TEMA 1: INTRODUCCIÓN. CONCEPTOS BÁSICOS SOBRE SSOO.

LABORATORIO DE REDES Y SISTEMAS OPERATIVOS

U-tad
CENTRO UNIVERSITARIO DE TECNOLOGÍA Y ARTE DIGITAL

Centro adscrito a
Universidad Camilo José Cela

ÍNDICE

ÍNDICE 2

1. OBJETIVOS DEL TEMA 4

2. INTRODUCCIÓN A LOS SISTEMAS OPERATIVOS 5

2.1 ¿Qué son los Sistemas Operativos? 5

2.2 Historia de los Sistemas Operativos 5

2.3 GNU/Linux 9

2.4 Tipos de Sistemas Operativos 12

2.5 Funciones y partes de los Sistemas Operativos 14

2.6 Características de los Sistemas Operativos 17

2.7 ¿Qué ocurre cuando se ejecuta una aplicación de usuario? 18

3. VIRTUALIZACIÓN 19

3.1 Introducción 19

3.2 ¿Qué es la virtualización? 19

3.3 ¿Cómo funciona la virtualización? 19

3.4 Tipos de virtualización 21

4. CONTENEDORES DE APLICACIONES 23

4.1 Introducción 23

4.2 Arquitectura de un contenedor 23

4.3 Contenedores de aplicaciones 24

4.4 Dockers 26

4.5 Kubernetes 27

4.6 Despliegue de aplicaciones en contenedores 28

5. INSTALACIÓN DE VIRTUAL BOX 29

5.1 Introducción 29

5.2 Instalación de Virtual Box 29

5.3 Creación de una máquina virtual sobre Virtual Box 30

5.4 Instalación de Ubuntu sobre una máquina virtual de Virtual Box 32

1. OBJETIVOS DEL TEMA

Los objetivos de este tema son:

- Conocer qué es un sistema operativo, sus partes y funciones.
- Identificar los sistemas operativos actuales, así como la historia que tienen detrás cada uno de estos.
- Saber cada uno de los componentes básicos de un equipo informático (ordenador) y su función.
- Conocer las capacidades de la Virtualización y Contenedores de aplicaciones actuales.
- Saber instalar el software de VirtualBox y sistema operativo Linux (ubuntu)
- Saber cómo crear máquinas virtuales sobre Virtual Box.

2. INTRODUCCIÓN A LOS SISTEMAS OPERATIVOS

2.1 ¿Qué son los Sistemas Operativos?

Un equipo informático (servidor, ordenador de sobremesa, portátil, tablet, smartphone,…) consta de una capacidad de procesamiento (uno o más procesadores), una memoria principal, discos, teclado, una pantalla o monitor, interfaces de red y otros dispositivos de entrada/salida.

Todos los componentes anteriores, son componentes Hardware (HW) por lo que se hace necesario de una capa Software (SW) que los coordine de manera organizada y óptima, permitiendo el uso de cada uno de los componentes por las aplicaciones de usuario. **Esta capa es la correspondiente al Sistemas Operativo.**

El objetivo de los sistemas operativos es ofrecer a las aplicaciones de usuario un equipo informático más simple, de forma que la administración de los recursos HW sea realizada por los sistemas operativos y se simplifiquen así las aplicaciones de usuario.

En la base del gráfico previo, se encuentra el hardware que incluye, tarjetas, discos, teclado, ... es decir todo aquello que sea un componente físico (tangible).

A continuación de la capa hardware se encuentra una capa software que es el sistema operativo. El sistema operativo tiene **dos funciones principales**: abstraer a las aplicaciones de los detalles físicos del equipo informático y permitir que múltiples aplicaciones y usuarios compartan el mismo equipo. **En la capa superior se encuentras las aplicaciones de usuario.**

2.2 Historia de los Sistemas Operativos

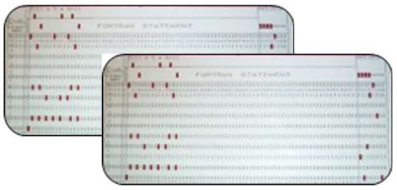
Antes de continuar con las funciones y partes de un SS.OO. nos adentraremos brevemente en el origen de los SSOO.

Década de los 50´s

Los Sistemas Operativos tienen su origen con los equipos informáticos de segunda generación, los Mainframe de los años 50. Estos equipos leían los programas y datos de las tarjetas perforadas y los cargaban en una cinta magnética que se ejecutaba en el mainframe.

Con el fin de automatizar el proceso de cargar las tarjetas para cada programa, surge la idea del procesamiento por lotes (programa llamado batch), se crea un programa que permitía secuenciar y mantener esta ejecución en pequeños equipos informáticos (como el IBM 1401), que eran muy adecuados para leer las tarjetas, copiar cintas e imprimir los resultados, pero no eran tan adecuados para los cálculos numéricos.

IBM 1401



Tarjetas Perforadas

Esos programas se pueden considerar el embrión de los Sistemas Operativos.

Años 60 …

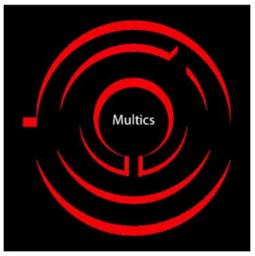
- Por un lado, IBM desarrolla su línea para equipos mainframes 360: **DOS/360 y VM**.

- Y el MIT, *Massachusetts Institute of Technology,* comienza un proyecto experimental llamado CTSS (Compatible Time Sharing System), basado en la idea de los sistemas de tiempo compartido. Este sistema fue el antecedente de Multics, a su vez, inspiración de UNIX de Bell Labs.

Multics (Multiplexed Information and Computing Service) 1964

MIT, Bell Labs y General Electric

- Proyecto muy innovador que desarrolló muchos elementos de los Sistemas Operativos modernos, pero que resultó un fracaso comercial, posiblemente porque estaba muy adelantado a su tiempo.
- Tuvo gran impacto en el campo de la computación gracias a sus muchas ideas nuevas y valiosas.



Se desarrollan los primeros sistemas operativos modernos, con capacidades multiprogramación y multiusuario

Década de los 70's….

A finales de los 60's (1969), Bell Labs se retira de Multics. Ken Thompson y Dennis Ritchie empiezan a trabajar en un Sistema Operativo propio con las ideas aprendidas en Multics, una versión reducida a la que llaman Unics, que finalmente se convierte en **UNIX**.

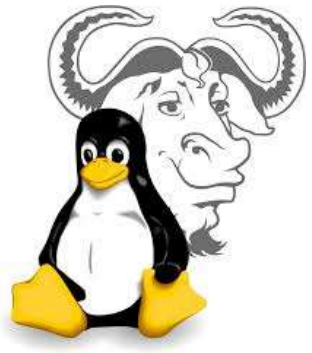
Los años 70's marcaron el inicio de UNIX

- Es el primer Sistema Operativo escrito en un ***lenguaje de alto nivel, C***, que desarrolla con este propósito Dennis Ritchie. Esto favoreció su portabilidad a distintas plataformas HW.
- Unix establece un estándar de facto que perdura hasta hoy.
- Bell Labs no podía legalmente vender software, por lo que empezó a ceder licencias gratuitas a universidades, incluyendo el código fuente.
- En 1978 en Berkeley se desarrolla la BSD (Berkeley Software Distribution), a partir de la V6 de Unix. Bell Labs cede licencias comerciales a fabricantes como HP, Sun y Silicon Graphics que desarrollan sus propias versiones.

Los grandes fabricantes de HW empiezan a tomar conciencia del valor del SW y dejan de entregar los códigos fuentes del sistema operativo de sus equipos y comienza a cuajarse la idea del software libre que años más tarde dará origen a Linux.

Década de los 80`s ….

- **Richard Stallman** se propuso crear su propio sistema operativo iniciando un proyecto llamado GNU en enero de 1984.
- **En el proyecto GNU** definió el **concepto de software libre** y la necesidad de que todo el mundo contribuyeran con él. **GNU desarrolla un sistema operativo de tipo Unix**, aunque su nombre GNU es un acrónimo recursivo que en realidad significa que "GNU No es Unix".



- **En este contexto,** y cuando el proyecto GNU no tenía ningún núcleo estable para su sistema operativo (kernel), **Linus Torvalds decide crear en agosto de 1991 su propio kernel** basado en MINIX (un sistema operativo creado por Andrew Tanenbaum, en 1987 con fines docentes) y **que recibe el nombre de Linux.** (1).
- **GNU/Linux es por tanto una familia de Sistemas Operativos, basados en el núcleo (kernel) de Linux**, y lanzado con licencia GNU/GPL.
- Actualmente, Torvalds trabaja con la Linux Foundation y la Open Source Development Labs, siendo el propietario de la marca Linux.

