**Unidad 7 – Virtualización**

**Virtualizar**

Significa aparentar que algo con una forma tiene otra. Virtualizar una computadora se trata de simular múltiples computadoras diferentes. También esta misma permite que los SO se ejecuten como apps dentro de otros SO.

**Desventajas:**

# Complejidad añadida a la ejecución del sistema operativo invitado. Hay capas intermedias hasta llegar al hardware.

# Pérdida de prestaciones ocasionada por las capas intermedias entre el sistema invitado y el hardware, además de las limitaciones impuestas por la compartición de los recursos hardware entre diferentes sistemas operativos.

**Máquina virtual**

Es aquella que **emula** a una computadora completa. Las máquinas virtuales **abstraen el hardware de la computadora,** formando entornos de ejecución diferentes. **También** podemos decir que las máquinas virtuales no es más que un software que es capaz de albergar en su interior otro SO. Las máquinas virtuales no significan que no tengan componentes, sino que los componentes están en la máquina host.

**Ventajas:**

# Permite ejecutar diferentes SO simultáneamente sobre un único hardware.

# Permite crear instantáneas, que es guardar un estado para luego volver en el momento que se necesite.

# Las apps ejecutadas están aisladas del SO anfitrión.

# Los sistemas virtualizados pueden ser “Portados” a otro equipo.

**Operaciones posibles con máquinas virtuales:**

* **Exportar** una máquina virtual
* **Importar** una máquina virtual
* **Clonar** una máquina virtual

**Operaciones con máquinas virtuales – redes:**

Conectar múltiples máquinas virtuales permite crear redes virtuales para una variedad de propósitos.

**Tipos de conexiones de red en máquinas virtuales (MV):**

En Virtual box, una máquina virtual puede tener hasta 8 tarjetas de red Ethernet.

* **Modo No conectado:** existe una tarjeta de red per no hay una conexión.
* **Modo Adaptador puente (bridge):** forma mas sencilla de acceder a una red.
* **Modo NAT:** En modo NAT, Virtual Box coloca un router virtual entre el exterior (hacia donde hace NAT) y el SO invitado. modo adecuado para conexión a internet. No se requiere configuración de red.
* **Modo Red Interna:** se pueden construir redes aisladas. Las MV no podrán comunicarse con el equipo anfitrión ni viceversa.
* **Modo Red NAT:** la red funciona como una LAN. Este modo es una extensión del modo NAT. Se trata de la unión entre las características una red NAT (para Internet) y una red Interna (conexión entre máquinas virtuales)
* **Modo solo anfitrión:** Se utiliza para crear una red interna a la que pertenecerá también el equipo anfitrión.

**Escritorios remotos**

Se utilizan para realizar una conexión remota de una PC a otra. Existen dos formas básicas de realizar una conexión remota o escritorio remoto:

* Por software de terceros
* Por Sistema Operativo

**Máquina virtual de proceso**

A diferencia de las maquinas virtuales comunes las maquinas virtaules de proceso **ejecutan un proceso concreto**. Por ejemplo, una app.

**Máquina virtual JAVA**

Interpreta y ejecuta instrucciones expresadas en un código binario especial, intermedio, denominado **bytecode**, el cual es generado por el compilador del lenguaje Java.

**Hipervisores**

El software de virtualización implementa lo que se denomina **hipervisor o VMM (Virtual Machine Monitor)** que consiste en una capa de abstracción entre el hardware de la máquina física y la máquina virtual formada por hardware y software virtualizado, haciendo el papel de intermediario entre lo real y lo virtualizado. Este mismo gestiona la CPU, memoria, almacenamiento y conexiones de red.

**Anfitrión (host):** es el SO con el que arranca.

**Huésped (guest):** es el SO que se instala en la VM.

**Tipos:**

**# Hipervisor tipo 1:** denominado también como “virtualización en modo nativo”, se caracteriza porque este software se instala directamente sobre el equipo haciendo las funciones tanto de sistema operativo (SO) como de virtualización. Se utiliza generalmente en las empresas.

**# Hipervisor tipo 2:** denominado también alojado (hosted), se caracteriza porque debe ser instalado en un equipo que cuente con un SO previo anfitrión (como Linux, MS Windows, Mac OS X, etc.). Este es el método de virtualización apropiado para probar software (sistemas operativos y aplicaciones) sin riesgo de afectar a nada de lo que haya en la máquina anfitriona. Este tipo es menos eficiente que el anterior.

