y microservicios.

¿Por qué utilizar Docker con Docker Compose?:

Beneficios:

- No necesitamos instalar ni mantener software adicional en nuestro equipo.
- Podemos tener todo nuestro entorno de desarrollo en un único repositorio.
- Levantar todo el entorno de desarrollo se limita a un solo comando `docker-compose up`.

Pero no todo es color de rosa, también tenemos una **desventaja** que está enfocada al rendimiento, ya que a pesar de que los contenedores están enfocados a ser eficientes, siguen consumiendo recursos de la máquina anfitriona tales como procesador y memoria por lo que si la cantidad de contenedores que están corriendo al mismo tiempo es grande o si los contenedores son pesados al momento de crearse y levantarse, podrían llevar a cuelgues del sistema y cosas similares.

Conclusión: Podemos probar con distintos entornos y lenguajes, la idea es conseguir optimizar al máximo nuestro entorno de desarrollo y enfocarnos en el principal objetivo que es escribir código de calidad.

x x
x x
x x

Docker Swarm



Arquitectura General

-  Contenedores de aplicaciones
-  Monitoreo: agente de Docker de sematext
-  Swarm Master / Agente Swarm



MÓDULO 4 – CLOUD COMPUTING

C16A – CLOUD COMPUTING, UNA MIRADA HOLÍSTICA E INTEGRADORA

¿Qué es el serverless?

Introducción: Informática sin servidor. Serverless es un modelo de desarrollo nativo de la nube que permite que los desarrolladores diseñen y ejecuten aplicaciones sin tener que gestionar servidores, que si bien se utilizan, están aislados de la etapa de desarrollo de las aplicaciones. El proveedor de la nube se encarga de las tareas de preparación, mantenimiento y adaptación de infraestructura de servidores. Una vez que se instalan las aplicaciones, sin el servidor, responden a las demandas y se amplían o estancan automáticamente en función de las necesidades. El costo se mide a través de un modelo de ejecución basado en eventos, por eso si una función permanece inactiva no se genera ningún costo.

¿Qué función desempeña el proveedor de nube con serverless?: Ejecutan los servicios físicos y asignan los recursos de forma dinámica para los usuarios que implementen el código directamente en la producción.

Se ocupan de:

- Gestionar el S.O y de archivos.
- Ejecutar los parches de seguridad.
- Equilibrar la carga.
- Administrar la capacidad.
- Adaptar los recursos.
- Llevar los registros.
- Supervisar el sistema.

Las ofertas de serverless se clasifican en dos grupos, back end como servicio BaaS y función como servicio FaaS.

-BaaS: Otorga a quienes desarrollan acceso a aplicaciones, servicios externos. Por ejemplo, un proveedor de nube puede ofrecer servicios de autenticación, cifrado adicional o bases de datos a las que se puede acceder desde la nube.

Las funciones sin servidor, se suelen solicitar mediante llamadas a las interfaces de programación de aplicaciones: APIs.

-FaaS: Se escribe la lógica personalizada del lado del servidor pero se ejecuta en los contenedores que el proveedor de servicios de nube gestiona por completo.

La arquitectura serverless es ideal para aplicaciones asincrónicas y sin estado que pueden iniciarse de forma instantánea. También es bueno para los casos en los que la demanda aumenta de manera inusual e impredecible. Son ideales para los casos de uso que involucran flujos de datos entrantes, chatbots, tareas programadas o lógica empresarial, aplicaciones web y API de back end, automatización de servicios empresariales, sitios web sin servidor y la integración en varios sistemas.

- Reduce el costo operativo y aumenta la productividad del desarrollador.
- Facilita la adopción de un enfoque de DevOps.
- Permite incorporar elementos completos de ofertas de Baas.
- Se reducen los costos operativos.



¿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 sistema operativo conectado a al menos una nube, ya sea pública o privada.

Funciones de las nubes híbridas: Todas deben poder:

- Conectar varias computadoras a través de una red.
- Consolidar los recursos de IT.
- Escalar horizontalmente e implementar los recursos nuevos con rapidez.
- 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: 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.

Si se utiliza el mismo sistema operativo, se extraen todos los requisitos del sistema de hardware, mientras que la plataforma de organización extrae todos los de las aplicaciones. Esto genera un entorno informático uniforme e interconectado en el que las aplicaciones pueden trasladarse de un entorno a otro sin tener que mantener un mapa complejo de las API que falle cada vez que se actualicen las aplicaciones o que cambie de proveedores de nube. Esta interconectividad permite que los equipos de desarrollo y operaciones trabajen juntos en un modelo de DevOps, que es un proceso con el cual los equipos trabajan en conjunto en entornos integrados utilizando una arquitectura de microservicios compatible con los contenedores.

¿Cómo conecto mi centro de cómputos a la nube?

