Contenido

[1. Autores del trabajo, planificación y entrega 3](#_Toc479021915)

[1.1 Autores 3](#_Toc479021916)

[1.2 Planificación 3](#_Toc479021917)

[1.3 Entrega 3](#_Toc479021918)

[2. Descripción de las tecnologías 4](#_Toc479021919)

[2.1 OpenVZ 4](#_Toc479021920)

[2.2 Docker 8](#_Toc479021921)

[3. Criterios de comparación 11](#_Toc479021922)

[3.1. Categoría A: Generales 11](#_Toc479021923)

[3.1.1. Criterio A.1: Soporte y documentación 11](#_Toc479021924)

[3.1.2. Criterio A.2: Usabilidad. 11](#_Toc479021925)

[3.1.3. Criterio A.3: Rendimiento 11](#_Toc479021926)

[3.1.4. Criterio A.4: Resultados 11](#_Toc479021927)

[3.1.5. Criterio A.5: Última actualización 11](#_Toc479021928)

[3.2. Categoría B: Usuarios 12](#_Toc479021929)

[3.2.1. Criterio B.1: Usuario Gratuito 12](#_Toc479021930)

[3.3 Categoría C: Modelo de Negocio 12](#_Toc479021931)

[3.3.1. Criterio C.1: Costes 12](#_Toc479021932)

[3.4 Categoría D: Características 12](#_Toc479021933)

[3.4.1 Criterio D.1: Sistema operativo 12](#_Toc479021934)

[3.4.2 Criterio D.2: Aislamiento de sistema de ficheros 12](#_Toc479021935)

[3.4.3 Criterio D.3: Cuotas de disco 12](#_Toc479021936)

[3.4.4 Criterio D.4: Limite de uso de memoria 13](#_Toc479021937)

[3.4.5 Criterio D.5: Limite de uso de CPU 13](#_Toc479021938)

[3.4.6 Criterio D.6: Puntos de control de particiones 13](#_Toc479021939)

[3.4.7 Criterio D.7: Privilegios de administración 13](#_Toc479021940)

[3.5. Categoría E: Generales 14](#_Toc479021941)

[3.5.1. Criterio E.1: Autor de la herramienta. 14](#_Toc479021942)

[3.5.2. Criterio E.2: Fecha de creación. 14](#_Toc479021943)

[3.5.3. Criterio E.3: Redes Sociales 14](#_Toc479021944)

[3.6. Categoría F: Rendimiento 14](#_Toc479021945)

[3.6.1. Criterio F.1: Número de kernels en los que funciona. 14](#_Toc479021946)

[3.6.2. Criterio F.2: Capacidad de contención. 14](#_Toc479021947)

[3.6.3. Criterio F.3: Necesidad de extensión IP. 15](#_Toc479021948)

[3.6.4. Criterio F.4: Funcional dentro del otro. 15](#_Toc479021949)

[4. Evaluación de los criterios por tecnología 15](#_Toc479021950)

[4.1 Evaluación de los criterios para la tecnología Docker 15](#_Toc479021951)

[4.2 Evaluación de los criterios para la tecnología OpenVZ 16](#_Toc479021952)

[5. Comparación de las tecnologías 17](#_Toc479021953)

[6. Recomendaciones 20](#_Toc479021954)

[6.1 Situación 1 20](#_Toc479021955)

[6.1.1 Descripción de la situación 20](#_Toc479021956)

[6.1.2 Recomendación de tecnología a utilizar 20](#_Toc479021957)

[6.2 Situación 2 22](#_Toc479021958)

[6.2.1 Descripción de la situación 22](#_Toc479021959)

[6.2.2 Recomendación de tecnología a utilizar 22](#_Toc479021960)

# 1. Autores del trabajo, planificación y entrega

## 1.1 Autores

Grupo M4

* Diego Cárdenas Cuadrado (Coordinador)
* Adrián Blanco Domínguez
* Alejandro Martínez Pantín
* Marcos Rodríguez Castillo
* Roberto Sánchez Leal

## 1.2 Planificación

[Enlace planificación GanttPro](https://app.ganttpro.com/shared/token/15957a85a42133e47210c13c34c9917d2e19765437b506b44fd1cbedd3b3b25a)

## 

## 1.3 Entrega

[Enlace repositorio GitHub](https://github.com/DiegoUAH/TG2)

# 2. Descripción de las tecnologías

## 2.1 OpenVZ

Es una tecnología de virtualización a nivel de sistema operativo para Linux. Permite que un servidor físico ejecute múltiples instancias de sistemas operativos aislados, conocidos como Servidores Privados Virtuales (VPS), Entornos Virtuales (EV) o Contenedores.