**Unidad 8 – Microservicios**

**Sistema Monolítico**

Es un sistema autónomo el cual no depende de otras piezas de software y obedece únicamente a los componentes que están en su interior

# Los procesos están estrechamente asociados y se ejecutan como un solo servicio.

# Agregar o mejorar características de una aplicación monolítica se vuelve más complejo a medida que crece.

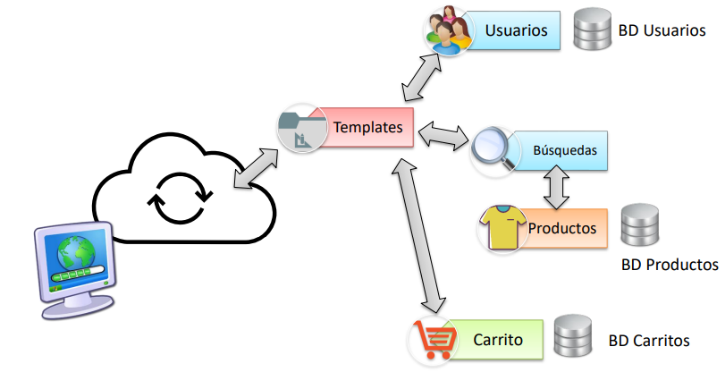
# Las arquitecturas monolíticas aumentan el riesgo de la disponibilidad de la aplicación porque muchos procesos dependientes y vinculados aumentan el impacto del error de un proceso.

**Microservicios**

Es un distintivo **sistema de desarrollo de software y tienen una sencilla escalabilidad**. Los microservicios son un enfoque arquitectónico y organizativo para el desarrollo de software donde el software está compuesto por pequeños servicios independientes que se comunican a través de APIs bien definidas.

* Los propietarios de estos servicios son equipos pequeños independientes.
* Las arquitecturas de microservicios permiten rapidez en el desarrollo de apps.
* También permite innovación.

**Arquitectura de microservicios**



**Apis y Servicios Web**

API (Application Programming Interface – Interfaz de Programación de Aplicaciones) – método de interacción entre los programas.

Son Funciones y protocolos informáticos con los que los desarrolladores pueden crear programas concretos para bases de datos, sistemas operativos, plataformas online o redes sociales. Es la interfaz que facilita que programas distintos puedan comunicarse.

Una API lleva a cabo sus funciones desde dentro de un programa de software. Si se trata de aplicaciones web, la API tiene su base en la web. Las APIs no necesitan, por defecto, basarse en web. Las APIs son un conjunto de especificaciones y reglas que permiten que un programa se comunique con otro.

Un **Servicio Web** utilizará, la mayor parte de las veces, protocolo HTTP para lograr la comunicación. También puede emplear SOAP, REST y XML-RPC.

**Microservicios – Características**

**Autónomos**

* Cada servicio componente en una arquitectura de microservicios se puede desarrollar, implementar, operar y escalar sin afectar el funcionamiento de otros servicios.
* Los servicios no necesitan compartir ninguno de sus códigos o implementaciones con otros servicios.

**Especializados**

* Cada servicio está diseñado para un problema específico.
* Si los desarrolladores aportan más código a un servicio a lo largo del tiempo y el servicio se vuelve complejo, se puede dividir en servicios más pequeños.

**Microservicios – Beneficios**

**Agilidad:** Los equipos se apropian de los servicios y trabajan de forma más rápida.

**Escalado Flexible:** permiten que cada servicio escale de forma independiente y se acomoden dependiendo el aumento de la demanda de ese servicio.

**Implementación sencilla:** Los microservicios permiten la integración y la entrega continuas, lo que facilita probar nuevas ideas y revertirlas si algo no funciona correctamente.

**Libertad tecnológica:** Los equipos eligen la herramienta para resolver problemas específicos.

**Código reutilizable:** La división del software en microservicios permite que un servicio especifico para una tarea pueda ser utilizado por muchas tareas más.

**Resistencia a errores:** cuando surge un error se puede arreglar sin bloquear toda la aplicación ya que se divide en microservicios. Esto mismo no se logra con una arquitectura monolítica.

**Separación del problema:** El microservicio podrá ser desarrollado con cualquier lenguaje de programación o base de datos, podrá ser compilado y ejecutado por cualquier sistema operativo.

**Proyectos pequeños:** Cualquier cambio podrá ser monitoreado y en caso de algún fallo este no afectará al funcionamiento global de la aplicación.

**Componentes de la arquitectura de microservicios**

* **Registro:** Proporciona independencia de la ubicación lógica o física.
* **Servidor perimetral:** Es el punto de acceso a cualquier llamada externa.
* **Balanceador:** Reparte la carga entre los servicios.
* **Servidor de configuración:** Proporciona la configuración del sistema.
* **Sistema de tolerancia a fallos:** Objetivo evitar fallos en cascada.
* **Gestion de logs:** Sistema para poder explotar la trazabilidad de los servicios.