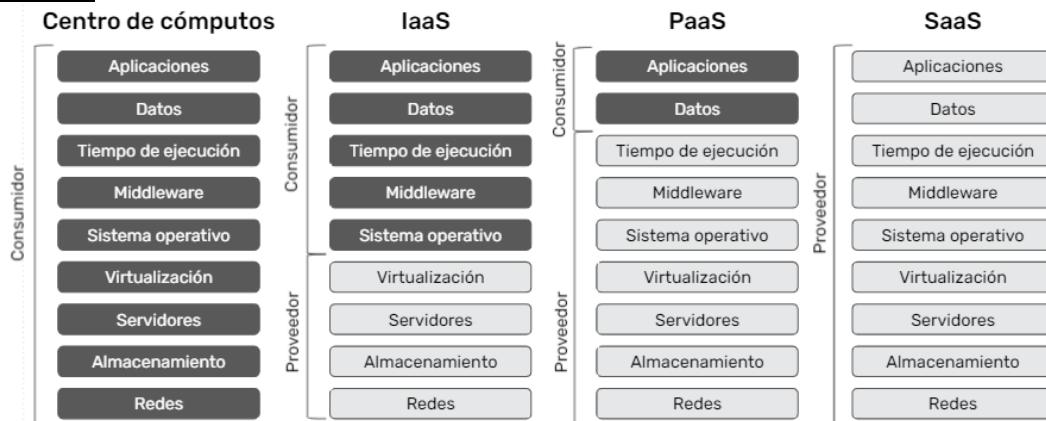


C17A – COMPUTACIÓN EN LA NUBE

Modelo de responsabilidades

¿Por qué es necesario?: En cloud computing lo que hacemos es utilizar recursos de un tercero. El modelo de responsabilidades 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.

El modelo:



Si miramos el modelo podemos identificar:

- Columnas: centro de cómputos, IaaS, PaaS y SaaS.
- Roles: consumidor y proveedor.
- Cajas que definen un conjunto de tareas, tecnologías y disciplinas con las que ya estamos familiarizados, como pueden ser: redes, virtualización o middleware.
- Colores que tiñen cada una de ellas, dependiendo la columna en la que nos encontremos.

Los roles – consumidor y proveedor: Se utilizan para definir quién es responsable. Ya vimos que en cloud computing lo que hacemos es en realidad contratar recursos para la administración de los mismos por un tercero. Ese tercero, dueño de la nube o de los centros de cómputos en los que decidamos correr nuestras aplicaciones, es a quien vamos a llamar proveedor. Nosotros somos los consumidores, ya que contratamos los servicios del proveedor para ejecutar nuestras aplicaciones en sus centros de cómputo.

Las columnas – centro de cómputos: Cuando hablamos de centro de cómputos lo que hacemos es definir de qué tareas nosotros seríamos responsables si somos quienes alojamos en su totalidad los servicios de tecnologías.

- **Pros:** Tenemos control total sobre los activos, podemos decidir cómo, para qué, dónde y cuándo los utilizamos.
- **Contras:** Tenemos responsabilidad total por la salud, utilización y mantenimiento del parque informático. No podemos escalar de manera flexible.

IaaS – Infrastructure as a Service: El proveedor nos da la posibilidad de instanciar máquinas virtuales en su centro de cómputos sin que nosotros tengamos la necesidad de administrar la infraestructura subyacente.

- **Pros:** Seguimos teniendo control sobre el sistema operativo, un ambiente ideal para instalar aplicaciones legacy. Nos permite escalar la infraestructura de forma más dinámica.
- **Contras:** Seguimos siendo responsables por la administración y salud del sistema operativo y todos los servicios y aplicaciones que se ejecuten encima de este.

Paas – Platform as a Service: El proveedor nos ofrece una tecnología específica, expuesta por medio de interfaces definidas, ya sean gráficas o APIs, y nos abstrae de toda la gestión de los recursos subyacentes.

- **Pros:** Nos abstraemos completamente de la administración de los recursos de las capas inferiores. Delegamos en el proveedor la gestión, monitoreo y escalabilidad de la tecnología. Permittiéndonos a nosotros enfocarnos plenamente en la entrega de valor.
- **Contras:** Tiende a ser una opción costosa, ya que nos abstrae de todo el músculo requerido para administración y monitoreo de la infraestructura subyacente. 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. La mejor manera de entender este modelo de responsabilidad es pensar en ejemplos: Trello, Salesforce o Gmail son ejemplos de SaaS.

- **Pros:** Con tan solo una tarjeta de crédito podemos acceder a una aplicación lista para utilizar y resolver una necesidad. Y 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. Dependiendo la flexibilidad de software puede que tengamos que adaptar nuestros procesos a lo que el software permite hacer. No hay control sobre las nuevas funcionalidades del sistema y cuándo se liberan.