Mientras que las tecnologías de virtualización como VMware y Xen proporcionan una virtualización completa y pueden ejecutar múltiples sistemas operativos y versiones diferentes de kernel, en OpenVZ tanto el servidor anfitrión como los VPS deben ser Linux, aunque las distribuciones de GNU/Linux pueden ser diferentes en cada Contenedor. Sin embargo, la virtualización a nivel de sistema operativo de OpenVZ proporciona mejor rendimiento, escalabilidad, densidad, administración de recursos dinámicos, y facilidad de administración que las máquinas virtuales o que otras tecnologías de virtualización de servidores.

OpenVZ es la base de Virtuozzo (para Linux) que es un software comercial desarrollado por SWsoft Inc. OpenVZ es un producto de software libre y licenciado bajo los términos de la licencia GNU GPL versión 2.

OpenVZ consiste del **núcleo** y de **herramientas a nivel de usuario**.

**Núcleo:**

El núcleo de OpenVZ es un núcleo Linux modificado que agrega soporte para contenedores. El núcleo modificado proporciona virtualización, aislamiento, administración de recursos y Puntos de comprobación.

**Virtualización y aislamiento:**

Cada contenedor es una entidad separada, y se comporta en gran medida como lo haría un servidor físico real. De manera que tiene sus propios archivos, usuarios y grupos, árbol de procesos, red, dispositivos y objetos de comunicación entre procesos.

**Administración de recursos:**

Como todos los contenedores usan el mismo kernel, la administración de recursos es de suprema importancia. Con la administración de recursos de OpenVZ se consigue que cada contenedor permanezca dentro de sus límites y no afecte al resto.

La administración de recursos de OpenVZ se compone de cuatro componentes: cuota de disco de dos niveles, planificador de CPU justo, planificador de E/S de disco y monitor de usuarios. Estos recursos pueden cambiarse durante el tiempo de ejecución de un contenedor sin necesidad de reiniciar el sistema.

*Cuota de disco de dos niveles:*

El dueño (root) del sistema anfitrión (OpenVZ) puede configurar la cuota de disco por contenedor, en términos de bloques de disco e inodos (aproximadamente igual al número de archivos). Éste es el primer nivel de cuota de disco. También, un dueño de contenedor (root) puede usar las herramientas usuales de cuota dentro de su propio contenedor para definir cuotas de disco estándar de UNIX por usuario y por grupo.

Si se desea ampliar el espacio a un contenedor, solamente hay que incrementar la cuota de dsico. No es necesario redimensionar las particiones de disco, etc.

*Planificador de CPU:*

El planificador de CPU en OpenVZ es una implementación de dos niveles de la estrategia de programación de fair-share (cuota justa).

En el primer nivel, el planificador decide a qué contenedor debe asignar la división de tiempo de la CPU, en función de los valores de las cpuunits por contenedor. En el segundo nivel, el planificador de Linux estándar decide qué proceso ejecutar en ese contenedor, utilizando las prioridades de proceso estándar de Linux.

Es posible establecer diferentes valores para las CPUs en cada contenedor. El tiempo real de la CPU se distribuirá proporcionalmente a estos valores.

Además de lo anterior, OpenVZ proporciona formas de:

* Establecer límites estrictos de la CPU, como el 10% del tiempo total de la CPU.
* Limitar el número de núcleos de CPU disponibles para el contenedor.
* Enlazar un contenedor a un conjunto específico de CPUs.

*Planificador de E/S:*

Al igual que el planificador de CPU, el planificador de E/S en OpenVZ también tiene dos niveles, utilizando el programador de E/S CFA de Jens Axboe en su segundo nivel.

A cada contenedor se le asigna una prioridad de E/S y el planificador distribuye el ancho de banda de E/S disponible de acuerdo con las prioridades asignadas. Por lo tanto, ningún contenedor único puede saturar un canal de E/S.

*Monitor de usuarios:*

El monitor de usuarios es un grupo de contadores por Contenedor, límites, y garantías. Hay un conjunto de alrededor de 20 parámetros que se eligen cuidadosamente para cubrir todos los aspectos de la operación de Contenedor, de manera que ningún Contenedor por sí solo pueda abusar de cualquier recurso el cual es limitado por todo el nodo y así hacer daño a otros EVs.

Los recursos contabilizados y controlados son principalmente memoria y objetos en el kernel tales como los segmentos de memoria compartidos IPC, buffers de red etc. Cada recurso tiene cinco valores asociados con este: uso actual, uso máximo (por el tiempo de vida de un EV), contador de barrera, de límite y de falla. El significado de barrera y límite es dependiente del parámetro. Si cualquier recurso alcanza el contador de límite o de falla, entonces el dueño del EV puede ver si algo malo está pasando.

**Punto de comprobación y migración:**

La característica de la migración en vivo y punto de comprobación se liberó para OpenVZ a mediados de abril de 2006. Esta permite migrar un EV desde un servidor físico a otro sin necesidad de apagar/reiniciar un EV. El proceso se conoce como punto de comprobación: un EV se congela y todo su estado se guarda en un archivo en disco. Este archivo puede ser transferido a otra máquina y un EV puede descongelarse (restaurarse) allí. La demora es de unos pocos segundos.