Video: https://www.youtube.com/watch?v=V1y-mbWM3B8

- Es en también en la década de los 80 cuando IBM encargó a una empresa desconocida, Microsoft, el desarrollo de un Sistema Operativo para su Personal Computer. **La versión de este sistema operativo de IBM se llamó: MS-DOS.**

- MS-DOS era monousuario, monoproceso y dominó el mercado de ordenadores personales durante la década de los 80.

- A finales de esa década se lanza Windows, que originalmente sólo fue un recubrimiento gráfico de MS-DOS.

- IBM cometió el error de no firmar exclusividad, con lo que Microsoft podía vender su Sistema Operativo a los fabricantes asiáticos de clones.
- MS-DOS fue por tanto la base de los sistemas operativos Windows en sus posteriores versiones posteriores.

Década de los 90`s ….

- **GNU/Linux,** se despliega con un entorno de ventanas, gracias a un servidor gráfico y a gestores de ventanas como **KDE, GNOME** entre muchos. Lo que permite utilizar Linux de una forma visual atractiva.
- **Windows,** lanza una nueva versión **Windows NT** donde se decide cambiar el núcleo del sistema compatible con MS-DOS y de ahí en adelante basarlo en uno compatible con Windows NT desarrollando también algunos drivers básicos nuevos para esta versión.

Año 2000 y siguientes …..

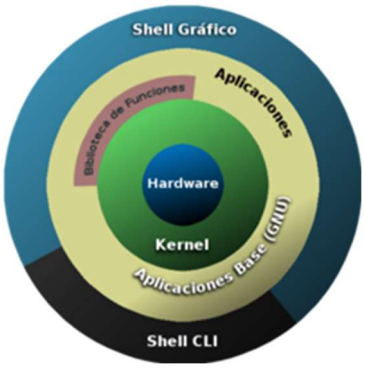
- Se lanzan diferentes versiones de Windows, Windows XP, Windows Vista, Windows 7, Windows 10 y el actual, Windows 11.

2.3 GNU/Linux

Las distribuciones GNU/Linux, incluyen el kernel de Linux, más software del sistema y librerías, una parte de las cuales las proporciona el Proyecto GNU.

El objetivo es que el software creado por los programadores beneficie a todo el mundo y se mantenga gratis en colaboración por todos.

La mayoría de las distribuciones usan el nombre de **Linux** para el conjunto GNU/linux.



Uso de Linux:

- El uso de Linux **en ordenadores de usuario es minoritario,** prácticamente sólo lo usan desarrolladores, administradores de sistemas y amantes de la tecnología.
- En cambio, **los centros de datos con "granja de servidores" usan Linux de forma mayoritaria** (+90%), en vez de Windows, como es el caso de Google, Twitter, Facebook, Amazon. **Las razones más importantes al respecto:**
  - **Coste** (gratis). El Costo total (TCO, Total Cost of Ownership), incluso en versiones empresariales (con mantenimiento 24 x 7) es menor. No olvidemos que una granja de servidores puede llegar a tener miles de ordenadores.
  - **Código abierto**, lo que permite ver el Kernel, arreglar problemas e incluso crear tus programas y añadirlos a Linux (scripting).
  - **Estabilidad y Escalabilidad**. Diseño multiusuario desde el principio (basado en UNIX). Mejor resistencia a fallos, en funcionamiento continuo que Windows. Cuando se introducen cambios no se necesita re-arrancar el sistema.
  - **Acceso a aplicaciones** (gratis) de código abierto. Un ejemplo LibreOffice vs MicroSoft Office.
  - **Soporte comunitario.** Miles de empresas comerciales e individuos soportan su desarrollo, evolución y mantenimiento.
  - **Impacto mínimo de virus.**
  - **Flexibilidad.** Es posible "componer" tu Linux, basado en las necesidades específicas de su escenario de uso (ej. Programación, servidores web, escritorio, sistema de servidores), usando, además del kernel común, los bloques necesarios con su adaptación específica.
    - Ejemplos; Guadalinex, gnuLinuEX, Gibraltar Linux, CERN(CentoS)
  - Además del núcleo Linux, una **distro** (o distribución Linux) suele incluir bibliotecas y herramientas del proyecto GNU y el sistema de ventanas X Window System,

pudiendo incluir también otro tipo de software interesante como procesadores de texto, hojas de cálculo, reproductores multimedia, etc.

Los servidores, especialmente los usados en **granjas de ordenadores**, no tienen teclado ni monitor (**headless servers**). Es importante una eficaz red IP interna, para distribuir la carga de trabajo de los diferentes servidores que componen la granja.
En este tipo de arquitecturas hardware es muy importante el papel del sistema operativo, ya que va a realizar la compartición de carga de trabajo, así como la sincronización y comunicación de los diferentes equipos.

Es por ello **por lo que las distribuciones Linux profesionales, incluyen versiones "ligeras**" para servidores (Red Hat, CentoS, Ubuntu, etc), en las que se elimina lo que no es necesario, como la interfaz gráfica de usuario con ventanas, al estilo de Windows (Desktop), solo se incluye el terminal de Linux, para poder ejecutar scripts.

Por ejemplo, Ubuntu ofrece versiones Desktop, Server, (y también para **IoT** y **Cloud**)

Los Superordenadores…

Los superordenadores usan una arquitectura de grupos de máquinas trabajando en paralelo, similar a la de las granjas de servidores usadas para las aplicaciones web cliente – servidor.

En los superordenadores se usa Linux y son utilizados para simulaciones de diversas clases (meteorológicas, astrofísica, bombas nucleares, materiales, energía, terremotos, petróleo, etc), cálculos matemáticos, Inteligencia Artificial, etc.

La potencia de los superordenadores se mide en FLOPS. Un FLOP es una operación en coma flotante, por segundo. Las operaciones en coma flotante son las típicamente usadas en cálculos científicos. Un MIP es una operación entre números enteros por segundo.

Una gran mayoría de empresas así como los entornos cloud, tienen granja de servidores, superordenadores, almacenamiento, equipos de comunicaciones,…. en edificios creados de manera específica para ello y no en sus propios edificios.

Estos edificios se llaman **datacenters** y ofrecen una serie de capacidades, como redundancia eléctrica (más de una compañía), backups, aislamiento ante fenómenos atmosféricos... En este video se presenta uno de los datacenter utilizado por Google que revolucionó la tecnología usada hasta el momento en dichos edificios. https://youtu.be/zRwPSFpLX8I

Una granja de servidores de Google

Curiosidad: En la lista de los TOP 500 supercomputadores (https://www.top500.org/), **desde 2015 no hay ordenadores con Windows, todos usan Linux en diversas versiones**, desde versiones configuradas a medida, a Red Hut y CentoS. En la lista TOP 500 hay un superordenador de España, **el Mare Nostrum** (Barcelona).

Supeordenador Summit, del Oak Ridge National Laboratory.

Con 200 Petaflops, y se compone de 4608 servidores

https://www.ornl.gov/news/ornl-launches-summit-supercomputer

2.4 Tipos de Sistemas Operativos

- **A nivel de usuario**, podemos decir que hay dos tipos de Sistemas Operativos:
  - Sistemas Operativos para PCs: **Windows, Linux, MacOS**
  - Sistemas Operativos para teléfonos móviles: **Android y el iOS**

- Pero también hay una amplia familia de sistemas operativos categorizados **según los tipos de equipos informáticos que controlan y el tipo de aplicaciones que admiten:**

  - **Sistema operativo en tiempo real**:

    Los sistemas operativos en tiempo real se utilizan para controlar maquinaria, instrumentos científicos y sistemas industriales.

  - **Sistema operativo monotarea:**

    Este sistema operativo está diseñado para administrar la computadora de modo que un usuario pueda hacer una cosa a la vez.

  - **Sistema operativo multitarea:**

    Este es el tipo de sistema operativo que la mayoría de la gente usa en sus equipos de escritorio y portátiles en la actualidad.

    Windows de Microsoft y las plataformas MacOS de Apple son ejemplos de sistemas operativos que permitirán que un solo usuario tenga varios programas en funcionamiento al mismo tiempo.

  - **Sistema operativo multiusuario:**

    Un sistema operativo multiusuario permite que muchos usuarios diferentes aprovechen los recursos del equipo simultáneamente.

- También podemos clasificarlos como **sistemas operativos de pago y sistemas operativos libres**.

- Y por último decir que también hay **Sistemas Operativos en la Nube (CloudOS):**
  El conocido sistema operativo ya no está enclaustrado en tu equipo físico, ahora se puede utilizar nuestro navegador web como vía de acceso a un sistema operativo, estando realmente el sistema operativo alojado en la nube y ejecutándose en un servidor remoto.