Máquinas virtuales en la nube:

La virtualización es el proceso de creación de una versión virtual de algún recurso tecnológico, como un sistema operativo, una plataforma de software o un periférico, disco de almacenamiento o cualquier otro equipamiento que exista en la vida real.

¿Cómo puedo crear una máquina virtual en la nube?

Actualmente hay dos grandes sistemas que ofrecen servicios de máquinas virtuales en la nube, operados 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”.

Ofrece plantillas para que el cliente configure y cree en línea las máquinas que desee, cada una basada en un sistema operativo 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 (Infrastructure 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 máquinas virtuales 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 máquina virtual 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 máquinas virtuales de los clientes.
- *Solo pagamos por el uso:* Al contratar una máquina virtual 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 máquinas virtuales en la nube desde un equipo que tenga conexión a internet.
- *Mayor rapidez de implementación:* El costo y tiempo necesario desde que contratan una máquina virtual 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 máquina virtual 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 máquinas virtuales o modificar los parámetros de los servicios que estas requieren.

Desventajas de usar máquinas virtuales ubicadas en la nube:

- *Menor rendimiento:* Entre una máquina virtual y un equipo anfitrión físico hay una capa de programación que influye ligeramente en el rendimiento de los programas y sistemas operativos virtualizados. Este problema es mayor cuando la máquina virtual opera desde la nube porque se agregan más capas entre el hardware real del servidor de la empresa que presta el servicio de alojamiento y virtualización, el sistema operativo del servidor y el equipo usado por el cliente que accede a la máquina virtual desde su casa u oficina.
- *Su tiempo de respuesta o latencia es mayor:* Al depender de internet, como acceso a la máquina virtual, 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 máquina virtual cuando el flujo de internet esté muy saturado o si los servidores de la empresa que maneja la nube tienen mucha carga de trabajo.
- Si vivimos en una zona o país con conexión a internet lenta o irregular, enfrentaremos el riesgo de llegar a quedar desconectado de nuestras máquinas virtuales y los datos alojados en ellas durante los momentos en que el servicio sea más lento o se interrumpa. Esto se resuelve contratando más conexiones con proveedores distintos, pero implica un mayor gasto económico para los particulares o pequeños emprendedores.

- Se corre el riesgo de que los datos del usuario puedan ser vulnerados por un atacante que logre acceso al sistema del proveedor del servicio de virtualización.

Tipos de máquinas virtuales:



Escalabilidad, tolerancia al fallo y disponibilidad en la nube

Escalabilidad: Capacidad de crecer en capacidad en demanda.

Elasticidad: Crecer y reducir su tamaño. 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.

Tolerancia a fallos: Cuando el servicio es capaz de responder a cierto grado de errores sin dejar de ser funcional.

Disponibilidad: 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 – Amazon Elastic Compute Cloud: Servicio web que proporciona capacidad informática segura y de tamaño modificable en la nube. Está diseñado para simplificar el uso de la informática en la nube a escala web para los desarrolladores. **Proporciona un control completo sobre los recursos informáticos** y puede ejecutarse en el entorno informático acreditado de Amazon. Ofrece la plataforma informática más amplia y profunda con elección de procesador, almacenamiento, red, sistema operativo y modelo de compra. Cuenta con las instancias de GPU más poderosas para la capacitación de machine learning y las cargas de trabajo gráficas, así como las instancias de costo por inferencia más bajas de la nube. En AWS se ejecutan más cargas de trabajo de SAP, HPC, machine learning y Windows que en cualquier otra nube.

Azure Virtual Machines: Microsoft Azure brinda soporte para sistemas operativos Windows y Linux, e incluso soporte a Windows Server 2003. Permite configurar, a través de diversas opciones, la memoria RAM asignada y la CPU que va a tener asociada.

Componentes de una máquina virtual de Azure:

- Disco virtual: el disco es el que tendrá, por ejemplo, el sistema operativo instalado. Gracias al disco virtual puedo 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 una o más redes.
- Direcciones IP: mediante las que podré conectarme al equipo virtual. Estas direcciones IP pueden ser privadas y públicas.
- Grupos de seguridad de red: que nos ayudarán a definir desde qué orígenes me puedo conectar y hacia qué destinos puedo 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.

No somos los responsables de mantener los componentes de bajo nivel que permiten su funcionamiento (hardware).

Microsoft Azure permite parametrizar tres elementos fundamentales:

- El nombre de la máquina virtual: una vez elegido, no se podrá cambiar luego de su creación.
- El sistema operativo: lo que ejecutará como core el equipo virtual. No se puede modificar salvo operaciones avanzadas sobre la VM.
- El tamaño: a través de tamaños preconfigurados que me brindan diversas opciones de CPU, memoria, cantidad de discos soportados y calidad de componentes. Se puede cambiar en cualquier momento y a eso se le llama “cambio vertical” en el hardware virtual. A veces, estos cambios requieren que el equipo se reinicie.