Dado que cada pieza de estado de EV (incluyendo conexiones de red abiertas) se guarda, desde la perspectiva del usuario parece una demora en la respuesta: una transacción de base de datos toma más tiempo que el usual, cuando continúa como normal el usuario no nota que su base de datos está ya corriendo en otra máquina. Esta característica hace posible escenarios tales como actualizar un servidor sin necesidad de reiniciarlo: si la base de datos necesita más memoria o recursos de CPU, sencillamente se debe comprar un servidor mejor y más nuevo y migrar en vivo el EV a éste, migrar todos los EVs a otro, apagarlo, agregar memoria, arrancarlo de nuevo y migrar todos los EVs de nuevo.

**Herramientas en el nivel del usuario:**

OpenVZ viene con herramientas de línea de comandos para administrar Contenedores (vzctl), además de herramientas para administrar software en Contenedores (vzpkg).

**vzctl:** Esta es una simple herramienta de línea de comandos de alto nivel para administrar un Contenedor

Las plantillas son imágenes precreadas que se usan para crear un nuevo EV. Estas plantillas permiten crear contenedores en pocos segundos.

**Características distintivas:**

**Escalabilidad:**

Ya que OpenVZ emplea un modelo de kernel único, es tan escalable como kernel Linux 2.6, lo que significa que soporta hasta 64 CPUs y hasta 64 GiB de RAM. Un único contenedor se puede escalar hasta el equipo físico entero, usando todos los CPUs y toda la RAM.

De hecho, algunas personas están usando OpenVZ con único Entorno Virtual. Esto es extraño a primera vista, pero dado el hecho de que un EV único usa todo los recursos del hardware con rendimiento nativo, y tiene agregados beneficios tales como independencia del hardware, administración de recursos y migración en vivo, esto es una opción obvia en muchos escenarios.

**Densidad:**

OpenVZ es capaz de alojar cientos de Entornos Virtuales en hardware decente. Las principales limitaciones son RAM y CPU.

**Administración masiva:**

Un propietario (root) de un servidor físico OpenVZ (también conocido como Nodo de Hardware) puede ver todos los procesos y archivos de EV. Esto hace la administración masiva de escenarios posible. Considerar que VMware o Xen se usan para consolidación de servidores: para aplicar una actualización de seguridad a unos 10 servidores virtuales se debe iniciar una sesión en cada uno y ejecutar el procedimiento de actualización - el mismo que se haría con diez servidores físicos.

En el caso de OpenVZ, se puede ejecutar un simple script de intérprete de comandos que actualice todo (o sólo algunos seleccionados) EVs a la vez.

**Escenarios de uso:**

Los siguientes escenarios de uso son comunes para todas las tecnologías de virtualización. Sin embargo, una característica única de la virtualización en el nivel de SO como OpenVZ es que no se tiene que gastar gran cantidad de tiempo de procesamiento por la virtualización, esto hace los escenarios más atractivos.

**Seguridad:**

Se pone cada servicio de red (como Apache, Servidor de Correo, Servidor DNS etc.) en un Entorno Virtual separado. En caso de que un cracker encuentre un agujero en la seguridad de uno de las aplicaciones y entre, todo lo que puede hacer es en ese mismo servicio; dado que todos los otros servicios están en EVs separados no puede acceder a ellos.

**Consolidación de Servidores:**

Actualmente, la mayoría de los servidores están infra-utilizados. Usando OpenVZ, tales servidores se pueden consolidar migrándolos a Entornos Virtuales. Se ahorra en espacio de racks, consumo de electricidad, y esfuerzo de administración.

**Hosting:**

Aparentemente, la virtualización en el nivel de SO es el único tipo de virtualización que las empresas de hosting pueden pagar y usar para vender Entornos Virtuales baratos a sus clientes. Notar que cada EV tiene acceso completo de root, lo que significa que el dueño del EV puede reinstalar cualquier cosa, y aún usar cosas tales como iptables (reglas de cortafuego).

**Desarrollo y Pruebas:**

Usualmente los desarrolladores y testers necesitan acceder a un grupo de distribuciones de Linux, y necesitan reinstalarlas desde cero con frecuencia. Con OpenVZ, pueden tener todo en un sólo equipo, sin ninguna necesidad de reiniciar, con rendimiento nativo, y un nuevo EV se puede crear en sólo un minuto. Clonar un VE es también muy simple: solamente se necesita copiar el área del EV y el archivo de configuración.

**Educativo:**

Varios alumnos pueden compartir un EV. Cada uno puede trabajar con una diferente distribución de Linux. Un nuevo EV se puede (re)crear en un minuto.