  Estos Sistemas Operativos también se pueden llamar "**Sistema Operativo Virtual**", "**Sistema Operativo Web**" o "**OS Cloud**".

  Permite al usuario acceder al servicio con aplicaciones preinstaladas a través de la navegación por Internet. Se incluyen aplicaciones como procesador de textos, lector de PDF y muchas más, pero en

este caso, la aplicación o aplicaciones se utilizan como un **servicio** en lugar de un software independiente.

La bueno de la idea es que todo se almacena en la nube y, por lo tanto, estará disponible en cualquier lugar del mundo que vayas y en cualquier momento.

El servidor realiza todo el procesamiento y guarda los datos, mientras que el equipo de usuario solo necesita suficientes recursos para ejecutar y mostrar la interfaz, lo que podría hacerse con solo un navegador web.

Las empresas de todos los tamaños están adoptando la computación en la nube a un ritmo cada vez mayor, ya que les brinda grandes beneficios como la rentabilidad, ya que en realidad no tienen que comprar los recursos de hardware y software, sino simplemente pagar por uso (servicios).

Un ejemplo de Sistema operativo en la nube es: **SilveOS** y su url es https://www.silveos.com/

2.5 Funciones y partes de los Sistemas Operativos

Los Sistemas Operativos, tienen dos funciones claramente definidas y ya mencionadas anteriormente:

- **Abstraer de los detalles físicos del equipo informático gestionando todos los recursos hardware para todas las aplicaciones que se ejecutan en él.** La ejecución de una aplicación puede generar N procesos.
    - Los procesos se ejecutan en un entorno "virtual" y acceden a los recursos usando instrucciones primitivas del sistema (llamadas a interfaces que ofrece el sistema operativo para acceder a la memoria, o acceder a un dato del almacenamiento, o procesar datos,…).
    - Un proceso es un conjunto de líneas de código (instrucciones) de una aplicación de usuario orientada a realizar alguna determinada función.

- **Permite que múltiples aplicaciones y usuarios compartan el mismo equipo de forma segura.** Esto es que los procesos no se solapen en memoria, o que no se bloqueen al utilizar los procesadores del equipo, o mantengan un orden en las operaciones de E/S, etc….

Por otro lado, un sistema operativo tiene dos Modos de Operación:

I. **Modo Kernel o modo Supervisor:**

- Todo el software del Sistema Operativo se ejecuta en este modo.
- Cuando el sistema operativo trabaja en este modo tiene capacidad de acceso a cualquier recurso hardware y puede ejecutar cualquier software compatible con la arquitectura.

II. **Modo usuario:**

- En este modo se ejecutan las aplicaciones de usuario (tercera capa de la imagen anterior).

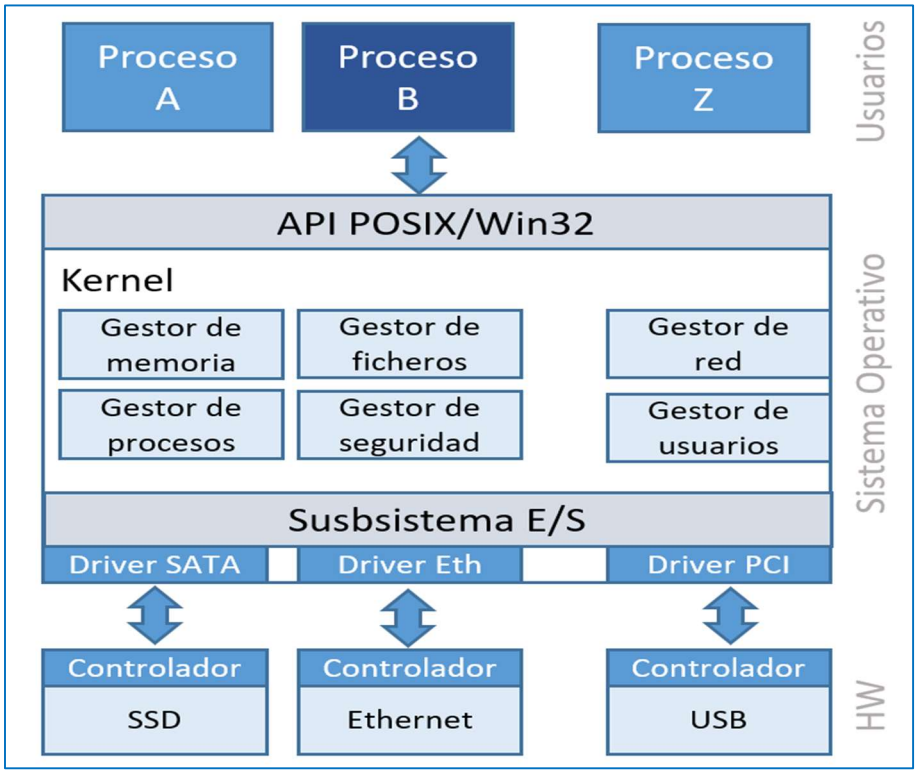
- Estas aplicaciones no pueden ejecutar determinadas instrucciones de control del sistema, así como instrucciones relacionadas con las operaciones de E/S.

¿Qué es una aplicación de usuario?

Software, formado por miles o millones de líneas de código (instrucciones), cuyo diseño y creación está orientado a ofrecer a los usuarios una o varios funciones o tareas.

Ejemplo: Editor de texto, Navegador Web o Aplicación cliente de correo electrónico.

Ya se ha esbozado la estructura de un sistema operativo, si bien en este apartado se profundizará, de forma que el alumno tenga dicha composición clara.



En el gráfico anterior**, un proceso**, pueden pertenecer a una o varias aplicaciones y lógicamente también a uno o varios usuarios, en caso de ser un sistema operativo multiusuario.

1. **El Kernel o "Núcleo"**:
   - Es la capa básica del sistema operativo donde se encuentra toda la lógica de éste.
   - Esta capa básica de código es la primera que se carga en memoria cuando arranca el SO.
   - Proporciona un control de nivel básico sobre todos los dispositivos de hardware del sistema.
   - Sus funciones principales incluyen leer y escribir datos en la memoria, procesar las órdenes de ejecución de cada uno de los procesos y determinar cómo gestionar los dispositivos externos.
   - Como se puede ver en la imagen, tiene diferentes gestores especializados en la realización de las diferentes tareas. Estos gestores son los ofrecen primitivas o interfaces a las aplicaciones de usuario (procesos) y son ellos quien se encargan de la tarea concreta.

2. **Drivers o Controladores**: permiten la comunicación entre el sistema operativo (o el gestor del kernel que corresponda) y componente hardware del equipo que corresponda.
   - Un driver es un programa de bajo nivel que maneja las particularidades de cada dispositivo hardware del sistema. Hace transparente la complejidad del dispositivo hardware al sistema operativo.
   - Es una pieza software especializada en gestionar y administrar de manera eficiente un dispositivo hardware (disco SSD, USB, Impresora,…). De esta manera cuando se adquiere un nuevo componente hardware para el equipo, es software es necesario instalarlo y configurarlo.

     Ejemplo: Nueva Impresora de marca HP Deskjet 4500. En este caso se tendrá que instalar el driver correspondiente a esta impresora que sea compatible con el sistema operativo instalado en mi equipo (Windows, Linux,….) de forma que un aplicación de usuario Microsoft Word pueda utilizar la misma.

3. **Interfaces de Usuario:** permite al sistema operativo interactuar con el usuario de dos formas:
   - A través de "Línea de comandos" (Shell) o intérprete de comandos, donde usuarios o administradores introducen un comando que se ejecuta para realizar una determinada acción.
   - A través de una interfaz gráfica muy común en los sistemas operativos modernos orientados a usuarios finales, de forma que facilitan las diferentes operativas y permiten la labor de administración de manera más simple. Estos interfaces no siempre ofrecen a este perfil de usuarios todas las capacidades y opciones que se pueden realizar mediante la "línea de comandos"

4. **APIs:** Interfaces de Programación de Aplicaciones para desarrollo y adaptación de módulos del sistema operativo. Mediante estas interfaces, las aplicaciones de usuario y módulos del sistema operativo generan llamadas para realizar las tareas que tienen que llevar a cabo (acceso a un fichero, presentación de datos en pantalla, abrir una aplicación,….)

    - **WIN32:** conjunto de funciones residentes en bibliotecas (llamadas DLL término usado para referirse a éstas en Windows) que permiten que una aplicación (o su conjunto de procesos) corra bajo un determinado sistema operativo.