Google VM Instances – Google Cloud: Una instancia es una máquina virtual (VM) alojada en la infraestructura de Google. Podemos crear una instancia si utilizamos Google Cloud Console, la herramienta de línea de comandos de gcloud o la API de Compute Engine.

Las instancias de Compute Engine pueden ejecutar las imágenes públicas de Linux y Windows Server que proporciona Google, así como las imágenes personalizadas privadas que podemos crear o importar desde nuestros sistemas existentes. También podemos 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 sus 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 borramos una instancia, se quita del proyecto.

Instancias y opciones de almacenamiento: 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, podemos agregar opciones de almacenamiento adicionales.

Herramientas para gestionar instancias: 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 nuestras instancias, nos conectamos a la instancia con Secure Shell (SSH) para instancias de Linux o Remote Desktop Protocol (RDP) para instancias de Windows Server.

C19A – CLOUD COMPUTING: REDES (VPC + ELB)

Concepto de red de VPC

Definición de VPC – Nube virtual privada: Es una nube privada que se ubica dentro de una nube pública que le permite aprovechar los beneficios de una red virtualizada, a la vez que utiliza los recursos de la nube pública. (Tenemos un entorno aislado dentro de algo público) La virtualización hospedada mantiene sus datos aislados de los de otras empresas, tanto cuando están en tránsito como cuando se encuentran en la red del proveedor de la nube. Esto ayuda a crear un **ambiente más seguro**.

- Una VPC se conecta con redes remotas a través de una conexión de **red privada virtual** (VPN).
- Una VPC es la opción ideal para las empresas que necesitan altos niveles de **seguridad, privacidad y control**, como las organizaciones del sector de la salud y financiero que deben cumplir con muchas regulaciones.
- Una VPC también es la mejor opción para ejecutar **aplicaciones indispensables**.

Generalmente, las empresas administran una VPC a través del panel de control del proveedor de servicio administrado. De esta manera, se puede ver la VPC y hacer los cambios necesarios fácilmente.

¿Por qué usar una nube privada virtual?:

- *Seguridad:* Una VPC permite proteger el ambiente de redes virtual, incluso las direcciones IP, subredes y puertas de enlace de red.
- *Control de datos:* La VPC está aislada y no tiene contacto con otras nubes en las capas de la red, por lo que podemos controlar los datos y evitar que se mezclen con los datos de otras empresas, uno de los riesgos potenciales que conllevan las nubes públicas.
- *Rendimiento:* Se puede priorizar el tráfico de red de ciertas aplicaciones para optimizar su rendimiento, lo que ayuda a eliminar la congestión y los posibles embotellamientos.
- *Flexibilidad bajo demanda:* Se puede diseñar la arquitectura de nube que mejor se adapte a las necesidades de la empresa.

Concepto de elasticidad vs escalabilidad

Escalabilidad: Capacidad de los recursos para aumentar o disminuir en tamaño o cantidad. Hay mucha infraestructura involucrada para hacer que algo así suceda, por lo que no es una tarea fácil. Justamente, muchos de los servicios en AWS son escalables de manera predeterminada. Esta es una de las razones por las que AWS es tan exitoso. La escalabilidad es bastante simple de definir, por lo que a menudo se le atribuyen algunos de los aspectos de la elasticidad.

(Aumenta o disminuye en forma vertical, la disminución NO es de forma automática)

Elasticidad: Capacidad de incrementar o disminuir la infraestructura y recursos de los que se dispone en la nube según las necesidades de la empresa. Es decir, que es la capacidad de los recursos para escalar en respuesta a los criterios establecidos. Esto es lo que sucede cuando un equilibrador de carga agrega instancias cada vez que una aplicación web recibe mucho tráfico. No todos los servicios de AWS admiten elasticidad, e incluso aquellos que sí a menudo necesitan configurarse de cierta manera. (Aumenta o disminuye en forma horizontal (más VM por ej, la disminución SI es de forma automática)

¡Escalabilidad es necesaria para la elasticidad, pero no al revés!