## 2.2 Docker

Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización a nivel de sistema operativo en Linux. Docker utiliza características de aislamiento de recursos del kernel de Linux, tales como cgroups y espacios de nombres (namespaces) para permitir que "contenedores" independientes se ejecuten dentro de una sola instancia de Linux, evitando la sobrecarga de iniciar y mantener máquinas virtuales.

El soporte del kernel de Linux para los espacios de nombres aísla de vista, en su mayoría, una aplicación del entorno operativo, incluyendo árboles de proceso, red, ID de usuario y sistemas de archivos montados, mientras que los cgroups del kernel proporcionan aislamiento de recursos, incluyendo la CPU, la memoria, el bloque de E/S y de la red. Desde la versión 0.9, Docker incluye la biblioteca libcontainer como su propia manera de utilizar directamente las facilidades de virtualización que ofrece el kernel de Linux, además de utilizar las interfaces abstraídas de virtualización mediante libvirt, LXC (Linux Containers) y systemd-nspawn.

Otro de los puntos fuertes de Docker es que Docker Engine (el servidor de Docker encargado de ejecutar los contenedores), por defecto es capaz de conectarse a Docker Hub (registro de imágenes público) donde se encuentran disponibles miles de imágenes ya creadas y descargables a demanda. Un análisis en mayo de 2015 mostró las siguientes organizaciones como las principales contribuyentes de Docker: Red Hat (mayores contribuyentes, aún más que el equipo de Docker en sí), el equipo de Docker, IBM, Google y Cisco Systems.

También es posible crear un registro privado de imágenes donde se almacenarían las imágenes asociadas a los servicios desarrollados por una compañía.

Los contenedores de Docker aportan muchos beneficios a los desarrolladores y administradores de sistemas, es un cambio de filosofía radical que requiere la implicación de las dos partes. Simplifica enormemente el ciclo de vida completo de la aplicación así como el despliegue entre los diferentes entornos.

Los contenedores están destinados a ser completamente estandarizados. Esto significa que el contenedor se conecta al host y a cualquier lado fuera del contenedor usando interfaces definidas. Una aplicación contenerizada no se debe involucrar ni preocupar sobre los detalles de los recursos o arquitectura del host. Esto simplifica los supuestos del desarrollo sobre el ambiente operativo. Del mismo modo, que el host, cada contenedor es una caja negra. No se preocupan de los detalles de la aplicación en el interior.

Uno de los beneficios de la abstracción entre el sistema host y los contenedores es eso, a partir de un diseño correcto de aplicación, escalar puede ser simple y directo. Un diseño orientado a servicio combinado con aplicaciones contenerizadas proporcionan las bases para una fácil escalabilidad. Un developer puede ejecutar algunos contenedores en su estación de trabajo, mientras que este sistema puede escalar horizontalmente en ensayo o área de pruebas. Cuando los contenedores suben a producción, estos pueden escalar nuevamente.

Los contenedores permiten al desarrollador agrupar una aplicación o un componente de la misma junto a todas sus dependencias como unidad. El sistema host no tiene que preocuparse por las dependencias necesarias para ejecutar una aplicación específica. Mientras pueda correr Docker, será capaz de correr los contenedores de Docker. Esto hace que la administración de dependencias sea fácil y simplifica la administración de versiones de la aplicación. Los sistemas host y los equipos de operaciones dejan de ser responsables de administrar las dependencias necesarias para la aplicación ya que deben estar contenidas dentro del propio contenedor.

Algunas ventajas de Docker:

* Las instancias se arrancan en pocos segundos.
* Es fácil de automatizar e implantar en entornos de integración continua.
* Existen multitud de imágenes que pueden descargarse y modificarse libremente.
* Ambientes Replicables.

Como inconvenientes cabe destacar:

* Sólo puede usarse de forma nativa en entornos Unix aunque se puede virtualizar gracias a boot2docker tanto en OSX como en Windows.
* Las imágenes sólo pueden estar basadas en versiones de Linux modernas (kernel 3.8 mínimo).
* Como es relativamente nuevo, puede haber errores de código entre versiones.

**Dockerfiles:**

Los archivos de Docker (Dockerfiles) permiten a los usuarios definir las acciones exactas necesarias para crear una nueva imagen de contenedor. Esto te permite escribir la ejecución de tu ambiente como si fuera código, almacenándola en un controlador de versiones si es necesario. El mismo archivo de Docker en el mismo entorno siempre producirá una imagen de contenedor idéntica.

Debido a que es posible crear imágenes de contenedores a través de un proceso iterativo, es muy frecuente agregar los pasos de configuración dentro de un Dockerfile para que se conozcan los pasos necesarios. Los Dockerfiles son simples archivos de compilación que describen como crear la imagen del contenedor desde un punto de partida conocido.

