TEMA 4. DESPLIEGUE DE APLICACIONES

Contenido

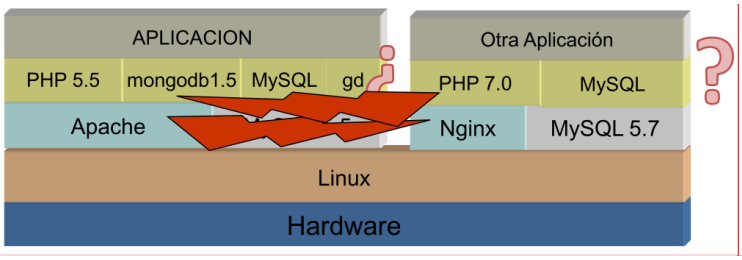
DIFICULTADES DEL DESPLIEGUE

Las aplicaciones actuales difícilmente constan de un simple ejecutable. Normalmente constan de uno o más ejecutables, que dependen de:

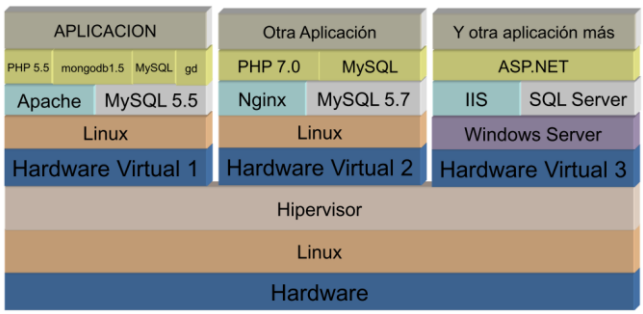
- **API** sistema operativo
- Máquina virtual o intérprete de lenguaje de programación (Java, Python…)
- Plataforma o conjunto de bibliotecas (.NET, archivos .DLL…)
- Servidor web (Apache, Nginx, Microsoft IIS…)

El **problema al desplegar** una aplicación es que esta debe instalarse acompañada de la infraestructura de la que depende (entorno). Al instalar varias aplicaciones en un mismo sistema pueden aparecer conflictos entre sus dependencias.

Ejemplo: aplicación web implementada en PHP, requiere Apache (con módulo PHP 5.5), guarda su estado en una base de Datos MySQL 5.5, utiliza las extensiones de PHP mongodb1.5, gd y mysql



Una **posible solución** es el despliegue mediante máquinas virtuales. Así para cada aplicación se crearía una máquina virtual sobre la que se instalaría el SO requerido (o que mejor se adapte) y el entorno de ejecución de la aplicación. De esta forma, cada aplicación se ejecuta con una máquina dedicada solo a ella, aislamos las aplicaciones. No hay por tanto ningún tipo de conflicto entre los entornos de ejecución de estas. De hecho, incluso se pueden ejecutar aplicaciones que se ejecuten sobre SO diferentes.



Sin embargo, con el uso de despliegue mediante máquinas virtuales, estaríamos replicando diferentes SO en máquinas virtuales, algo que no es necesario y que elevaría el coste de las licencias del SO invitado. Para solucionar este nuevo inconveniente surge el concepto de **contenedor**.

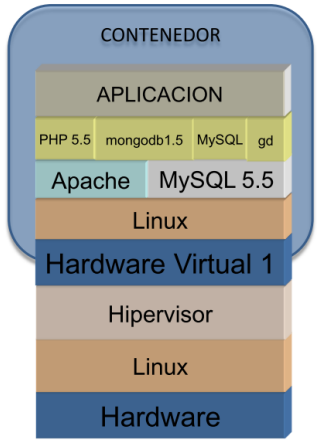
Comentado [id1]: Apache, junto a PHP 5.5, MySQL 5.5, y las extensiones mongodb 1.5, MySQL y gd para apache constituyen el entorno de la aplicación. Ahora bien, ¿Y si quisieramos instalar en el Sistema también otra aplicación que en lugar de Apache require Nginx, con MySQL 5?7, PHP 7.0 con su extension MySQL?
¿Seguro que no habría problema en hacer funcionar juntos ambos servidores web? ¿Y a ambas versions de PHP y de MySQL? Se debería respetar la compatibilidad ascendente… pero… ¿y si no?

Comentado [id2]: Realmente, no es necesario virtualizar todo el hardware ¿Teclado? ¿Pantalla? ¿Dispositivos de E/S?... Ni tampoco necesitaremos el SO completo ¿aplicaciones de ayuda? ¿reproductores multimedia? ¿interfaz gráfica de usuario? ¿herramientas del Sistema?