**Microservicios – Desventajas**

**Alto consumo de recursos:** Cada microservicio tiene sus propios recursos y bases de datos, por lo tanto, consumen más memoria y CPU.

**Inversión de tiempo inicial:** Se necesita mas tiempo para la comunicación de microservicios.

**Complejidad en la gestión:** Con un gran número de microservicios, será más complicado controlar la gestión e integración de los mismos.

**Perfil de desarrollador:** Se requiere desarrolladoras de alto nivel de experiencia.

**No uniformidad:** Se produce una arquitectura no uniforme si no se gestiona el equipo.

**Dificultad en la realización de pruebas:** Los test de pruebas son complicados de hacer.

**Coste de implementación alto:** El costo es alto debido a costes de infraestructura y pruebas distribuidas.

Unidad 9. Contenedores de software.

***¿Qué son los contenedores de software?***

**Los contenedores de software son un paquete de elementos que permiten ejecutar una aplicación determinada en cualquier sistema operativo.** Se utilizan para determinar que se ejecute correctamente cuando cambie su entorno. Funciona como una especie de funda.

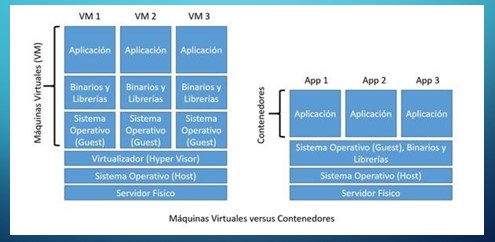
***¿Cómo hacen los contenedores para funcionar aislados?***

Consiste en un proceso, implementando y ajustando la escala de aplicaciones en cualquier entorno, velozmente. Proporciona comandos que, se inician y detienen en caso de ser necesario logrando así una manera sencilla y práctica.

***Contenedores VS. Máquinas virtuales.***

Máquina virtual: **funciona como un emulador de recursos computacionales virtuales** y habilitan la virtualización de la infraestructura computacional.

Contenedores de software: **separan procesos, memoria, interfaces, red y sistemas de archivo usando espacios o hosts.** Además, permiten la virtualización de las aplicaciones. Estos son ambientes de ejecución livianos que proveen a las aplicaciones archivos, variables y librerías que necesitan para operar. Solo requieren recursos computacionales mínimos y son rápidos y fáciles de instalar. Puede ser desplegados en clusters. Los desarrolladores pueden actualizar componentes individuales sin retrabar la aplicación completa.



***Principios para la construcción de contenedores.***

1. Utilizar la imagen del contenedor como la primitiva básica y la plataforma de orquestación de contenedores como el entorno de tiempo de ejecución del contenedor de destino.
2. Los contenedores resultantes se comportarán como un buen integrante cloud native en la mayoría de los motores de orquestación de contenedores, lo que permite programarlos, escalarlos y supervisarlos de manera automatizada. **Cabe destacar que estos principios no llevan un orden.**

***¿Qué es cloud native?***

Cloud native en la nube es un enfoque en el desarrollo de software que utiliza la computación en la nube para "crear y ejecutar aplicaciones escalables en entornos modernos y dinámicos, como nubes públicas, privadas e híbridas".

***Principio de preocupación única (SCP)***

La palabra "preocupación" en el principio de SCP resalta la preocupación como un nivel más alto de abstracción que la responsabilidad y describe mejor el alcance como un contenedor en lugar de una clase. **La principal motivación para SCP es la reutilización y la sustitución de la imagen del contenedor.**

Por lo tanto, el principio de SCP indica que cada contenedor debe abordar una sola preocupación y que debe hacerlo correctamente.

***Principio de gran observancia (HOP)***

**El HOP indica que el contenedor debe proporcionar API para que la plataforma pueda leer.**

**Los contenedores proporcionan una forma unificada para empaquetar y ejecutar aplicaciones tratándolas como una caja negra.**

En términos prácticos, como mínimo, la aplicación en contenedor debe proporcionar API para los diferentes tipos de comprobaciones de estado: disponibilidad y agilidad. La aplicación debe registrar eventos importantes en el error estándar (STDERR) y la salida estándar (STDOUT) para la agregación de registros mediante herramientas.

***Principio del cumplimiento del ciclo de vida (LCP)***

**El LCP indica que la aplicación debe proporcionar una forma de leer los eventos provenientes de la plataforma.** Es casi como tener una "API de escritura" en la aplicación para interactuar con la plataforma.