Beneficios de los Dockerfiles:

*Fácil versionamiento*: Los Dockerfiles por si solos pueden ser rastreables a través de un controlador de versiones para comparar sus cambios y revertir cualquier error.

*Predictibilidad*: Crear una imagen desde un Dockerfile ayuda a reducir el error humano desde el proceso de creación de la imagen.

*Trazabilidad*: Si tu planeas compartir tus imágenes, es una buena idea proporcionar el Dockerfile que crea una imagen como una forma para que otros usuarios puedan auditar el proceso. Esto básicamente proporciona el historial de la línea de comandos con los pasos tomados para crear la imagen.

*Flexibilidad*: Crear imágenes desde Dockerfile permite sobrescribir los valores predeterminados que las compilaciones proporcionan. Esto significa que no tienes que proporcionar las opciones de tiempo de arranque para que la imagen funcione según lo previsto.

**Arquitectura de aplicaciones contenerizadas:**

Cuando se diseñan aplicaciones para ser implementadas en contenedores, una de las principales áreas de preocupación es la actual arquitectura de la aplicación.

Generalmente, las aplicaciones contenerizadas funcionan mejor cuando se implementan en un diseño orientado a servicio.

Las aplicaciones orientadas a servicio dividen la funcionalidad del sistema en componentes discretos que se comunican unos con los otros a través de interfaces bien definidas. La tecnología de contenedores por si misma se encarga de este tipo de diseño debido a que permite a cada componente escalar o crecer de manera independiente.

# 3. Criterios de comparación

**3.1. Categoría A: Generales**

***3.1.1. Criterio A.1: Soporte y documentación***

*Nombre del criterio: Soporte y documentación*.

*Descripción:* Facilidad de un usuario para acceder a contenidos adicionales.

*Tipo de valor:* Texto libre.

***3.1.2. Criterio A.2: Usabilidad.***

*Nombre del criterio:* Usabilidad.

*Descripción: Facilidad de uso de la herramienta*.

*Tipo de valor:* Boolean

***3.1.3. Criterio A.3: Rendimiento***

*Nombre del criterio: Rendimiento*.

*Descripción:* Utilidad del producto.

*Tipo de valor:* Texto libre.

***3.1.4. Criterio A.4: Resultados***

*Nombre del criterio: Resultados*.

*Descripción:* Resultados que han obtenido las tecnologías.

*Tipo de valor:* Texto libre.

***3.1.5. Criterio A.5: Última actualización***

*Nombre del criterio: Última actualización*

*Descripción:* Fecha de la última actualización

*Tipo de valor:* Fecha.

**3.2. Categoría B: Usuarios**

***3.2.1. Criterio B.1: Usuario Gratuito***

*Nombre del criterio: Usuario Gratuito*

*Descripción:* posibilidades a las que aspira un usuario demo.

*Tipo de valor:* Texto libre.

**3.3 Categoría C: Modelo de Negocio**

***3.3.1. Criterio C.1: Costes***

*Nombre del criterio: Costes*.

*Descripción:* Coste de disponibilidad de la herramienta.

*Tipo de valor:* Texto libre y numérico.

## 3.4 Categoría D: Características

### **3.4.1 Criterio D.1: Sistema operativo**

*Nombre del criterio: Sistema operativo.*

*Descripción: Opción especifica en que sistemas operativos se puede trabajar con las diferentes herramientas.*

*Tipo de valor: Texto libre.*

### **3.4.2 Criterio D.2: Aislamiento de sistema de ficheros**

*Nombre del criterio: Aislamiento de ficheros.*

*Descripción: Opción que permite ofrecer seguridad a nivel de hardware y aislar los fallos de los diferentes paquetes o máquinas virtuales.*

*Tipo de valor: Booleano (Si/No).*

### **3.4.3 Criterio D.3: Cuotas de disco**

*Nombre del criterio: Cuotas de disco.*

*Descripción: Opción que permite poner cuotas de disco a los paquetes o máquinas virtuales.*

*Tipo de valor: Booleano (Si/No).*

### **3.4.4 Criterio D.4: Limite de uso de memoria**

*Nombre del criterio: Limite de uso de memoria.*

*Descripción: Opción que permite especificar el porcentaje de uso de la memoria por parte de los paquetes o máquinas virtuales.*

*Tipo de valor: Booleano (Si/No).*

### **3.4.5 Criterio D.5: Limite de uso de CPU**

*Nombre del criterio: Limites de uso de CPU.*

*Descripción: Opción que permite especificar el límite de núcleos de la CPU y el porcentaje que serán utilizadas por los paquetes o máquinas virtuales.*

*Tipo de valor: Booleano (Si/No).*

### **3.4.6 Criterio D.6: Puntos de control de particiones**

*Nombre del criterio: Puntos de control.*

*Descripción: Opción que permite respaldar o realizar backups de los paquetes o máquinas virtuales.*

*Tipo de valor: Booleano (Si/No).*

### **3.4.7 Criterio D.7: Privilegios de administración**

*Nombre del criterio: Privilegios de administración.*

*Descripción: Opción de permite a los administradores poder establecer permisos para la instalación, uso y demás opciones de los recursos empleados por los paquetes o máquinas virtuales.*

*Tipo de valor: Booleano (Si/No).*

## 3.5. Categoría E: Generales

### **3.5.1. Criterio E.1: Autor de la herramienta.**

*Nombre del criterio:* Autor.

