Técnico superior en Desarrollo de Aplicaciones Multiplataforma

Sistemas de Gestión Empresarial

Rubén Gómez Olivencia

2023-2024

- Github: https://github.com/yuki

Puedes encontrar la última versión de este libro en formato **HTML** en el siguiente link, así como otros libros que he creado. Para descargar el código fuente en formato **Markdown** visita el repositorio en GitHub.

Información

Por favor, ponte en contacto conmigo si encuentras algún fallo, falta de ortografía o quieres mejorar de alguna manera este libro. Gracias.

I Sistemas de Gestión Empresarial

II ERP

III CRM

IV Odoo

V Uso de datos en informes

VIII Introducción a Docker

Sistemas de Gestión Empresarial

1. Introducción

La **informática** es un área de la ciencia que abarca distintas disciplinas teóricas (como la creación de algoritmos, teoría de computación, teoría de la información, ...) y disciplinas prácticas (diseño de hardware, implementación de software).

A la hora de crear programas (o *software*), podemos identificarlos de distintos tipos:

- **Software de sistema**: Programas o aplicaciones que pertenecen al sistema y nos ayudan a mejorarlo, administrarlo ... Pueden ser aplicaciones de monitorización, de auditoría de logs, *drivers*, ...
- **Software de desarrollo**: En este caso serán aplicaciones que nos ayudarán a crear otras aplicaciones. Por ejemplo: librerías de funciones, compiladores, *debuggers*, IDEs...
- **Aplicaciones de usuario**: Son aplicaciones que los usuarios finales utilizarán en su día a día. Podríamos diferenciarlas como:
  - **Aplicaciones generalistas**: Son aquellas que cualquier tipo de usuario utilizará en cualquier momento. Son creadas con un propósito específico, pero que no hay que tener grandes conocimientos para usarlas. Por ejemplo: navegadores web, clientes de correo, aplicaciones de ofimática simple, calculadora, calendario, ...
  - **Aplicaciones de uso específico**: En este caso son aplicaciones creadas para un usuario específico, con una utilidad muy concreta y que normalmente deben existir conocimientos para utilizarla.

    Pueden ser aplicaciones no muy complejas, pero cuya utilidad, o lo que hagan, tenga importancia y conste de procesos complejos. Por ejemplo: aplicaciones CAD, sistemas de virtualización, aplicaciones científicas (R, JupyterLab), aplicaciones empresariales, ...

En esta asignatura veremos distintos tipos de software especializado dentro de la gestión empresarial como son los ERP y los CRM, que podríamos englobar como **sistemas de información**.

2. Sistemas de información

Un sistema de información, de manera generalizada, es aquel que ayuda a administrar, recolectar, recuperar, procesar, almacenar y distribuir información relevante para ser usados dentro de los procesos fundamentales de una organización.

Normalmente estos sistemas de información son fáciles de usar, tienen cierto grado de flexibilidad (se pueden adaptar a las empresas), permiten guardar y recuperar información de manera rápida y sencilla.

De esta manera, la información resultante será más valiosa para la propia organización, ya que tendrá una “imagen” más amplia y habiendo podido relacionar más información que de no haber utilizado este tipo de software.

2.1. Componentes

Un sistema de información debe contar con los siguientes componentes básicos, que deben interactuar entre sí de manera adecuada para un buen funcionamiento global:

- El **hardware**, equipo físico utilizado para procesar y almacenar datos.
- El **software** y los procedimientos utilizados para transformar y extraer información.
- Los **datos** que representan las actividades de la empresa.
- La **red** que permite compartir recursos entre computadoras y dispositivos.
- Las **personas** que desarrollan, mantienen y utilizan el sistema.

El último punto es muy importante, ya que de nada sirve tener la mejor herramienta, en el mejor hardware, si luego las personas que van a hacer uso de ella no tienen los conocimientos suficientes.

¡Cuidado!

Las personas que utilizan los sistemas de información deben tener los conocimientos adecuados para su correcta utilización.