Pero algunos eventos son más importantes que otros.

***Principio de inmutabilidad de las imágenes (IIP)***

Las aplicaciones en contenedores deben ser inmutables y, una vez construidas, no se espera que cambien entre diferentes entornos. Cualquier cambio en la aplicación en contenedor debería dar como resultado el desarrollo de una nueva imagen de contenedor y su reutilización en todos los entornos.

**Si se sigue el principio de IIP, se debería evitar la creación de imágenes de contenedor similares para diferentes entornos y se debería conservar una imagen de contenedor configurada para cada entorno.**

***Principio de deshecho de procesos (PDP)***

**Una de las principales motivaciones para pasar a aplicaciones en contenedores es que los contenedores deben ser lo más efímeros posible y deben estar listos para ser reemplazados por otra instancia de contenedor en cualquier momento.**

Esto significa que las aplicaciones en contenedor deben mantener su estado externalizado o distribuido y redundante.

Tener contenedores más pequeños conduce a tiempos de inicio más rápidos porque, antes de reiniciarse, los contenedores se deben copiar físicamente en el sistema host.

***Principio de autonomía (S-CP)***

**Este principio indica que un contenedor debe contener todo lo que necesita en el momento de la compilación.** El contenedor debe confiar solamente en la presencia del kernel de Linux® y tener bibliotecas adicionales agregadas al momento del desarrollo del contenedor

**Algunas aplicaciones están compuestas por múltiples componentes en contenedores.** Por ejemplo, una aplicación web en contenedor también puede requerir un contenedor de base de datos. Este principio no sugiere la fusión de ambos contenedores.

***Principio de limitación de tiempo de ejecución (RCP)***

**Este principio de RCP sugiere que cada contenedor declare sus requisitos de recursos y transmita esa información a la plataforma.**

***Mejores prácticas a tener en cuenta.***

**Tenga como objetivo imágenes pequeñas**. Esto reduce el tamaño del contenedor, el tiempo de compilación y el tiempo de la red al copiar imágenes de contenedores.

**Admita ID de usuario arbitrarias**. Evite usar el comando sudo o requiera una ID de usuario específica para ejecutar el contenedor.

**Marque los puertos importantes**. Si bien es posible especificar números de puerto en el tiempo de ejecución, al especificarlos mediante el comando EXPOSE, es más fácil para los humanos y para el software utilizar la imagen.

**Use volúmenes para los datos persistentes.** Los datos que se deben conservar después de destruir un contenedor se deben escribir en un volumen.

**Configure metadatos de imágenes**. Los metadatos de las imágenes en forma de etiquetas, indicadores y anotaciones hacen que las imágenes de los contenedores sean más utilizables, lo que genera una mejor experiencia para los desarrolladores que usan sus imágenes.

**Sincronice el host y la imagen**. Algunas aplicaciones en contenedores requieren que el contenedor se sincronice con el host en ciertos atributos, como la hora y la ID de la máquina.

Beneficios del uso de contenedores.

* Instalación y despliegue más simple
* Independencia de plataforma
* Incremento en la eficiencia del proceso de virtualización
* Mejor y mayor aislamiento
* Mejor y mayor capacidad de gestión y automatización de contenedores

UNIDAD 10. CLOUD COMPUTING

***Introducción. Cloud Computing.***

La computación en nube (cloud computing) puede verse como un nuevo estilo de computación en el cual los recursos, dinámicamente escalables y frecuentemente virtualizados, son provistos como servicios sobre Internet.

***Definición.***

**El término computación en nube, más allá de un concepto monolítico, representa una arquitectura donde se integran el conjunto de tecnologías.**

1. ***Software como servicio (SaaS: Software as a Service):***

Es el modelo en el cual una aplicación es alojada como un servicio para los consumidores quienes acceden a él a través de Internet. (Algunas de estas aplicaciones incluyen video conferencia, análisis web, etc.)

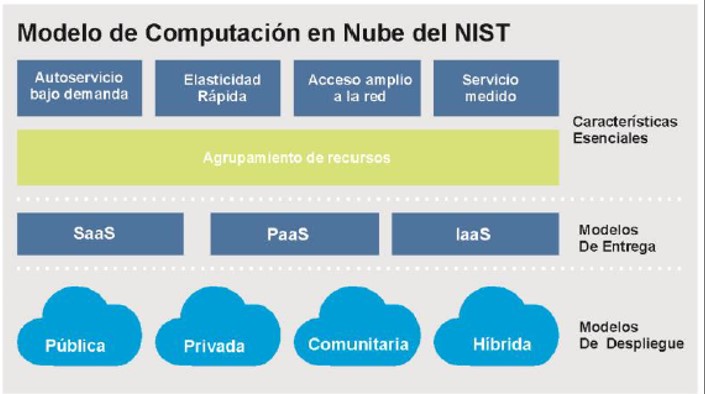
1. ***Plataforma como servicio (PaaS: Plataform as a Service):***

Suministra todos los recursos necesarios para desarrollar completamente aplicaciones y servicios desde Internet, sin tener que descargar o instalar software.