    - **POSIX:** norma, descrita por la IEEE, que define una interfaz estándar del sistema operativo y el entorno, incluyendo un intérprete de comandos (o "shell") y programas de utilidades comunes para apoyar la portabilidad de las aplicaciones de usuario a nivel de código fuente. De forma general cuando se escribe una aplicación que cumple la norma POSIX, esto indicará que es portable en la mayoría de los sistemas operativos tipo Linux.

2.6 Características de los Sistemas Operativos

- **Eficiencia**: El sistema operativo permite el uso de los recursos del equipo informático o sistema de manera mucho más eficiente, exponiendo una serie de interfaces hacia los programas de usuario para su uso. De esta forma todos los programas de usuario acceden de la misma manera a los recursos del sistema y no cada uno como su diseñador entiende más adecuado.

- **Actualizable**: El sistema operativo se tiene que poder actualizar en cualquier momento por parte del administrador del usuario, de manera que se incorporen nuevas funciones al mismo pero siempre manteniendo el funcionamiento de las existentes, es decir, por incluir nuevas funcionalidades o hacer más eficientes las existentes, no pueden dejar de funcionar las que ya lo hacen con versiones anteriores.

- **Administrador del Hardware**: El sistema operativo se administra los recursos hardware del equipo, (memoria, procesadores, almacenamiento,…) de manera óptima, asignando dichos recursos de manera compartida a las diferentes aplicaciones de usuario que se están ejecutando.

- **Habilitador de comunicaciones** entre las aplicaciones de usuario y los diferentes dispositivos o periféricos cuando así sea necesario.

- **Organizar los datos** del equipo para un acceso rápido y seguro.

- **Gestionar las comunicaciones de red**: permitir la gestión de las tarjetas de red, interfaces de red externas con otras redes. Ofrecer a los administradores del sistema capacidad de instalación y configuración de éstas.

- **Permitir gestionar un equipo informático** sin tener que conocer las características a bajo nivel de cada uno de los componentes.

2.7 ¿Qué ocurre cuando se ejecuta una aplicación de usuario?

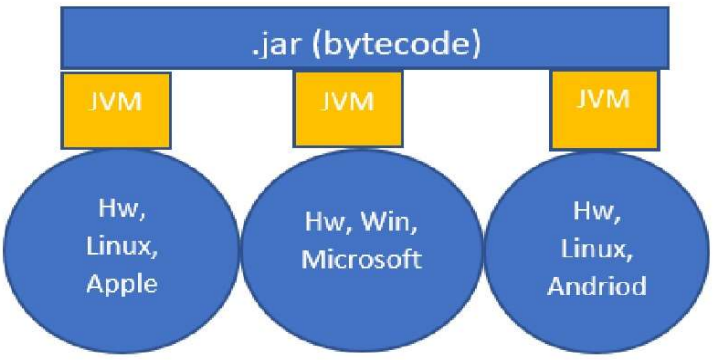
El sistema operativo se encarga de mover, **desde el disco duro a la memoria RAM, tanto el bloque con el código de la aplicación de usuario** (compilado, en código máquina – códigos de ceros y unos que se cargan en memoria y que son las instrucciones y datos que se ejecutan) **como el bloque de datos del programa,** siendo la CPU quien ejecutará las operaciones.

Igualmente es el sistema operativo es que decide mover un programa o aplicación de usuario a memoria secundaria (disco duro o almacenamiento masivo) cuando no está siendo utilizado o cuando se necesita la memoria principal para otro aplicativo.

El sistema operativo tiene estrategias de planificación de aplicaciones en memoria, así como cuando, con qué prioridad, se ejecutarán los procesos de un aplicativo de usuario.

Existen diferentes tipos de programas con los que un sistema operativo tiene que saber trabajar:

- **Compilados**, en los que se necesita un compilador (que es realmente un programa) para pasar del código fuente del aplicativo al código máquina, ejecutándose el bloque de código máquina generado (fichero.exe). Son por ejemplo: C, C++, Pascal, etc.
- Hay programas **interpretados**, en los que al ejecutarlos se va traduciendo directamente a código máquina, no hay código máquina accesible. Son por ejemplo: Phyton, Visual Basic, JavaScript, PHP, etc.
- **Java**: es un lenguaje particular porque es compilado, pero se compila a un lenguaje intermedio llamado bytecode (.jar), que después es interpretado.
    - El JRE (Java Runtime Environment) es el programa que se encarga de interpretar el bytecode al que son compilados los programas de Java.
    - Estos programas son ejecutados sobre la Máquina Virtual Java (JVM), Así el .jar (bytecode) es transportable a cualquier sistema operativo / hardware (Android, Windows, MacOS) y corre sobre la JVM.



Portabilidad de Java

3. VIRTUALIZACIÓN

3.1 Introducción

En este apartado veremos algunos aspectos relevantes sobre la virtualización, los tipos que existen y su aplicación práctica en general.

3.2 ¿Qué es la virtualización?

La **virtualización es una tecnología que permite crear servicios de IT, con recursos que están tradicionalmente limitados al hardware**.

Esto se consigue gracias a que distribuye las funciones de una máquina física entre varios usuarios o entornos, de forma que se permite el uso de toda la capacidad de la máquina.

Históricamente la mayoría de las empresas tenían servidores físicos de un solo proveedor, por lo cual las aplicaciones heredadas no podían ejecutarse en sistemas de hardware de otro proveedor y por tanto requería otro hardware específico y dedicado. Con el consiguiente desaprovechamiento de recursos y los costes de inversión tan elevados.

A medida que las empresas actualizaban sus entornos de TI con servidores básicos, sistemas operativos y aplicaciones menos costosas y de diferentes proveedores, el hardware físico se usaba de manera insuficiente por lo que la ***virtualización*** cobró gran protagonismo.

La virtualización se tornó como solución natural a esos dos problemas consiguiendo que:

i. **Las empresas pudieran dividir los servidores *y* ejecutar aplicaciones en varios tipos y versiones de sistemas operativos.**

ii. **Los servidores se empezaron a utilizar más eficientemente y, en consecuencia, se redujeron los costes relacionados con las compras, la instalación, la refrigeración y el mantenimiento.**

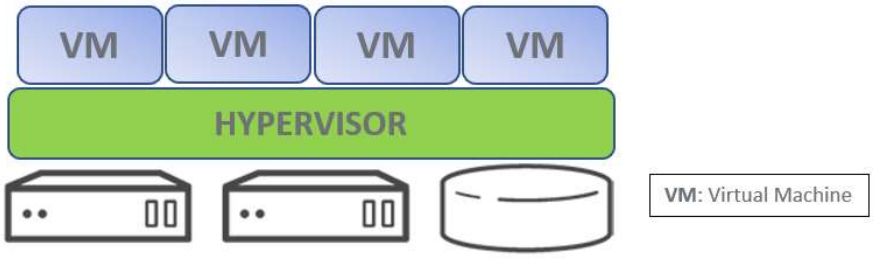
La adopción generalizada de la virtualización ayudó a reducir la dependencia de un solo proveedor y constituyó la base del **cloud computing**.

Actualmente, su uso se ha expandido tanto en las empresas que a menudo se necesita un software de gestión de virtualización especializado para realizar el seguimiento.

3.3 ¿Cómo funciona la virtualización?

El software denominado **hipervisor** separa los recursos físicos de los entornos virtuales que los necesitan.

Los hipervisores controlan los recursos físicos y los dividen de manera tal que los entornos virtuales puedan usarlos como si estos fueran utilizados exclusivamente por ellos.



Los recursos se dividen según las necesidades, desde el entorno físico hasta los numerosos entornos virtuales.

Cuando el entorno virtual se está ejecutando, y un usuario o aplicación ejecuta una instrucción que requiere recursos adicionales del entorno físico, el hipervisor transmite la solicitud al sistema hardware físico, en coordinación con sistema operativo y almacena los cambios en la caché. El hipervisor gestiona las necesidades de recursos de manera dinámica.

Todo esto sucede prácticamente a la misma velocidad que habría si este proceso se realizara dentro de la máquina física.

Hay aplicaciones de virtualización libres: chroot, Docker, etc (Linux) y propietarias: VMWare, HP-UX, Spoon, Sandboxie (Windows).

VirtualBox (Oracle) es software libre y soporta diferentes sistemas operativos **(este será el que utilicemos para la asignatura).**

En la siguiente figura podemos ver **la arquitectura de un contenedor de máquinas virtuales, VirtualBox**, en la que **el sistema operativo anfitrión es Windows**, y los sistemas operativos invitados son: Linux, Win, MacOS.