*Descripción:* Nombre del autor que ha creado la herramienta.

*Tipo de valor:* Texto libre.

### **3.5.2. Criterio E.2: Fecha de creación**.

*Nombre del criterio:* Fecha.

*Descripción:* Fecha en la que se ha creado la herramienta.

*Tipo de valor:* Texto libre.

### **3.5.3. Criterio E.3: Redes Sociales**

*Nombre del criterio:* Redes Sociales.

*Descripción:* Redes sociales que tienen ambas empresas.

*Tipo de valor:* Boolean (Sí/No) y texto libre.

## 3.6. Categoría F: Rendimiento

### **3.6.1. Criterio F.1: Número de kernels en los que funciona.**

*Nombre del criterio:* Número de kernels.

*Descripción:* Cantidad de kernels de Linux en los que funciona cada herramienta.

*Tipo de valor:* Numérico.

### **3.6.2. Criterio F.2: Capacidad de contención.**

*Nombre del criterio:* Capacidad.

*Descripción:* La capacidad de la que disponen ambas herramientas.

*Tipo de valor:* Texto libre.

### **3.6.3. Criterio F.3: Necesidad de extensión IP.**

*Nombre del criterio:* Extensión IP.

*Descripción:* Si es necesario hacer una extensión IP.

*Tipo de valor:* Booleano (Sí/No).

### **3.6.4. Criterio F.4: Funcional dentro del otro.**

*Nombre del criterio:* Funcional.

*Descripción:* Saber si una herramienta puede funcionar dentro de la otra.

*Tipo de valor:* Booleano (Sí/No).

# 4. Evaluación de los criterios por tecnología

## 4.1 Evaluación de los criterios para la tecnología Docker

|  |  |
| --- | --- |
| Criterios | Evaluación |
| Criterio A.1: Soporte y documentación | Encontramos varios cursos de pago y tutoriales gratuitos |
|  |  |
| Criterio A.2: Usabilidad | Sí |
| Criterio A.3: Rendimiento | Docker tiene un gran rendimiento, por lo que puede manejar gran cantidad de daos y a su vez ser bastante rápida su utilización. |
| Criterio A.4: Resultados | El resultado de Docker es muy positivo en los últimos años ya que ha proporcionado usabilidad y anteriormente en el resto de tecnologías era muy complejo. |
| Criterio A.5: Última actualización | 18 de enero de 2017 |
|  |  |
| Criterio B.1: Usuario gratuito | Disponible la edición comunidad con funciones muy limitadas. |
|  |  |
| *Criterio C.1: Costes* | *$750-2000$ (por año)* |
|  |  |
| Criterio D.1: Sistema operativo | Linux |
| Criterio D.2: Aislamiento de sistema de ficheros | Si |
| Criterio D.3: Cuotas de disco | No |
| Criterio D.4: Limite de uso de memoria | Si |
| Criterio D.5: Limite de uso de CPU | Si |
| Criterio D.6: Puntos de control de particiones | No |
| Criterio D.7: Privilegios de administración | Si (desde la versión 1.10) |
|  |  |
| Criterio E.2: Fecha | Marzo de 2013. |
| Criterio E.3: Redes Sociales | Sí (Twitter, Facebook, Youtube, Google+, Github, Linkedin, Reddit, Slideshare). |
|  |  |
| Criterio F.1: Número de kernels | Todos. |
| Criterio F.2: Capacidad. | Una aplicación. |
| Criterio F.3: Extensión IP | No. |
| Criterio F.4: Funcional. | Sí. |

## 4.2 Evaluación de los criterios para la tecnología OpenVZ

|  |  |
| --- | --- |
| Criterios | Evaluación |
| Criterio A.1: Soporte y documentación | Encontramos varios cursos de pago y tutoriales gratuitos |
| Criterio A.2: Usabilidad | Sí |
| Criterio A.3: Rendimiento | OpenVZ ofrece un mejor rendimiento en comparación con soluciones similares. |
| Criterio A.4: Resultados | Dispone de aspectos negativos como que es mucho menos flexible que otros sistemas y altamente dependiente del sistema anfitrión. |
| Criterio A.5: Última actualización | 27 Mar 2017 |
|  |  |

|  |  |
| --- | --- |
| **Criterio B.1: Usuario gratuito** | Dispone de todas las herramientas con capacidad limitada. |