Es por eso que las personas que hagan uso del sistema de información deberán ser entrenadas y/o tener manuales para su correcto uso, así como **también tener en cuenta sus opiniones para mejorar los procesos internos de la empresa**.

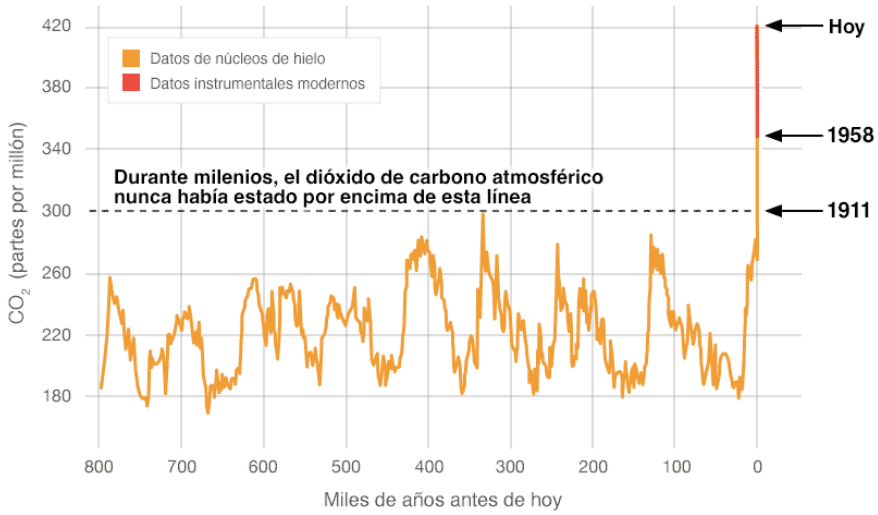
2.2. Datos vs información

Los datos reflejan hechos recogidos en la organización y que están todavía sin procesar (reflejan valores o resultados de mediciones). Estos datos serán hechos o cifras sobre algún tema específico concreto, que a simple vista no tienen por qué decir nada.

Por otro lado, la información se obtiene una vez se han procesado, agregado y/o presentado de manera adecuada esos datos para que puedan ser útiles y de esta manera obtener un valor que de otra manera no se podría obtener.

El ejemplo más claro entre datos e información se puede obtener en cualquier **estudio científico**, en el que a tras la obtención de unos datos, a través del método científico se llega una conclusión y con ello información.

Por ejemplo, mediciones del dióxido de carbono (CO_2) en la atmósfera, se obtienen datos y se llega a la siguiente imagen que es la información:



Mediciones de CO2 en los últimos miles de años. Fuente: NASA

2.3. Objetivo

El objetivo de los sistemas de información, y en este caso, los utilizados para la gestión empresarial, es el de realizar acciones de manera más rápida y eficiente, por lo que también debería ser más económico para la empresa.

El uso de las tecnologías de la información y la comunicación en las empresas se ha convertido en un **elemento esencial como motor vertebrador y fuente de ventajas competitivas**.

Hoy en día una empresa que no haga uso de la informática está condicionando su estrategia empresarial, y es bastante probable que esté perdiendo oportunidades de negocio, así como la posibilidad de desarrollar sus productos y servicios.

Es por eso que el uso de la informática y de *software* especializado de gestión empresarial puede ayudar a las empresas en:

- Obtener ventajas competitivas.
- Mejorar la eficiencia interna de la empresa: reducir costes, mejorar la productividad, mejorar la organización de la información, ...
- Mejorar y facilitar la toma de decisiones a través de la recopilación de la información.
- Para desarrollar nuevas estrategias de negocios.

2.4. Requisitos

Para que la información sea útil en la toma de decisiones dentro de una organización, debe cumplir una serie de requisitos:

- **Exactitud**: debe ser precisa y libre de errores.
- **Comprensión**: inteligible por el usuario.
- **Completitud**: debe contener todos aquellos hechos que pudieran ser importantes.