Los sistemas operativos invitados pueden ser totalmente estancos entre ellos, sin compartición de recursos, ni comunicación entre ellos e incluso con diferentes direcciones IP

VirtualBox – Contenedor de máquinas virtuales

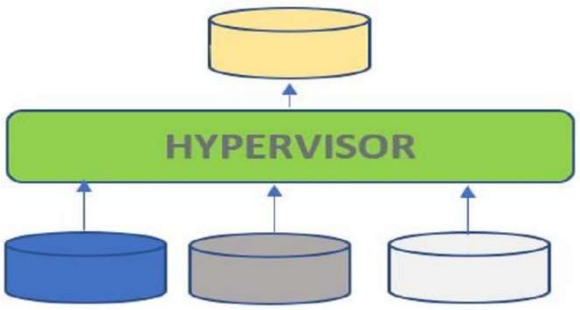
La máquina virtual funciona como un archivo de datos único; por eso, es posible trasladarla de un equipo a otro, abrirla en cualquiera de ellas y trabajar de la misma forma.

3.4 Tipos de virtualización

- *Virtualización de los datos*

  Los datos que se encuentran distribuidos en varias ubicaciones pueden consolidarse en una sola fuente.

  La virtualización de los datos posibilita que los sistemas IT de las empresas traten los datos como si fueran un único suministro dinámico, ya que proporciona funciones de procesamiento que permiten reunir datos de varias fuentes, incorporar fuentes nuevas fácilmente y transformar los datos según las necesidades de los usuarios.
  Las herramientas que forman parte de este proceso interactúan con varias fuentes de datos y permiten tratarlas como si fueran solo una. Gracias a ello, cualquier aplicación o usuario puede obtener los datos que necesita, de la manera que los requiere en el momento justo.



- *Virtualización de escritorios*

  La virtualización de escritorios suele confundirse con la virtualización de los sistemas operativos, la cual permite implementar muchos de estos en una sola máquina.

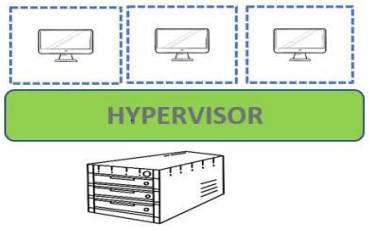
  La virtualización de escritorios permite que un administrador central o una herramienta de administración automatizada implementen entornos simulados de escritorio en cientos de máquinas físicas al mismo tiempo. En este modelo el SO y las aplicaciones se ejecutan en un servidor centra generalmente ubicado en la nube o en un centro de datos.

  A diferencia de los entornos de escritorio tradicionales que se instalan, configuran y actualizan físicamente en cada máquina, la virtualización de escritorios permite que los administradores realicen múltiples configuraciones, actualizaciones y controles de seguridad en todos los escritorios virtuales de forma remota, sistemática y automatizada.

- *Virtualización de los servidores*

  La virtualización de un servidor, implica dividirlo en varios servidores virtuales, para que sus componentes puedan utilizarse para realizar varias tareas o funciones de manera mucho más específica.

- *Virtualización de los sistemas operativos*



Los sistemas operativos se virtualizan en el **kernel**, es decir, los gestores de tareas propias del sistema operativo.
Es una forma sencilla de ejecutar varios SSOO de manera paralela en un mismo equipo.

Las empresas también pueden insertar sistemas operativos virtuales en los equipos, lo cual:

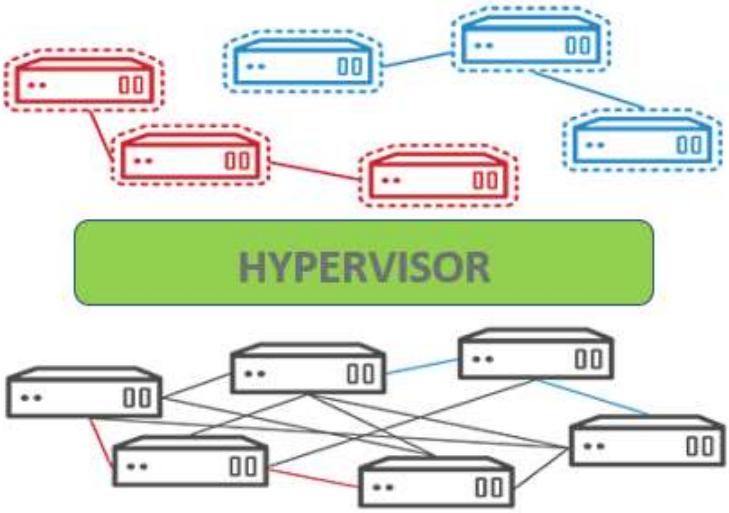
  - Reduce grandes inversiones en sistemas hardware.
  - Aumenta la seguridad porque todas las instancias virtuales se pueden supervisar y aislar.
  - Limita el tiempo que se destina a los servicios de TI, como las actualizaciones de software.

- *Virtualización de las funciones de red*

La virtualización de las funciones de red (NFV – Network function virtual) permite transformar una red que depende típicamente del hardware en una que se basa en el software.

La virtualización de red puede combinar varias redes físicas en una red virtual, mediante software o dividir una red física en redes virtuales independientes y separadas. La virtualización de red puede combinar varias redes físicas en una red virtual, mediante software o dividir una red física en redes virtuales independientes y separadas.

La virtualización de redes, que se utiliza con frecuencia en el sector de las telecomunicaciones, reduce la cantidad de elementos físicos (como conmutadores, enrutadores, servidores, cables y centrales) que se necesitan para crear varias redes independientes.



4. CONTENEDORES DE APLICACIONES

4.1 Introducción

En este apartado, daremos un repaso sobre los conceptos de contenedores y contenedores de aplicaciones, haciendo foco también en las diferencias existentes con las máquinas virtuales (VM).

4.2 Arquitectura de un contenedor

Los contenedores son una forma de virtualización del sistema operativo.

Los contenedores son unidades ejecutables de software donde se empaqueta el código de aplicación, junto con sus bibliotecas y dependencias, y por tanto todo lo necesario para ejecutarse.

Un solo contenedor se puede usar para ejecutar cualquier cosa, desde un microservicio o un proceso de software a una aplicación de mayor tamaño.

Dentro de un contenedor se encuentran todos los ejecutables, el código binario, las bibliotecas y los archivos de configuración necesarios.

Sin embargo, en comparación con los métodos de virtualización de máquinas o servidores, los contenedores **no contienen imágenes del sistema operativo**. Esto los hace más ligeros y portátiles, con una sobrecarga significativamente menor.

En ocasiones, se confunde la tecnología de contenedores con máquinas virtuales (VM) o con la tecnología de virtualización de servidores, cuando realmente los contenedores son muy diferentes de las máquinas VM:

- Las VM se ejecutan en un entorno de hipervisor en el que cada máquina virtual debe incluir su propio sistema operativo invitado dentro del mismo, junto con sus archivos binarios, bibliotecas y archivos de aplicaciones correspondientes. Esto consume una gran cantidad de recursos y genera mucha sobrecarga, especialmente cuando se ejecutan varias VM en el mismo servidor físico, cada una con su propio sistema operativo invitado.
- Por el contrario, cada contenedor comparte el mismo sistema operativo host o kernel del sistema y tiene un tamaño mucho menor. Esto suele implicar que un contenedor puede tardar unos segundos en iniciarse en comparación con los minutos necesarios que requiere una VM típica.

Los contenedores aparecieron por primera vez hace décadas con versiones como FreeBSD Jails y AIX Workload Partitions, pero el inicio de la era moderna del contenedor comienza con la introducción de Docker.

En implementaciones de aplicaciones de mayor tamaño, se pueden poner en marcha varios contenedores como uno o varios clústeres de contenedores. Estos clústeres se pueden gestionar mediante un orquestador de contenedores, como Kubernetes que veremos con más detalle en los siguientes apartados.