|  |  |
| --- | --- |
|  |  |
| Criterio C.1: Costes | Proporcionan aumento de RAM y de almacenamiento. |
|  |  |
| Criterio D.1: Sistema operativo | Linux |
| Criterio D.2: Aislamiento de sistema de ficheros | Si |
| Criterio D.3: Cuotas de disco | Si |
| Criterio D.4: Limite de uso de memoria | Si |
| Criterio D.5: Limite de uso de CPU | Si |
| Criterio D.6: Puntos de control de particiones | Sí. |
| Criterio D.7: Privilegios de administración | Si |
|  |  |
| Criterio E.2: Fecha | 2005. |
| Criterio E.3: Redes Sociales | Sí (Twitter, Google+, Facebook, Youtube, Slideshare, Linkedin, Github, OpenHUB, Garmin Connect, Reddit, Instagram). |
|  |  |
| Criterio F.1: Número de kernels | 1. |
| Criterio F.2: Capacidad. | Sistema Operativo entero. |
| Criterio F.3: Extensión IP | Sí. |

# 5. Comparación de las tecnologías

|  |  |  |  |
| --- | --- | --- | --- |
| Criterios | Docker | OpenVZ | Comentarios |
| Características | | | |
| A.1: Soporte y documentación | Encontramos varios cursos de pago y tutoriales gratuitos | Encontramos varios cursos de pago y tutoriales gratuitos | Por lo general encontramos más documentación de Docker. |
| A.2: Usabilidad | Sí | Sí | El éxito de Docker es proporcionado por la usabilidad, aunque OpenVZ también cuenta con esta característica. |
| A.3: Rendimiento | Docker tiene un gran rendimiento, por lo que puede manejar gran cantidad de datos y a su vez ser bastante rápida su utilización. | OpenVZ ofrece un mejor rendimiento en comparación con soluciones similares. | Ambos disponen de buen rendimiento. |
| A.4: Resultados | El resultado de Docker es muy positivo en los últimos años ya que ha proporcionado usabilidad y anteriormente en el resto de tecnologías era muy complejo. | Dispone de aspectos negativos como que es mucho menos flexible que otros sistemas y altamente dependiente del sistema anfitrión. | Resultado positivo en los últimos años. |
| A.5: Última actualización | 18 de enero de 2017 | 27 Mar 2017 | Actualizados con relativa frecuencia. |
| A.6: Puntos de control | No | Si | Docker no permite hacer backups de los paquetes |
| A.7: Privilegios de administración | Si | Si | Las dos aplicaciones permiten establecer permisos de los paquetes, pero la mayoría de acciones requieren tener permisos de administrador |
| Generales | | | |
| B.1: Usuario gratuito | Disponible la edición comunidad con funciones muy limitadas. | Dispone de todas las herramientas con capacidad limitada. | OpenVZ nos permite más capacidades gratuitamente. |
| C.1: Costes | $750-2000$ (por año) | Proporcionan aumento de RAM y de almacenamiento. | Para un uso avanzado de estas tecnologías tendremos que aspirar a ediciones de pago. |
| D.1: Sistema operativo | Linux | Linux | Las dos aplicaciones permiten trabajar únicamente en Linux |
| D.2: Aislamiento de sistema de ficheros | Si | Si | Las dos aplicaciones permiten tener un aislamiento de sistema |
| D.3: Cuotas de disco | No | Si | Docker no permite establecer una cuota de disco |
| D.4: Limite de uso de memoria | Si | Si | Las dos aplicaciones permiten establecer el uso de memoria que hacen los paquetes |
| D.5: Limite de uso de CPU | Si | Si | Las dos aplicaciones permiten establecer el uso de la CPU que hacen los paquetes |
| D.6: Puntos de control de particiones | No | Sí. | Docker no permite hacer backups de los paquetes |
| D.7: Privilegios de administración | Si (desde la versión 1.10) | Si | Las dos aplicaciones permiten establecer permisos de los paquetes, pero la mayoría de acciones requieren tener permisos de administrador. En Docker solo desde la versión 1.10 |
| E.2: Fecha | Marzo de 2013 | 2005 | OpenVZ es más antiguo y lleva más tiempo en desarrollo, pero Docker es más popular. |
| E.3: Redes Sociales | Sí  Twitter, Facebook, Youtube, Google+, Github, Linkedin, Reddit, Slideshare | Sí  Twitter, Google+, Facebook, Youtube, Slideshare, Linkedin, Github, OpenHUB, Garmin Connect, Reddit, Instagram | Ambas herramientas tienen las Redes Sociales básicas e importantes. |
| Rendimiento | | | |
| F.1: Número de kernels | Todos | 1 | Docker puede funcionar en todos los kernels importantes mientras que OpenVZ es un kernel propio. |
| F.2: Capacidad | Una aplicación | Un Sistema Operativo | Docker es un contenedor de una aplicación mientras que OpenVZ puede contener un Sistema Operativo entero. |
| F.3: Extensión IP | No | Sí | Dado que OpenVZ es un contenedor de VPS, necesita una extensión de la IP. |
| F.4: Funcional | Sí | No | A partir del kernel 042stab105.4 de OpenVZ se puede introducir la herramienta Docker. |