Escalabilidad	Elasticidad
Capacidad de un sistema para aumentar la carga de trabajo en sus recursos de hardware actuales (escalar).	Capacidad de un sistema para aumentar la carga de trabajo en sus recursos de hardware actuales y adicionales – agregados dinámicamente bajo demanda – (escalar).
Aumento de la capacidad para cumplir con la carga de trabajo creciente .	Aumento o reducción de la capacidad para cumplir con la carga de trabajo.
En un entorno de escala, los recursos disponibles pueden exceder para satisfacer las demandas futuras .	En el entorno elástico, los recursos disponibles coinciden con las demandas actuales lo más cerca posible.
La escalabilidad se adapta solo al aumento de la carga de trabajo al aprovisionar los recursos de manera incremental .	La elasticidad se adapta tanto al aumento de la carga de trabajo como a la disminución , al aprovisionar y desaprovechar recursos de manera autónoma.
El aumento de la carga de trabajo sirve para aumentar el poder de un solo recurso informático o para aumentar el poder de un grupo de recursos informáticos.	La carga de trabajo variable se sirve con variaciones dinámicas en el uso de los recursos de la computadora.
La escalabilidad permite a una empresa satisfacer las demandas esperadas de servicios con necesidades	Elasticidad permite a una empresa satisfacer cambios inesperados en la demanda de servicios con
estratégicas a largo plazo.	necesidades tácticas a corto plazo.
Está aumentando la capacidad de servir a un entorno donde la carga de trabajo está aumentando.	Es la capacidad de aumentar o disminuir la capacidad de servir a voluntad.

Elasticidad es la capacidad de un sistema para aumentar (o disminuir) su capacidad de cómputo, almacenamiento, trabajo neto, etc.

Ejemplo. Se puede configurar un sistema para aumentar el espacio total en disco de su clúster back end en un orden de 2 si se utiliza más del 80% del almacenamiento total disponible actualmente. Si por alguna razón, en un momento posterior, los datos se eliminan del almacenamiento y, por ejemplo, el almacenamiento total utilizado desciende por debajo del 20%, se puede disminuir el espacio total disponible en el disco a su valor original. Sin embargo, algunos sistemas (por ejemplo software heredado) no están distribuidos y tal vez solo pueden usar un único núcleo de CPU. Entonces, aunque necesitemos aumentar la capacidad de cómputo disponible a pedido, el sistema no puede usar esta capacidad adicional de ninguna forma. Dichos sistemas son no escalables.

Conclusión: Un sistema escalable no depende de la elasticidad. Los entornos en la nube (AWS, Azure, Google Cloud, etc.) ofrecen elasticidad y algunos de sus servicios principales también son escalables desde el primer momento. Además, si creamos un software escalable, podemos implementarlo en estos entornos de nube y beneficiarnos de la infraestructura elástica que proporcionan para aumentar/disminuir automáticamente los recursos de cómputo disponibles.

AWS VPC

Funcionamiento: Amazon Virtual Private Cloud (VPC) brinda un completo control sobre su entorno de redes virtuales, incluidas la ubicación de los recursos, la conectividad y la seguridad.

El primer paso es crear una VPC. Luego, se puede agregar recursos, como instancias de Amazon Elastic Compute Cloud (EC2) y Amazon Relational Database Service (RDS). Por último, se define cómo se comunican las VPC entre sí, entre cuentas, zonas de disponibilidad (AZ) o regiones.

Características de Amazon VPC: Posee características que se pueden utilizar para aumentar y mejorar la seguridad de la nube privada virtual (VPC):

- *Analizador de accesibilidad:* Herramienta de análisis de configuración estática que permite analizar y depurar la accesibilidad de red entre dos recursos en la VPC. Después de especificar los recursos de destino y origen en la VPC, el analizador de accesibilidad produce detalles salto por salto de la ruta virtual entre ellos cuando son accesibles e identifica el componente que genera un bloqueo cuando no son accesibles.
- *Registros de flujos de la VPC:* Puede monitorear los registros de flujo de la VPC entregados a Amazon S3 o Amazon CloudWatch para obtener una visibilidad operativa de las dependencias de red y los patrones de tráfico, detectar anomalías y evitar filtraciones de datos, y solucionar problemas de configuración y conectividad de la red. Los metadatos enriquecidos de los registros de flujo ayudan a adquirir información adicional sobre quién inició las conexiones TCP, así como el destino y el origen reales a nivel de paquete para el tráfico que fluye a través de capas intermedias, como la gateway de NAT. También puede archivar los registros de flujo para ayudar a cumplir ciertos requisitos de conformidad.
- *Replicación de tráfico de VPC:* Permite copiar el tráfico de red de una interfaz de red elástica de instancias de Amazon EC2 y enviar el tráfico a dispositivos de monitoreo y seguridad fuera de banda para la inspección profunda de paquetes. Con la replicación de tráfico de VPC, se puede detectar anomalías de seguridad y de red, obtener información operativa, implementar controles de seguridad y conformidad, y solucionar problemas. La replicación de tráfico de VPC es una característica que proporciona acceso directo a los paquetes de red que fluyen a través de la VPC.
- *Direccionamiento de entrada:* Permite dirigir todo el tráfico de entrada y de salida que fluye a/desde un gateway de Internet (IGW) o gateway privada virtual (VGW) a la interfaz de red elástica de una instancia de EC2 específica. Con esta característica, se puede configurar una nube privada virtual para enviar todo el tráfico a una IGW, una VGW o la instancia de EC2 antes de que el tráfico alcance las cargas de trabajo empresariales.