CONTENEDORES

En realidad, sólo necesitamos…

- Sistema de archivos para alojar archivos del entorno de ejecución, programas y datos
- Interfaz de conexión a la red
- Núcleo del sistema operativo



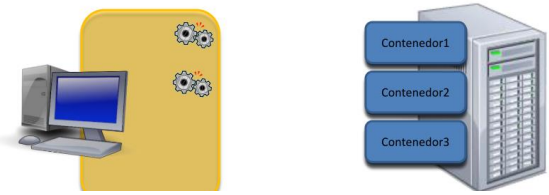
Los **contenedores** se almacenan en un repositorio. Las máquinas, que van a ejecutar las aplicaciones, ejecutan una aplicación que permite:

- Descargar, instalar y gestionar contenedores
- Instanciar (ejecutar) contenedores. Cada contenedor se ejecuta de manera totalmente aislada

Un contenedor es una aplicación totalmente autocontenida. Es decir, es equivalente a un ejecutable tan solo que a un nivel de abstracción más alto. Un **contenedor** es una aplicación completa junto con todo el entorno que necesita para su funcionamiento. Se utiliza para aislar nuestro entorno de ejecución (archivos necesarios para que el programa funcione), programas y datos.

Los contenedores están pensados para ser reutilizados: podemos intercambiar aplicaciones dentro de un mismo contenedor.

El contenedor se despliega en un servidor que actúa de almacén: en él se guardan todos los contenedores que se van a utilizar.



Ejemplos de tecnología de contenedores:

Comentado [id3]: Un **ejecutable** (archivo .exe) es una aplicación, o parte de ella

Comentado [id4]: Un contenedor es una aplicación sin intermediario con el SO, un espacio aislado que ofrece el SO para la ejecución de uno o más procesos

Contenedor: son una tecnica de virtualizacion a nivel del SO SO ofrece servicios para crear un "sandbox" en que el lanzar la pila de aplicaciones

La **escabilidad del contenedor** es del orden de cientos de miles / millones de contenedores sobre bare metal:

- bare metal + contenedor = 1000000 millon contenedor
- bare metal + MV virtual (con paravirtualizacion, con extensiones VT del procesador, con el IOMMU y extensions SR-IOV de los dispositivos de E/S)= 100

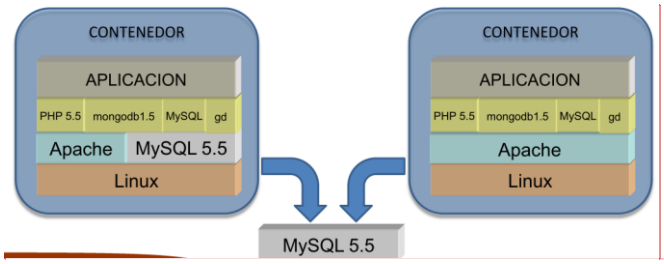
- **Docker**: Solución comercial de amplia implantación. Se integra con Kubernetes
- **Podman**: Tecnología compatible con Docker (también basado en OCI). También se integra con Kubernetes. Utiliza misma CLI que Docker
- **Linux Containers**: Solución nativa Linux
- **RKT**: Hipervisor de contenedores de CoreOS. Se integra con Kubernetes. Capaz de ejecutar contenedores OCI
- **Singularity**: Platforma orientada a High Performance Computing. Capaz de importar contenedores Docker
- **Solaris Containers:** Solución nativa para sistema operativo Solaris

ESCALABILIDAD Y TOLERANCIA A FALLOS MEDIANTE CONTENEDORES

Escalabilidiad: propiedad deseable de un sistema que indica su habilidad para reaccionar y adaptarse a los cambios -> la aplicación siempre está expuesta a continuos cambios.

Tolerancia a fallos: mecanismo para garantizar el correcto funcionamiento de los servicios aún en presencia de fallo. Detectamos el errar y, tras ello, desplegamos mecanismos de recuperación.

- **Replicar aplicación sin estado (contenido estático):** basta replicar contenedor
- **Replicar aplicación con estado (contenido va cambiando):** sacamos el estado fuera de contenedor, replicamos contenedor y conectamos ambos conectores al estado



Para escalar la aplicación, necesitamos que la aplicación no tenga estado, o bien que el estado de la aplicación (es decir, los datos que gestiona la aplicación) no estén en el contenedor, sino que se almacenen de manera externa. El contenedor de esta manera solo contiene la lógica. De esta forma, si necesitamos duplicar la capacidad de servicio de la aplicación, duplicamos el contenedor.

La capacidad de procesamiento del sistema se puede adaptar dinámicamente aumentando o disminuyendo el número de instancias en ejecución del contenedor.

Tolerancia a fallos: si cae una instancia de un contenedor, se lanza otra

Comentado [id5]: La **escalabilidad** nos dice cuanto podemos crecer en número de MV en un sistema y en número de contenedores, hace referencia al número de máquinas virtuales que puede tener un bare metal

PREGUNTA EXAMEN
¿qué ofrece mayor escalabilidad un contenedor o una máquina virtual?
Contenedor, pero, de forma general, no da fiabilidad

MV ofrece mayor aislamiento que un contenedor (seguridad), **MV son más fiables.**
Si la aplicación falla, el contenedor me aumenta la fiabilidad del sistema. Si lo que falla es el SO con contenedores entonces tenemos un problema, todos los contenedores se pierden. Luego, **los contenedores solo nos ofrecen más fiabilidad a nivel aplicativo**

Comentado [id6]: Linux está fuera del contenedor

Si MV2 (son procesos pero que simular todo el HW de la MV) entonces un fallo en la maquina implica el fin de ejecución de proceso