- **Economicidad**: el coste para obtener la información debe ser menor que el beneficio.
- **Confianza**: garantizar tanto la calidad de los datos utilizados, como la de las fuentes de información.
- **Relevancia**: ha de ser útil para la toma de decisiones.
- **Nivel de detalle**: se debe proporcionar con la presentación y el formato adecuados, para que resulte sencilla y fácil de manejar.
- **Oportunidad**: se debe entregar la información a la persona que corresponde y en el momento adecuado.
- **Verificabilidad**: la información ha de poder ser contrastada y comprobada en todo momento.

¡Atención!

A tener en cuenta: **el exceso de información también puede ser contraproducente**.

2.5. Actividades

A la hora de hacer uso de un sistema de información, las actividades que se pueden realizar con él se pueden resumir en:

- **Recopilación**: Es la recogida de datos en bruto. Estos datos pueden ser de dentro de la organización, del exterior, recopilados de manera automática o de manera manual.
- **Almacenamiento**: Los datos deben ser guardados de manera estructurada para su posterior uso. Por otro lado, **nos debemos asegurar que su persistencia no corra peligro**, por lo que deberemos contar con un sistema de almacenamiento que sea capaz de asegurar posibles problemas. Para ello deberemos tener un sistema en **alta disponiblidad**, y con un buen sistema de **backups** configurado.

  También hay que asegurar que **el acceso a los datos estará limitado y asegurado mediante sistemas de control de acceso y de autenticación**.
- **Procesamiento**: Es el punto clave en el que los datos se convierten en información, de esta manera cumpliendo la labor de ayudar a la organización en la toma de decisiones.
- **Distribución**: El sistema permitirá distribuir la información entre las personas que la necesiten.

2.6. Tipos de sistemas de información

Aunque existen distintos tipos de sistemas de información, y su clasificación se puede realizar teniendo en cuenta distintas funcionalidades y/o objetivos, nos vamos a centrar en dos tipos:

- **ERP**: *Enterprise Resource Planning* o planificación de recursos en la empresa. Se trata de los sistemas de gestión integrados que permiten dar soporte a la totalidad de los procesos de una empresa: control económico financiero, logística, producción, mantenimiento, recursos humanos, ...

- **CRM**: *Customer Relationship Management*, sistemas para gestionar las relaciones con los clientes y el soporte a todos los contactos comerciales.

ERP

1. Introducción

Los sistemas de planificación de recursos empresariales (**ERP**, por sus siglas en inglés, *enterprise resource planning*) son los sistemas de información gerenciales que integran y manejan muchos de los negocios asociados con las operaciones de producción y de los aspectos de distribución de una compañía en la producción de bienes o servicios.

Los sistemas ERP son llamados ocasionalmente ***back office*** (trastienda) ya que el cliente y el público general no tienen acceso a él. Sólo los usuarios internos de la empresa (y no tienen por qué ser todos) accederán a distintos apartados para realizar modificaciones. Estas modificaciones se visualizarán, o tendrán efecto, sobre lo que el usuario final verá.

2. Objetivos

Los sistemas ERP son sistemas de gestión de información que automatizan muchas de las prácticas de negocio asociadas con los aspectos operativos o productivos de una empresa.

Los sistemas ERP suelen estar compuestos por distintos módulos para realizar diferentes acciones dentro de la empresa. En caso de necesitar cualquiera de ellos, se realizará la instalación o configuración del mismo, ya que lo habitual suele ser que estén desactivados por defecto.

Entre los módulos más habituales que nos podemos encontrar se pueden destacar: producción, ventas, compras, logística, contabilidad (de varios tipos), gestión de proyectos, GIS, inventarios y control de almacenes, pedidos, nóminas, ...

Los objetivos principales de los sistemas ERP son:

- Unificación y trazabilidad de todos los procesos en un mismo sistema.
- Optimización de los procesos empresariales.
- Planificación de los recursos.
- Automatización de los procesos entre las áreas de la empresa.
- Acceso a los datos y creación de información estructurada.
- Posibilidad de compartir información entre todos los componentes de la organización.
- Eliminación de datos y operaciones innecesarias de reingeniería.