- *Grupos de seguridad:* Funcionan como un firewall para instancias de Amazon EC2 asociadas. Controlan el tráfico de entrada y de salida a nivel de instancia. Al lanzar una instancia, se la puede asociar a uno o más grupos de seguridad creados. Cada instancia dentro de la VPC puede pertenecer a un conjunto diferente de grupos de seguridad. Si no se especifica un grupo de seguridad al lanzar la instancia, esta se asocia automáticamente al grupo de seguridad predeterminado de la VPC.
- *Lista de control de acceso (ACL) de red:* Es una capa de seguridad opcional para la VPC que actúa como un firewall para controlar el tráfico entrante y saliente de una o más subredes. Se puede configurar ACL de red con reglas similares a las de los grupos de seguridad para agregar una capa de seguridad adicional a la VPC.

Uso de otros recursos de AWS con Amazon VPC:

- *AWS Transit Gateway:* Permite conectar fácilmente las VPC de Amazon, las cuentas de AWS y las redes en las instalaciones a una única gateway. Optimiza la VPC y hace fácilmente escalable la infraestructura. Dirige todo el tráfico de cada VPC o VPN y le brinda un lugar para monitorear.
- *AWS Private Link:* Sirve para establecer conectividad privada entre las VPC y los servicios alojados en AWS o en las instalaciones, sin exponer los datos a Internet.
- *AWS Network Firewall:* Permite implementar seguridad en las redes en las VPC de Amazon de manera sencilla y clara.
- *AWS VPN:* Posibilita extender las redes en las instalaciones a la nube y acceder a ellas de forma segura desde cualquier lugar:
 - *AWS Client VPN:* Servicio de VPN completamente administrado y elástico que aumenta o disminuye automáticamente en función de sus requisitos. Debido a que es una solución de VPN en la nube, no necesita instalar y administrar soluciones basadas en hardware o software, ni intentar estimar cuántos usuarios remotos puede admitir a la vez.
 - *AWS Site-to-Site VPN:* Crea una conexión segura entre su centro de datos o sucursal y sus recursos de la nube de AWS. Para aplicaciones distribuidas globalmente, la opción Accelerated Site-to-Site VPN proporciona un rendimiento todavía mayor gracias al funcionamiento con AWS Global Accelerator.
- *Gateway de traducción de direcciones de red (NAT):* Proporciona que las cargas de trabajo de la subred privada de la VPC obtengan acceso a Internet, a la vez que evita que Internet inicialice una conexión con esas instancias.

C20A – ARMAMOS UN PEQUEÑO AMBIENTE EN AWS

¿Qué es Load Balancer y para qué sirve?

Load Balancer: Distribuye todas las peticiones de forma ordenada. Es una tecnología diseñada para distribuir la carga de trabajo entre distintos servidores o aplicaciones. El objetivo es optimizar las prestaciones globales de la infraestructura, así como su rendimiento y su capacidad. Redistribuye las peticiones a las instancias que están más liberadas.

El balanceo de carga se refiere a la distribución eficiente del tráfico de red entrante a través de un grupo de servidores back end, las granjas de servidores.

Además de garantizar una alta disponibilidad y confiabilidad al enviar solicitudes a servidores que están en línea, permite agregar o quitar servidores automáticamente según lo requiera la demanda del momento.

Con el uso del balanceador de carga aseguramos una reducción del tiempo de actividad, agregamos mayor escalabilidad, disminuimos la redundancia, maximizamos la flexibilidad y aumentamos la eficiencia de nuestra infraestructura.

Load balancer (balanceador de carga): Sistema muy utilizado cuando se tiene que atender mucho tráfico en una aplicación o sitio web. Básicamente, consiste en crear un grupo de servidores que se encarga de atender las solicitudes que recibe un servicio.

El objetivo es optimizar las prestaciones globales de la infraestructura, así como su rendimiento y su capacidad.

Elastic Load Balancing distribuye automáticamente el tráfico de aplicaciones a través de varios destinos, tales como las instancias de Amazon EC2, los contenedores, las direcciones IP, las funciones Lambda y los dispositivos virtuales.

Maximiza la velocidad y la utilización de la capacidad, garantizando que ningún servidor esté sobrecargado de trabajo.

Funciones:

- *Distribuye* las solicitudes de los clientes o la carga de la red de manera eficiente en varios servidores.
- *Garantiza* alta disponibilidad y confiabilidad al enviar solicitudes solo a servidores que están en línea.
- *Proporciona* la flexibilidad de agregar o quitar servidores según lo requiera la demanda.