1. ***Infraestructura como servicio (IaaS: Infraestructure as a service):***

Permite alquilar los recursos tales como servidores, software, redes, memoria, capacidad de procesamiento y almacenamiento, del proveedor en lugar de tener que comprarlos.

Este modelo de nube está compuesto por cinco características esenciales, tres modelos de servicios y cuatro modelos de despliegue. Según NIST: National Institute of Standards and Technology.



***Cloud Computing. Modelo NIST.***

* ***1. Autoservicio bajo demanda.***

Un consumidor puede proveerse unilateralmente recursos sin un proveedor de servicio.

* ***2. Acceso amplio a la red.***

Las capacidades están disponibles en la red y se accede mediante mecanismos estándares.

* ***3. Agrupamiento de recursos.***

Los recursos de computación del proveedor están agrupados (*pooling)* para servir a múltiples consumidores utilizando un modelo multidistribuido (*multitenant*).

* ***4. Elasticidad rápida.***

Las funcionalidades se pueden proporcionar de manera rápida y elástica y, en algunos casos, automáticamente.

* ***5. Servicio medido.***

Los sistemas en nube controlan y optimizan automáticamente el uso de recursos, potenciando la capacidad de medición en un nivel de abstracción apropiado al tipo de servicio (almacenamiento, procesamiento, ancho de banda y cuentas activas de usuario).

***Cloud Computing. Características esenciales NIST.***

En este contexto las principales modalidades de pago utilizadas por los proveedores de la nube son:

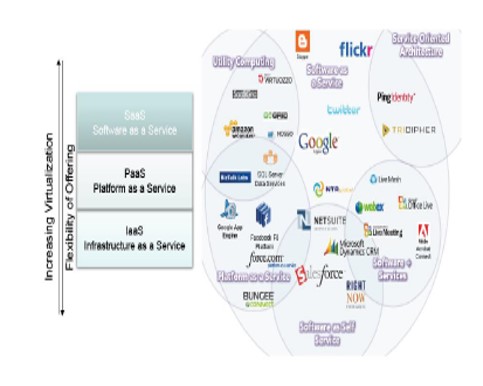
* Pago por disponibilidad del servicio.
* Pago por uso.
* Pago por paquetes escalables.

A partir de estas características esenciales, se definen cinco pasos tecnológicos necesarios para evolucionar hacia una infraestructura en la nube [5]:

* Virtualización de servidores.
* Provisión automática.
* Desprovisión automática.
* Asignación dinámica de recursos.
* Medición que posibilite la facturación basada en el uso.

***Cloud Computing. Modelos de servicios.***

**La computación en nube puede verse como un conjunto de servicios, los cuales se presentan como una Arquitectura de Computación en Nube en Capas.** A continuación, los servicios específicos de la nube.



***Cloud Computing. Modelos de despliegue.***

**El mismo refiere a la localización y administración de la infraestructura de la nube**. El NIST los clasifica en nubes de 4 categorías.

* ***1. Nube Privada.***

La infraestructura en nube está preparada para el uso exclusivo de una única organización que comprende varios consumidores (por ejemplo, unidades de negocio).

* ***2. Nube Pública.***

La infraestructura en nube está preparada para el uso abierto por el público en general.

* ***3. Nube Comunitaria.***

La infraestructura en nube está preparada para el uso de una comunidad específica de consumidores de organizaciones que tienen intereses compartidos (por ejemplo, misión, requerimientos de seguridad y políticas).

* ***4. Nube Hibrida.***

La infraestructura en nube es una combinación de una o más infraestructuras en nube distintas (privada, pública o comunitaria) que permanecen como entidades únicas, pero están unidas por tecnologías estándares o propietarias que permiten la portabilidad de los datos y las aplicaciones.

***Cloud Computing. Beneficios.***

* Reducción de costos.
* Facilidad de uso.
* Calidad del servicio (QoS: Quality of service).
* Fiabilidad.
* Tercerización de servicios IT.
* Facilidad de mantenimiento y actualización.
* Barreras de entrada bajas.
* Debido a que las redes en nube operan más eficientemente y con mayor utilización, frecuentemente se encuentran reducciones significativas en los costos.