3. Características

Tal como nos dice Wikipedia, las características que distinguen a un ERP de cualquier otro software empresarial son que deben ser modulares, configurables y especializados:

- **Modulares**. Los ERP entienden que una empresa es un conjunto de departamentos que se encuentran interrelacionados por la información que comparten y que se genera a partir de sus procesos. Una ventaja de los ERP, tanto económica como técnica, es que la funcionalidad se encuentra dividida en módulos, los cuales pueden instalarse de acuerdo con los requerimientos del cliente. Ejemplo: ventas, materiales, finanzas, control de almacén, recursos humanos, etc.
- **Configurables**. Los ERP pueden ser configurados mediante desarrollos en el código del software. Por ejemplo, para controlar inventarios, es posible que una empresa necesite manejar la partición de lotes, pero otra empresa no. Los ERP más avanzados suelen incorporar herramientas de programación de cuarta generación para el desarrollo rápido de nuevos procesos.
- **Especializados**. Un ERP especializado, brinda soluciones existentes en áreas de gran complejidad y bajo una estructura de constante evolución. Estas áreas suelen ser, el verdadero problema de las empresas, además de contener todas las áreas transversales. Trabajar bajo ERP especializados es el paso lógico de las empresas que requieren soluciones reales a sus verdaderas necesidades.Un ERP genérico solo ofrece un bajo porcentaje de efectividad basado en respuestas generalistas, que requieren ampliaciones funcionales.

CRM

1. Introducción

Tal como nos dice Wikipedia, la gestión o administración de relaciones con el cliente (*customer relationship management*), más conocida por sus siglas en inglés **CRM**, puede tener varios significados:

- **Administración o gestión basada en la relación con los clientes**: un modelo de gestión de toda la organización, basada en la satisfacción del cliente (u orientación al mercado según otros autores). El concepto más cercano es marketing tradicional.

- **Software para la administración o gestión de la relación con los clientes**: Sistemas informáticos de apoyo a la gestión de las relaciones con los clientes, a la venta y al marketing, y que se integran en los llamados Sistemas de Gestión Empresarial (SGE), y que incluyen CRM, ERP, PLM, SCM y SRM.

  El software de CRM puede comprender varias funcionalidades para gestionar las ventas y los clientes de la empresa: automatización y promoción de ventas, tecnologías data warehouse («almacén de datos») para agregar la información transaccional y proporcionar capa de reporting, dashboards e indicadores claves de negocio, funcionalidades para seguimiento de campañas de marketing y gestión de oportunidades de negocio, capacidades predictivas y de proyección de ventas.

El CRM es un enfoque para gestionar la interacción de una empresa con sus clientes actuales y potenciales, es una forma de pensar y de actuar de una empresa hacia los clientes/consumidores. **Utiliza el análisis de datos de la historia de los clientes con la empresa y para mejorar las relaciones comerciales** con dichos clientes, centrándose específicamente en la retención de los mismos y, en última instancia, impulsando el crecimiento de las ventas.

2. Objetivos

Teniendo en cuenta lo dicho anteriormente, si el CRM está separado del programa que gestiona las ventas y recursos, puede resultar complejo el realizar una buena gestión de todo ello. Al final, la relación con el cliente es debido a las ventas o proyectos que se le realizan.

Los sistemas de software CRM aportan distintas funciones para a la gestión de las relaciones con el cliente:

- Almacenamiento de los datos de los clientes a nivel comercial.
- Creación de segmentos personalizados para distintos objetivos.
- Seguimiento tanto de clientes como de ventas.
- Automatización de los procesos de venta (leads, alertas, tareas…).
- Generación de promociones específicas.
- Gestión del servicio postventa de los productos.

3. Componentes

Los componentes principales de un CRM están construidos y manejan la relación con el cliente en base al marketing, observando la relación que existe a lo largo del tiempo a través de las distintas etapas de proyectos, ya que estas no son homogéneas.

¡Atención!