Beneficios:

- Reducción del tiempo de inactividad
- Escalable
- Redundancia
- Flexibilidad
- Eficiencia

Implementación de un load balancer:

Otros beneficios:

- Bajo costo.
- Continuidad de nuestra app si hubiera algún problema con una instancia.
- Un gateway de seguridad para dejar ingresar solamente a determinados usuarios y puertos.
- Balanceo de carga y mejor uso de los recursos.
- Permite una escalabilidad muy sencilla de aplicar nuestra app.

C22A – ALMACENAMIENTO EN LA NUBE

Almacenamiento en la nube

El almacenamiento en la nube —cloud storage— permite guardar cualquier tipo de archivo en una infraestructura digital con la comodidad de la administración y backup remotos.

¿Qué puedo guardar en la nube?

¿Qué es un BLOB?: 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 BLOB representan datos que no necesariamente se encuentran en un formato nativo.

¿Qué es un file share?: Son archivos para compartir. El intercambio de archivos es el acto de distribuir o proveer acceso a información almacenada digitalmente. 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?: 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 pseudónimo (alias), dependiendo del tipo de base de datos elegida.

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 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.

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, pero 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.

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, por ejemplo, cuando contrata servicios a terceros que no están dentro de la propia empresa. Tiene mucho potencial ya que permite hacer crecer el sistema de almacenamiento de la empresa basándose en 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

Microsoft: Windows Azure Storage: La plataforma principal de Azure Storage es la solución de almacenamiento en la nube de Microsoft. Proporciona almacenamiento para objetos de datos, presenta una alta disponibilidad, es seguro, durable, redundante y se puede escalar de forma masiva.

La plataforma de Azure Storage incluye los servicios de datos siguientes:

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 APIs 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.
-----------------	--	--

Amazon: servicios de almacenamiento de AWS:

Almacenamiento de objetos:



Amazon Simple Storage Service (S3)

Almacenamiento de objetos creado para almacenar y recuperar cualquier volumen de datos desde cualquier ubicación

Almacenamiento de archivos:

<h4>Amazon Elastic File System</h4> <p>Sistema de archivos NFS escalable, elástico y nativo en la nube</p>	<h4>Amazon FSx for Windows File Server</h4> <p>Almacenamiento de archivos completamente administrado basado en Windows Server</p>	<h4>Amazon FSx for Lustre</h4> <p>Sistema de archivos de alto rendimiento completamente administrado e integrado con Amazon S3</p>
--	---	--

Almacenamiento en bloque:



Amazon Elastic Block Store

Almacenamiento en bloque de alto rendimiento y con facilidad de uso a cualquier escala

Transferencias de datos:

<h4>AWS Storage Gateway</h4> <p>Almacenamiento en la nube híbrida que proporciona acceso local al almacenamiento prácticamente ilimitado en la nube</p>	<h4>AWS DataSync</h4> <p>Transfiere los datos fácilmente hacia y desde AWS hasta 10 veces más rápido</p>	<h4>AWS Transfer Family</h4> <p>Transferencia de archivos, simple y sin problemas a Amazon S3 con protocolos SFTP, FTPS y FTP</p>
<h4>Familia de productos AWS Snow</h4> <p>Dispositivos físicos para migrar datos hacia y desde AWS</p>		

Almacenamiento e información de borde:



Conclusión: Actualmente tanto Azure Storage como Amazon Storage soportan sitios web estáticos. Estos se componen de HTML, CSS, JS y otros archivos estáticos, como imágenes o fuentes. Un sitio estático suele ser una aplicación de página única (SPA) escrita con diversos marcos de JS, 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 se necesita procesamiento en el lado del servidor, a menudo se puede satisfacer esas necesidades mediante funciones sin servidor, como las admitidas por cada proveedor.

Amazon S3

Introducción: 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.

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

Ventajas:

- *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:* Permite brindar o denegar acceso a otras personas que desean cargar o descargar datos en nuestro bucket de Amazon S3. También podemos conceder 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:* Permite utilizar las interfaces REST y SOAP basadas en estándares diseñados para trabajar con cualquier conjunto de herramientas de desarrollo de Internet.

Conceptos principales:

Bucket: Es un contenedor para objetos almacenados en Amazon S3, es decir, cada objeto está almacenado en un bucket. 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: 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.

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.

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.

Identity and Access Management: Se utiliza para administrar el acceso a nuestros recursos de Amazon S3.

Operaciones: A continuación, 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 sobrescribirlo 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.

Ventajas del almacenamiento en la nube:



Tiene más espacio físico: Con la nube es posible contratar más espacio si llegamos al límite de almacenamiento y expandirlo de manera simple e inmediata. En otras palabras, almacenar documentos en la nube es perfectamente ajustable a las necesidades de cada empresa, por lo tanto, fácilmente escalable.

Compartir archivos con mayor facilidad: Los documentos digitales hacen los procesos más ágiles porque todas las etapas son realizadas desde un único sistema en línea. Estos documentos pueden ser firmados electrónicamente con seguridad, integridad y conformidad.