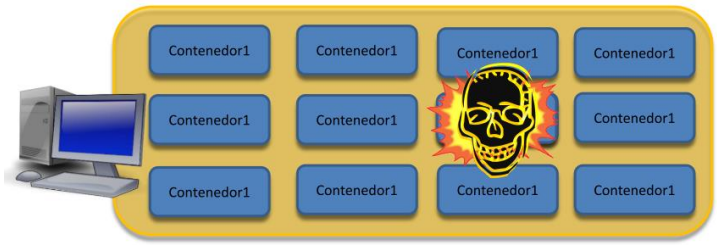
Un fallo en la ejecución del SO1 que se ejecuta en la MV1, la MV1 acaba en una situación de error fatal y hay que volver a lanzar la MV1 para que arranque de nuevo el SO1

En los contenedores no es así. Como son técnicas virtualizacion a nivel de SO, si hay un cuelgue (un error de programacion) en el SO (afitrion) se ejecuta el fin de la ejecucion de la ejecucion de todos los contenedores

Un contenedor es un proceso de partida y a partir de ahi se van lanzando más procesos que hacen llamadas al sistema que van directamente al SO afitrion (no hay simulación de HW)

Una llamada al sistema de un contenedor cuando hay un cuelgue significa que ha alcanzado una condición de error dentro del SO anfitrión
Si mi SO anfitrión tiene un bug (fallo de programación) todos los contenedores que se están ejecutando en ese sistema, se pierde su ejecución, hay que reiniciar el sistema
Con una MV no es así. Si la MV entra en una condición de error se reinicia esa MV y continuo sin problema

Los procesos del contenedor usan las llamadas al sistema del SO anfitrion



Inicialmente se puede crear un número de instancias del contenedor de acuerdo con una carga inicial de trabajo estimada. Si la carga de trabajo es mayor, se pueden crear más instancias para escalar el Sistema a la nueva situación. Si la carga de trabajo disminuye, se pueden eliminar instancias para escalar a la nueva carga de trabajo. Incluso, si por algún motivo una instancia falla, se crea una nueva instancia que la sustituye.

PET vs CATTLE

Las aplicaciones tradicionales son como mascotas (PET): si tienen un fallo nos preocupamos por solucionarlo inmediatamente, si fallan el mundo se para para nosotros… con el enfoque propuesto, las aplicaciones se convierten en ganado (CATTLE): si una instancia falla se sustituye por otra, si tienen un problema, pero se puede cambiar por otra que funcione, la cambiamos y no nos preocupamos.

SERVICIOS EN LA NUBE

Una organización necesita una solución informática.

Opción 1: bare metal (Housing)

- Organización adquiere hardware
- Organización instala y administra SO (actualizaciones, copias de seguridad, etc…)
- Organización instala y administra plataforma de desarrollo
- Organización desarrolla y mantiene aplicación

Opción 2: Hardware as a Service (HaaS)

- Organización alquila hardware (en instalaciones proveedor)
- Organización instala y administra SO (posibilidad de contratar este servicio con proveedor)
- Organización instala y administra plataforma de desarrollo
- Organización desarrolla y mantiene aplicación

Opción 3: Infraestructure as a Service (IaaS)

- Organización alquila hardware virtual (ubicación física irrelevante): Hardware virtual es escalable dinámicamente (organización paga por lo que usa), hay posibilidad de alquilar múltiples máquinas conectadas en red virtual y de alquilar espacio de almacenamiento en la nube
- Organización instala y administra SO y plataforma de desarrollo
- Organización desarrolla y mantiene aplicación

Comentado [id7]: Las operaciones rutinarias de mantenimiento, así como el Sistema operativo y su instalación suelen ser contratadas con el proveedor del servicio

Comentado [id8]: También cabe la posibilidad de que las opciones básicas de mantenimiento sean realizadas por proveedor. Este al menos garantizará la existencia de copias de seguridad
El hardware virtualizado puede crecer y decrecer de manera dinámica, permitiendo la posibilidad de que pago mediante una tarifa por tiempo de uso de recursos
Se puede contratar la infraestructura complete, es decir, se puede incluso contratar un conjunto de máquinas virtuales, interconectadas en una red virtual privada

Opción 4: Platform as a Service (PaaS)

- Organización alquila hardware virtual con el entorno necesario para ejecutar su aplicación
- Proveedor instala y administra SO
- Proveedor instala y administra plataforma
- Organización desarrolla y mantiene aplicación
- Caso particular de plataforma: repositorio de contenedores

Opción 5 Software as a Service (SaaS)

- Organización alquila la solución informática que necesita: el proveedor instala y administra SO y plataforma. Además, el proveedor desarrolla y mantiene aplicación.
- Organización solo explota la solución alquilada.

Ejemplo de Sistemas Operativos para la nube: OpenStack

Ejemplo de nubes: Amazon Web Service, Google Cloud, Microsoft Azure, IBM BlueMix

Comentado [id9]: El hardware e incluso el Sistema operativo sobre el que se ejecuta la plataforma de la aplicación es irrelevante