4.3 Contenedores de aplicaciones

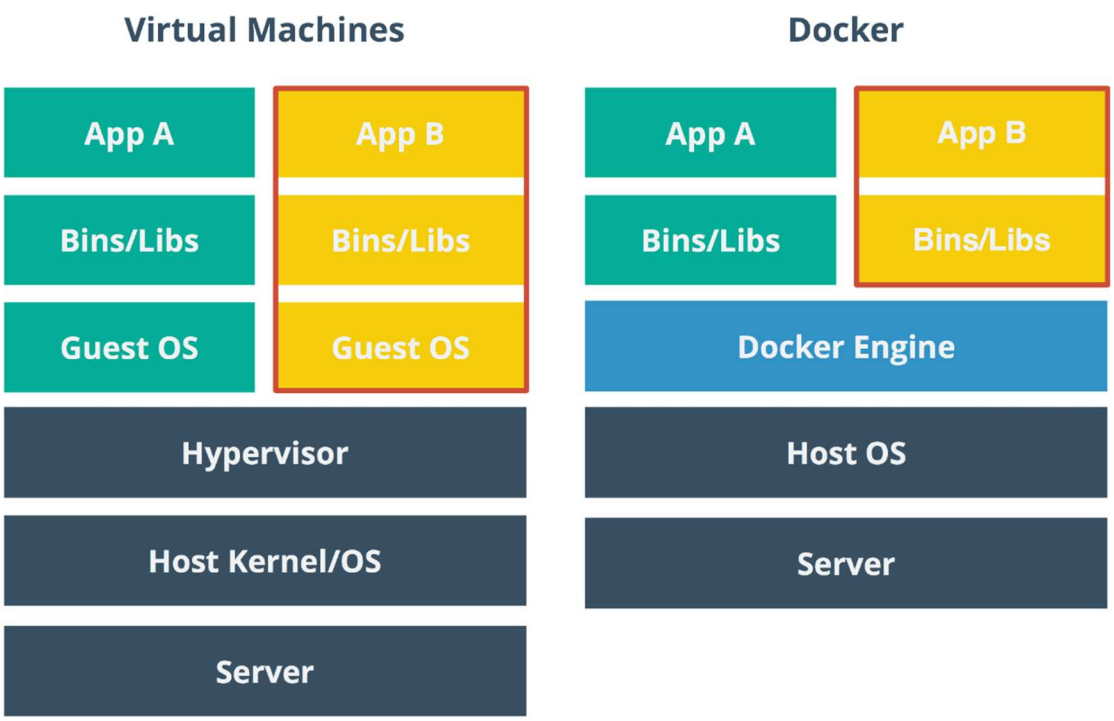
La diferencia clave entre los contenedores y las máquinas virtuales es que, estas virtualizan toda una máquina completa hasta las capas de hardware mientras que los contenedores solo virtualizan las capas de software por encima del nivel del sistema operativo.

Es un método de virtualización en el que el kernel (núcleo) del sistema operativo permite que coexistan varios contenedores de aplicaciones aislados como si se tratasen de uno solo, un ejemplo es Docker.

Dicha tecnología, consiste en empaquetar una aplicación con todas las dependencias que necesita para ejecutarse (librerías, configuraciones…), **todo ello sobre un Kernel de un Sistema Operativo.**

Este método de virtualización sólo permite el uso de aplicaciones del mismo sistema operativo, por ejemplo, Linux.

Los contenedores también se llaman contendores de software, motores de virtualización (VE o Virtualization Engines) o prisiones (prisión FreeBSD o prisión chroot).



Los contenedores son una forma optimizada de crear, probar, poner en marcha aplicaciones en varios entornos, desde un portátil local de un desarrollador hasta un centro de datos on-premises e incluso en despliegues en la nube.

Algunos de los beneficios de los contenedores son:

- ✓ **Menos sobrecarga**
  - Los contenedores requieren menos recursos del sistema (ordenador, servidor o equipo informático) que los entornos de máquinas virtuales tradicionales o de hardware porque no incluyen imágenes del sistema operativo.

- ✓ **Mayor portabilidad**
  - Las aplicaciones que se ejecutan en contenedores se pueden poner en marcha fácilmente en sistemas operativos y plataformas de hardware diferentes.

- ✓ **Funcionamiento más constante**
  - Los equipos de DevOps saben que las aplicaciones en contenedores van a ejecutarse igual, independientemente de dónde se despliegue dicha aplicación.

- ✓ **Mayor eficiencia**
  - Los contenedores permiten poner en marcha o desplegar en producción, aplicar parches o escalar las aplicaciones con mayor rapidez.

- ✓ **Mejor desarrollo de aplicaciones**

- Los contenedores respaldan los esfuerzos ágiles y de DevOps para acelerar los ciclos de desarrollo, prueba y producción.

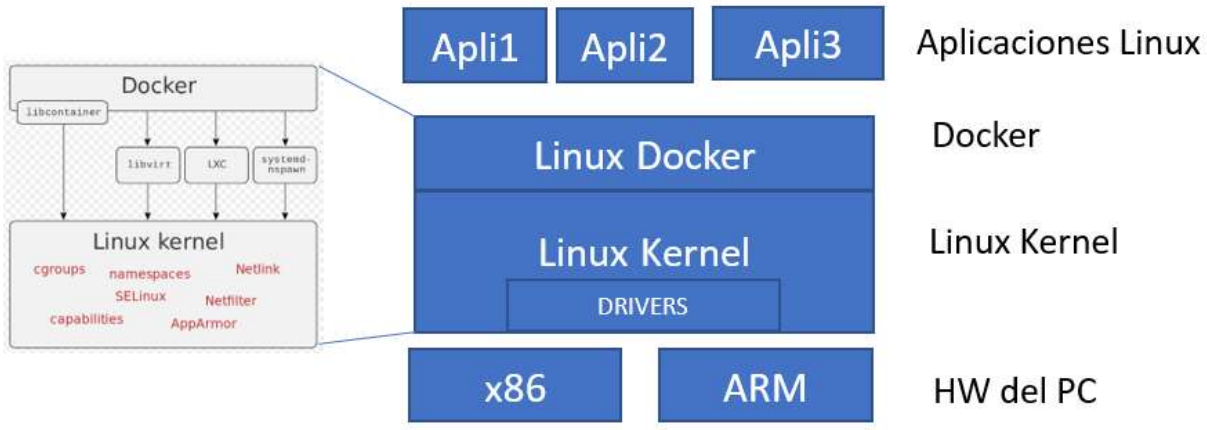
4.4 Dockers

Docker es un proyecto de código abierto, en Linux, que permite la virtualización de aplicaciones (Linux) aisladas dentro de contenedores de software.
Evita el uso de los contenedores de máquinas virtuales (VBox, VMWare).

Con **DOCKER**, puede usar los contenedores como "máquinas virtuales" extremadamente livianas y modulares. Además, obtiene flexibilidad con estos contenedores: puede crearlos, implementarlos, copiarlos y moverlos de un entorno a otro, lo cual le permite optimizar sus aplicaciones para la nube

Proporciona una API para proporcionar contenedores aislados y simplifica la creación de sistemas altamente distribuidos dentro de única instancia de Linux.

Docker, y la virtualización a nivel de sistema operativo, es más eficiente en recursos y velocidad, al eliminar una capa intermedia de software, **pero sólo permite aplicaciones Linux.**



Dockers soporta, entre otros:

- Amazon Web Services
- Google Cloud Platform
- IBM Bluemix
- Microsoft Azure
- Vagrant

Docker es una herramienta diseñada para beneficiar tanto a desarrolladores, testers, como administradores de sistemas, en relación a las máquinas, a los entornos en sí donde se ejecutan las aplicaciones software, los procesos de despliegue, etc.

En el caso de los desarrolladores, el uso de Docker hace que puedan centrarse en desarrollar su código, atendiendo a los requisitos, sin preocuparse de si dicho código funcionará en la máquina en la que se ejecutará.

Docker también es muy bueno para la fase de testing, para tener entornos de pruebas.

Por un lado, es muy sencillo crear y borrar un contenedor, además de que son muy ligeros, por lo que podemos ejecutar varios contenedores en una misma máquina (donde dicho contenedor tendría el entorno de nuestra aplicación: base de datos, servidor, librerías…).
Por otro, un mismo contenedor funcionará en cualquier máquina Linux: un portátil, un ordenador sobremesa residencial, máquinas alojadas en la nube, un servidor propio,…

Esto además beneficia a la parte económica, ya como los contenedores son más ligeros que las máquinas virtuales, se reduce el número de máquinas necesarias para tener un entorno.

Este enfoque como Dokers también impacta en la forma de despliegue de aplicaciones en entornos de producción o entornos pre-productivos que contaremos en el último apartado.

4.5 Kubernetes

Las aplicaciones en contenedores pueden ser complicadas y con gran cantidad de módulos.

Durante la producción, muchas pueden requerir cientos o miles de contenedores independientes.
Es en este punto donde los entornos en tiempo de ejecución de contenedores, como Docker, se benefician del uso de otras herramientas para orquestar o gestionar todos los contenedores en funcionamiento.

Una de las herramientas más populares para este fin es **Kubernetes**, un orquestador de contenedores que reconoce varios entornos en tiempo de ejecución de contenedores, incluido Docker.

Kubernetes orquesta el funcionamiento de varios contenedores juntos de forma armónica. Gestiona áreas como el uso de recursos de infraestructura subyacentes para aplicaciones en contenedores (por ejemplo, la cantidad de recursos de procesamiento, red y almacenamiento necesarios).
Las herramientas de orquestación como Kubernetes facilitan la automatización y el escalado de cargas de trabajo basadas en contenedores para entornos de producción activos.

Sirve para la automatización del despliegue, escalado y manejo de múltiples contenedores y sus aplicaciones.