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. 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.

Diversidad de archivos: 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.

Optimización automática: Esta es una de las ventajas más importantes.

C23A – BASES DE DATOS EN LA NUBE

Bases de datos como servicio

Modelos primarios de implementación para bases de datos en la nube:

Tradicional	Database as a service (DBaaS)
Muy similar a una base de datos administrada de forma local, excepto por el aprovechamiento de infraestructura.	Una organización se compromete por contrato con un proveedor de servicios en la nube a través de un servicio de suscripción de pago.
Una organización compra espacio de máquina virtual de un proveedor de servicios en la nube y la base de datos se implementa en la nube.	El proveedor de servicios ofrece una variedad de tareas operativas, de mantenimiento, administrativas y de administración de bases de datos en tiempo real para el usuario final.
Los desarrolladores de la organización utilizan un modelo DevOps o personal de tecnología informática tradicional para controlar la base de dato.	Generalmente incluye la automatización en las áreas de aprovisionamiento, respaldo, escalamiento, alta disponibilidad, seguridad, actualización y monitoreo de condición.
La organización es responsable de la supervisión y la administración de la base de datos.	Proporciona a las organizaciones el mayor valor, lo que les permite utilizar la administración de bases de datos tercerizadas optimizada por la automatización del software, en lugar de contratar y administrar expertos internos en bases de datos.

Beneficios:

- *Agilidad e innovación mejoradas:* Se pueden configurar muy rápidamente y se pueden retirar del servicio con la misma rapidez.
- *Menor tiempo de salida al mercado:* Es necesario solicitar hardware ni pasar tiempo esperando envíos, instalación y configuración de red cuando un producto nuevo está en espera para su desarrollo.
- *Riesgos reducidos:* Los proveedores de servicios en la nube pueden usar la automatización para ejecutar las mejores prácticas y funciones de seguridad.
- *Costos más bajos:* Los modelos de suscripción de pago por uso y el escalado dinámico permiten a los usuarios finales aprovisionar para el estado estable, luego escalar para la demanda máxima durante los períodos de mayor actividad y, más tarde, volver a bajar cuando la demanda vuelve al estado estable.

¿Qué buscar a la hora de seleccionar una base de datos en la nube?

- *Rendimiento* en línea y escalabilidad independiente de cómputo y almacenamiento, revisiones y actualizaciones.
- *Seguridad* con cifrado de dato en reposo y activo, y proporcionar actualizaciones de seguridad automatizadas.
- *Una base de datos legible en espera* (combinada con informes) para reducir los costos de alta disponibilidad y tecnologías flashbacks líderes de la industria para ayudar a proporcionar protección contra errores del usuario. Las bases de datos deben tener una amplia compatibilidad con aplicaciones de terceros.

RSD

Amazon RDS: Es un servicio web que facilita el establecimiento, la utilización y el escalado de bases de datos relacionales en la nube. Proporciona una capacidad rentable y redimensionable, al tiempo que gestiona las lentas tareas de administración de bases de datos. La solución de integración de informática Intelligent Cloud Services para Amazon RDS facilita el diseño de integraciones de grandes volúmenes de datos y su implantación en instancias de RDS desde cualquier fuente de la nube o del entorno local, con una compatibilidad total con los motores de bases de datos que admite RDS —como MySQL, Oracle, Microsoft SQL Server y PostgreSQL, entre otros—.

Dynamo DB: Es una base de datos de clave-valor y documentos que ofrecen rendimiento en milisegundos de un solo dígito a cualquier escala.

Se trata de una base de datos completamente administrada, duradera, multiactiva y de varias regiones que cuenta con copia de seguridad, restauración y seguridad integradas, así como almacenamiento de caché en memoria para aplicaciones a escala de Internet. DynamoDB puede gestionar más de 10 billones de solicitudes por día y puede admitir picos de más de 20 millones de solicitudes por segundo.

Beneficios:

- Rendimiento a escala:
 - Admite algunas de las aplicaciones de escala más grandes del mundo.
 - Proporciona tiempos de respuesta en [ms] de un solo dígito a cualquier escala.
- No más administrar servidores:
 - No hay servidores que aprovisionar, parchear o administrar, y no hay suficiente software que instalar, mantener o utilizar.
 - Aumenta o reduce automáticamente las tablas para ajustar la capacidad y mantener el rendimiento
- Uso empresarial:
 - Cifra todos los datos de forma predeterminada.
 - Proporciona un control de acceso e identidad detallado en todas las tablas.
 - Crea copias de seguridad completas de cientos de terabytes de datos al instante.

Casos de uso de DynamoDB:

- Tecnología publicitaria
- Videojuegos
- Venta minorista
- Sector bancario y financiero
- Contenido multimedia y entretenimiento
- Software e Internet