# 6. Recomendaciones

## 6.1 Situación 1

### **6.1.1 Descripción de la situación**

*Estamos ante el caso de una empresa dedicada al desarrollo de aplicaciones software, principalmente basadas en IOS, aunque también para Android. La empresa en cuestión se trata de una empresa pequeña de no más de 5 empleados que está empezando a inicializarse en los entornos virtuales para la realización de test y pruebas. Las máquinas utilizadas por los programadores corren bajo Windows 8, aunque cabe la posibilidad de migrar hacia Linux, ya que la situación económica de la empresa no es mu holgada, por lo tanto, la entidad tiene que decidir sin incorporar con un bajo coste y un máximo rendimiento la herramienta Docker o OpenVZ.*

### **6.1.2 Recomendación de tecnología a utilizar**

Basándonos en nuestro trabajo de investigación sobre las herramientas en cuestión, hemos decidido que la implantación de Docker es la que más se ajusta a las necesidades de la entidad cubriendo sus necesidades básicas y no produciendo un alto coste.

|  |  |  |
| --- | --- | --- |
| Criterios relevantes para la decisión | Docker | OpenVZ |
| Multiplataforma | Docker es capaz de correr en Linux, aunque gracias a extensiones y plugin se puede ejecutar en Windows e IOS | Funciona única y exclusivamente en Linux, tanto los clientes como el servidor han de correr bajo este SO |
| Escalabilidad | Al no tener tantas restricciones de SO y de portabilidad de imágenes, hace que sea más sencilla la escalabilidad del software | Como ya hemos comentado en el recuadro anterior, sufre más limitaciones lo que a la hora de trabaja con ello produce más inconvenientes, pero no impide su escalabilidad |
| Instalación | La instalación no es demasiado complicada, dependiendo del SO | La instalación tampoco es excesivamente complicada, el único punto débil podría ser la necesidad de conocimientos en Linux a la hora de la configuración |
| Carga en el sistema | Tiene un bajo nivel de incidencia, ya que, al no virtualizar, y funcionar a través de contendores aislados no consume casi recursos del host | No tiene tampoco una gran incidencia a nivel de rendimiento en el host |
| Repositorio | Incluye un repositorio para compartición de información | Repositorios no oficiales y de dudosa fiabilidad |
| Entornos | Los entornos pueden ser configurados para usuarios en concreto, para simular el entorno del cliente, y la facilidad de comunicación entre estas | No hay una gran interoperabilidad entre contendores |

## 6.2 Situación 2

### **6.2.1 Descripción de la situación**

En este caso hablaremos también de una empresa que desarrolla aplicaciones, en este caso única y exclusivamente en lenguaje de programación orientado a objetos (JAVA). Esta empresa de tamaño medio, utiliza distintas versiones de JAVA para realizar diferentes proyectos de desarrollo con los que cuentan actualmente, habiendo funcionalidades en las que utilizarán desarrollo basado en la última versión de JAVA, y en torso casos versiones anteriores. La empresa utiliza como SO principal Linux, ya que son expertos en él. Lo que necesita la empresa es poder crear entornos en los que poder desarrollar los distintos proyectos utilizado en cada uno distintas versiones de JAVA para el desarrollo, utilizando siempre Linux.

### **6.2.2 Recomendación de tecnología a utilizar**

|  |  |  |
| --- | --- | --- |
| Criterios relevantes para la decisión | Docker | OpenVZ |
| Sistema Operativo | Podemos considerarla como una aplicación multiplataforma, en este caso el rendimiento en Linux es bastante alto | Corre únicamente en Linux, y al igual que Docker su rendimiento es bastante alto, consumiendo escasos recursos del servidor |
| Adaptabilidad | Es una herramienta con buena adaptabilidad ha hardware con buen procesamiento que corran en Linux, a la hora de utilizar equipos más limitados puede producir limitaciones | OpenVZ además de tener una buena adaptabilidad en hardware limitados también ofrece un gran rendimiento en máquinas de mayor procesamiento |
| Facilidad | La complejidad de Docker en usabilidad no es complicada, con conocimientos previos, | La usabilidad basada en opiniones de usuarios encontradas en la red, es que la usabilidad del entorno de OpenVZ es más intuitivo |
| Carga en el sistema | Ofrece un alto rendimiento en los hosts | Ofrece un mayor rendimiento en host Linux |
| Administración | Administración de usuarios flexible | Ofrece también una alta flexibilidad |