Es código libre, donado por Google en 2015 y fruto de la experiencia de Google con sus granjas de ordenadores. Es compatibles con otras plataformas como AWS y Azure

Los desarrolladores pueden crear aplicaciones en la nube con Kubernetes como una plataforma de tiempo de ejecución utilizando patrones, que son las herramientas necesarias para diseñar aplicaciones y servicios basados en los contenedores.

Pero, realmente ¿Qué es lo que puede hacer con Kubernetes?

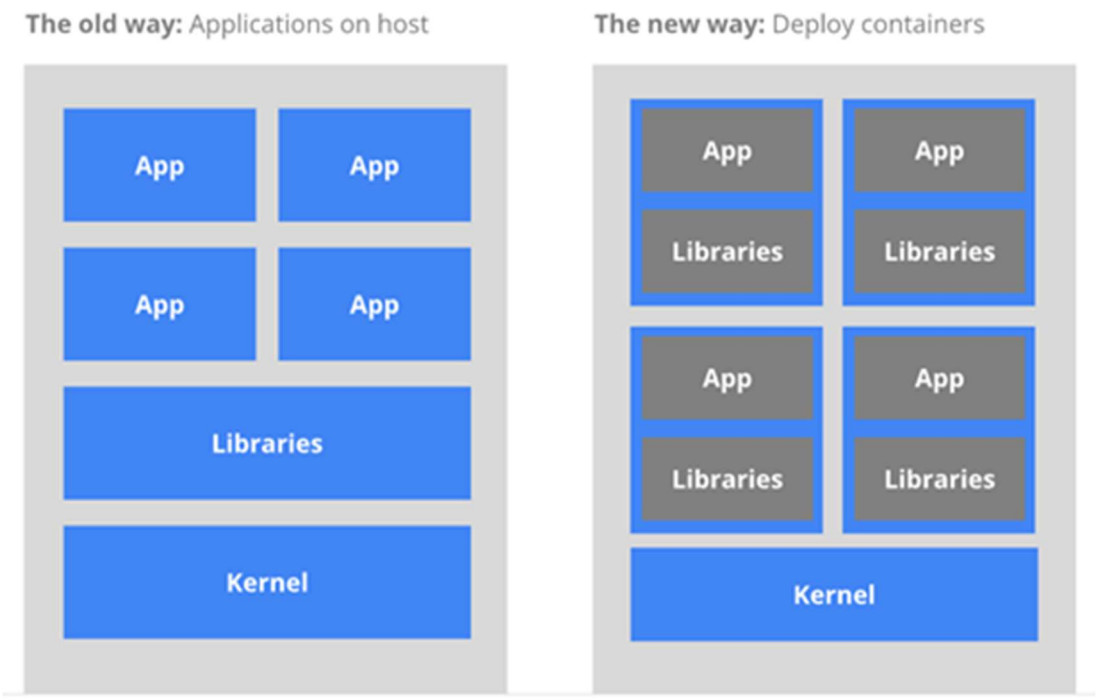
- Organizar los contenedores en varios hosts
- Hacer un mejor uso del hardware para aprovechar al máximo los recursos necesarios en la ejecución de las aplicaciones empresariales
- Controlar y automatizar las implementaciones y actualizaciones de las aplicaciones
- Agregar almacenamiento para ejecutar aplicaciones con estado. (Aplicaciones en las que cada flujo o transacción conoce el estado de la aplicación y en función de este decide, transitar a un estado o página o a otro)
- Ampliar las aplicaciones en contenedores y sus recursos según sea necesario (demanda dinámica – esto lo ofrecen entornos cloud de forma nativa)
- Gestionar los servicios de forma declarativa para garantizar que las aplicaciones implementadas siempre se ejecuten correctamente
- Realizar comprobaciones de estado y auto regeneraciones de sus aplicaciones con ubicación, reinicio, replicación y adaptación automáticos

4.6 Despliegue de aplicaciones en contenedores

La "**manera antigua**" de desplegar aplicaciones era instalarlas en un servidor usando el administrador de paquetes del sistema operativo.

La desventaja era que los ejecutables, la configuración, las librerías y el ciclo de vida de todos estos componentes se relacionaban unos con otros aumentando la complejidad.
Podríamos construir imágenes de máquina virtual inmutables para tener rollouts y rollbacks predecibles, pero las máquinas virtuales eran pesadas y poco portables.

La "**nueva manera**" es desplegar contenedores basados en virtualización a nivel del sistema operativo, en vez del hardware.
Estos contenedores están aislados entre ellos y con el servidor anfitrión: tienen sus propios sistemas de archivos, no ven los procesos de los demás y el uso de recursos puede ser limitado.
Son más fáciles de construir que una máquina virtual, y como no están acoplados a la infraestructura y sistema de archivos del anfitrión, pueden llevarse entre entornos Cloud y distribuciones de sistema operativo.



5. INSTALACIÓN DE VIRTUAL BOX

5.1 Introducción

En este apartamos veremos, desde cero, la instalación de Virtual Box.

Posteriormente crearemos una máquina virtual para contener un Sistema Operativo Linux, en concreto Ubuntu, en su versión más actual.

Por último, terminaremos con la instalación propia del sistema operativo Ubuntu sobre esta máquina virtual, y que servirá de base y entorno de trabajo para nuevos temas de la asignatura.

5.2 Instalación de Virtual Box

VirtualBox es un software para virtualización, también conocido como **hipervisor de tipo 2,** que se utilizar para virtualizar sistemas operativos dentro de nuestro ordenador, creando lo que se conoce como máquina virtual.

Un hipervisor de tipo 2 se diferencia con los de tipo 1 en que necesita un sistema operativo para funcionar, a diferencia de los de tipo 1 en los que el propio hipervisor funciona sobre el hardware, o máquina host, por este motivo, en el último apartado de esta sección se instalará un sistema operativo Linux (Ubuntu) sobre la máquina virtual creada, ya que de otra manera no es posible que funcione.

¿Para qué sirve VirtualBox?

Para virtualizar sistemas operativos que no podamos o no queramos ejecutar de forma nativa en nuestro equipo.

Esto servirá también para trabajar sobre esos sistemas operativos con un abanico de posibilidades, pues una de las principales virtudes de una máquina virtual es el aislamiento que ello proporciona.

Otro uso que le podemos dar a una máquina virtual es el de apoyarnos en términos de compatibilidad, y es que algunos programas/aplicaciones determinadas, no pueden funcionar en versiones más modernas del sistema operativo para el que se diseñaron y crearon, así que, en estos escenarios, este es un método imprescindible para poder seguir utilizando dichas aplicaciones en entornos productivos.

Una vez que tenemos el contexto suficiente de lo que es VirtualBox y para qué sirve vamos a explicar en el siguiente video cada uno de los pasos para la instalación y configuración del dicho software.

URL de descarga: **https://www.virtualbox.org/wiki/Downloads**

Video 1 de ayuda para la instalación: ***INSO_LRSO_Instalacion_VBOX*.mp4**

5.3 Creación de una máquina virtual sobre Virtual Box

Una vez tenemos Virtual Vox instalado el siguiente paso es proceder a crear máquinas virtuales.

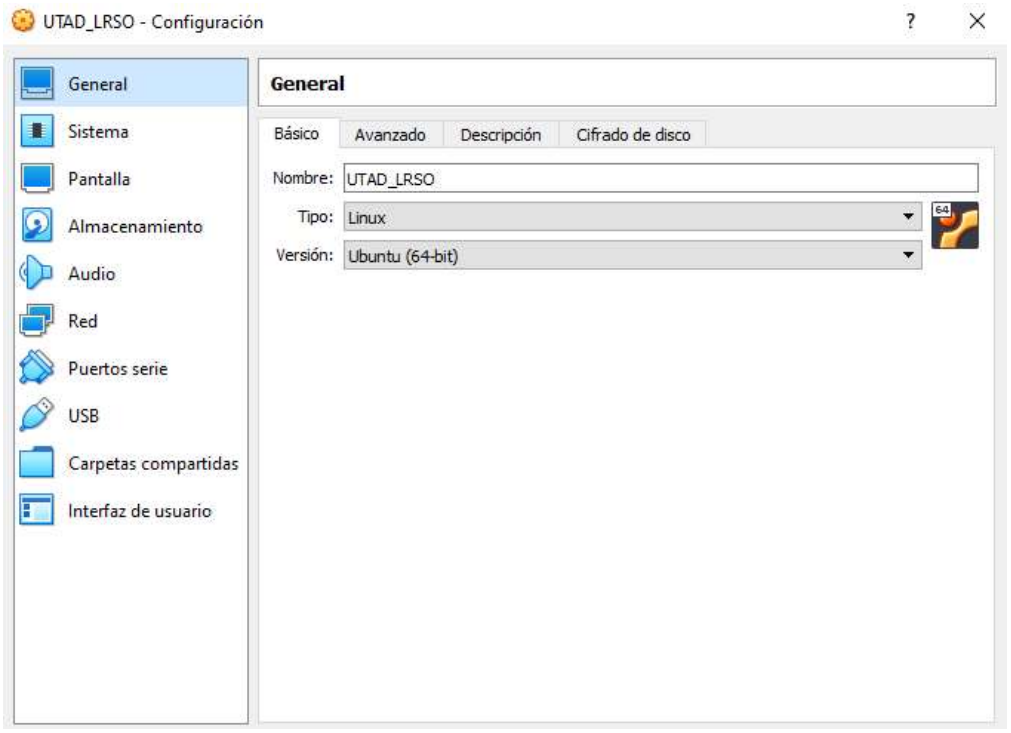
La creación de una primera MV se explica en el siguiente video:

Video 2 de ayuda para la instalación: **INSO_LRSO_Instalacion_VM.mp4**

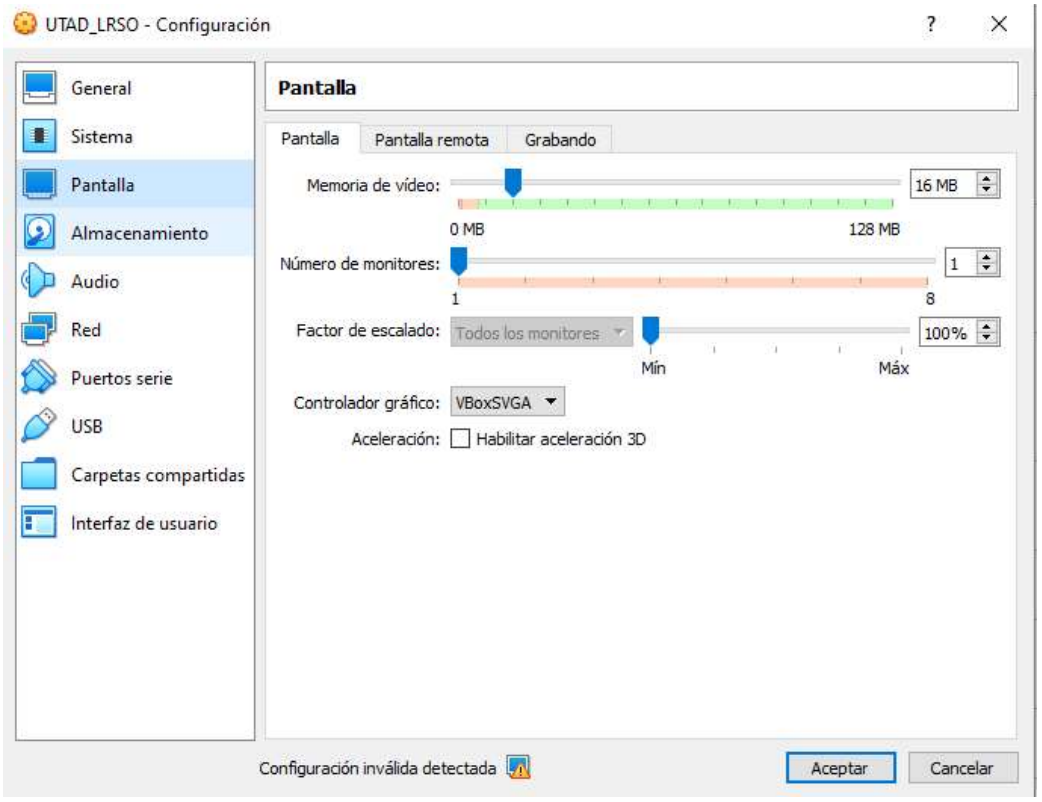
En este punto, deberías tener, Virtual Box instalado y creada sobre este una VM con una configuración para una distro de Linux Ubuntu de 64 bits.

Antes de continuar, vamos a cambiar el controlador gráfico de la máquina virtual para que las dimensiones de la pantalla de la VM sean acorde con nuestra pantalla disponible del equipo.

Para realizar estos pasos procedemos de la siguiente manera: Botón derecho sobre la máquina virtual creada y seleccionamos el menú “Configuración”.



Una vez dentro de esta ventana, seleccionamos la opción de la izquierda "Pantalla" y en la parte derecha, en la opción "Controlador Gráfico", de la lista desplegable seleccionamos "VBoxSVGA".



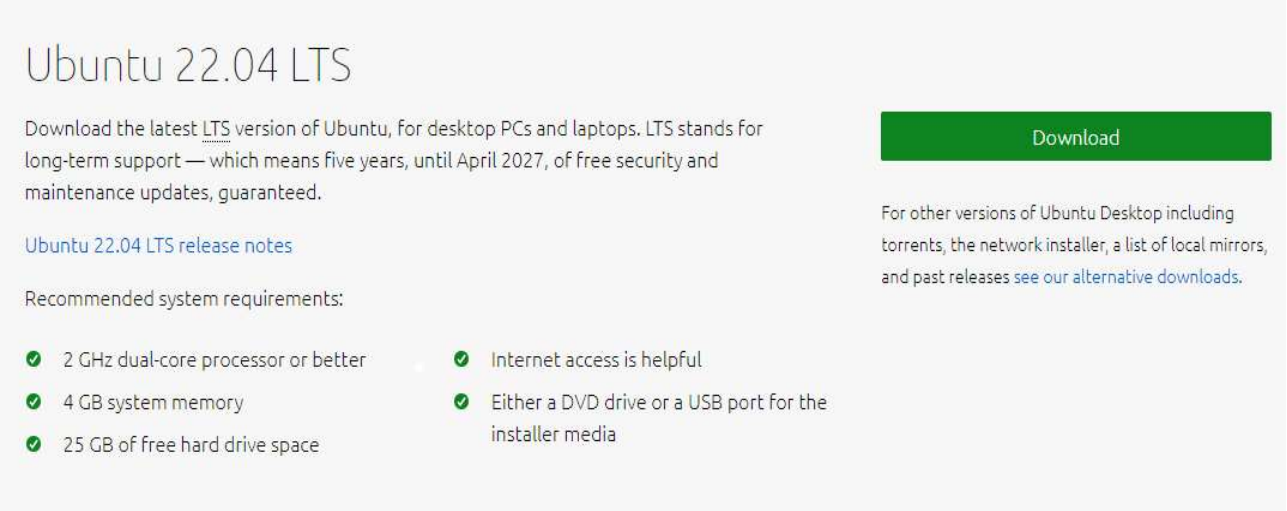
Y pulsamos sobre el botón “Aceptar”.

NOTA: Dependiendo de las características del equipo y la configuración del mismo, os puede dar un mensaje “Configuración inválida detectada”, no pasa nada, pulsad “Aceptar”, ya que esto se debe a chequeos internos de compatibilidad entre la versión que instalemos de VBox y configuración de BIOS que tengamos en el equipo.

5.4 Instalación de Ubuntu sobre una máquina virtual de Virtual Box

Antes de proceder a arrancar la VM, tenemos que descargar la versión de Ubuntu que vamos a utilizar, para ello nos vamos:

- Página oficial de Ubuntu: **Ubuntu.com**
- Pulsamos sobre el link “Ubuntu Desktop” (Ubuntu versión escritorio)

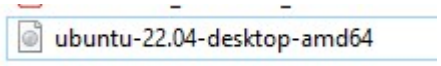


- Pulsamos sobre el botón “Download”
- Se nos comenzará a descargar la última versión disponible. En estos momentos, la última versión disponible para descargar es la versión 22.04 (Si en el momento de trabajar con esta UD existe otra versión disponible más actual, no pasará nada, nos bajamos la última disponible ya que para las tareas que tengamos que realizar son válidas).

LTS: Significa Long Term Support, es decir, que durante 5 años tendrá actualizaciones y mantenimiento, actualizaciones de seguridad,….

En la imagen anterior, se presentar los requisitos recomendamos, pero son para una máquina física, siendo para máquina virtual como es el caso que nos ocupa, algo menores, ya que el objetivo nuestro es que sea un entorno didáctico no un entorno real de producción.

Una vez descargado el software (.iso) tendremos en nuestra zona de descargas el fichero:



A continuación, vamos a ver un video de cómo proceder con la instalación y configuración del sistema operativo Linux en su versión de Ubuntu.

Una vez que el sistema operativo se encuentre instalado y configurado tendremos la máquina virtual disponible para trabajar con ella.

Video 3 de ayuda para la instalación: **INSO_LRSO_Instalacion_Conf_SO.mp4**