No es lo mismo la relación con un cliente al inicio del primer proyecto que durante la ejecución de un tercer proyecto.

3.1. Tipos de comunicación

Podríamos identificar como buenos componentes de relación con el cliente los siguientes:

- **Comunicación verbal**: Parte de la comunicación que realizaremos con el cliente será de forma verbal. De esta manera obtendremos una cercanía y un inicio de relación donde asentar las bases de la relación comercial.

  No tiene por qué ser el primer acercamiento, pero será, normalmente, el que más estreche la relación. También suele ser la manera más rápida para compartir cierta comunicación y para comenzar las bases de inicio de proyectos.

- **Internet**: Hoy en día el marketing a través de Internet es el más habitual. Ya sea a través de la página web corporativa, redes sociales, o en algunos casos incluso a través de campañas de publicidad incrustadas en otras webs.

  Hoy en día para la captación de nuevos clientes, para dar a conocer las novedades y los avances en productos/proyectos, es una manera clave de mantener o iniciar la relación con clientes.

- **Campañas por e-mail**: Aunque un poco relegadas por las redes sociales, las campañas por e-mail siguen siendo una manera en la que una empresa se debe relacionar con los clientes. Pueden ser de dos tipos:

  - **Dirigidas a clientes concretos**: Dependiendo de lo que queramos ofrecer, y teniendo en cuenta lo que los clientes han contratado previamente, se podrá dirigir las campañas a clientes concretos, tratando de buscar el mayor impacto posible.

  - **Generalizadas**: Por ser campañas más generalistas, o simplemente para ofrecer información general (posibles cambios dentro de la empresa, o mejoras en los servicios), estas campañas pueden ser dirigidas a todos los clientes en general.

- **Marketing telefónico**: Algunas empresas, dirigidas a servicios generalistas, pueden hacer uso de campañas de marketing a través del teléfono en busca de nuevos clientes.

- **Soporte**: Una vez terminado un proyecto, o con la contratación de un servicio, es habitual tener un

sistema de soporte al cliente.

A través de este sistema, el cliente podrá pedir “ayuda” y dependiendo del tipo de empresa un técnico especializado le resolverá las dudas o le ayudará.

IV

Odoo

1. Introducción

Odoo, antes conocido como *OpenERP* es un software de planificación de recursos empresariales con licencia dual. Existe una versión de código abierto y una versión con licencia comercial con características y servicios exclusivos.

Odoo también cuenta con un apartado de CRM (en inglés *customer relationship management*, o gestión de relaciones con el cliente), pudiendo también crear una web de comercio electrónico, facturación, ...

2. Instalación

Tal como hemos visto en el tema anterior, la instalación de un sistema se puede realizar de distintas maneras, por lo que deberemos atender a las necesidades del proyecto para realizar una instalación adecuada.

En nuestro caso, se va a optar por realizar una instalación a través de servicios Docker, de esta manera podemos realizar pruebas con distintas versiones (tanto de Odoo como de la base de datos).

Información

La alternativa sería realizar la instalación en una máquina virtual o haciendo uso de los distintos instaladores que existen en la web oficial

Para realizar la instalación seguiremos las indicaciones que aparecen en la web de Docker Hub haciendo pequeñas modificaciones.

2.1. Servicios independientes

Es el sistema más básico, que requiere de levantar dos contenedores Docker:

- Contenedor de base de datos **PostgreSQL**. Podremos elegir entre las distintas versiones del gestor de base de datos, pero haremos caso a las recomendaciones de la web. Habría que tener especial cuidado con el usuario y la contraseña que utilizamos.

  En este caso también se ha expuesto el puerto 5432 que es el puerto por defecto de PostgreSQL:

Crear y arrancar el contenedor de la base de datos

```
ruben@vega:~$ docker run -d -e POSTGRES_USER=odoo \
-e POSTGRES_PASSWORD=odoo -e POSTGRES_DB=postgres \
-p 5432:5432 --name odoo_db postgres:15
```

- Contenedor con el propio **Odoo**. Este contenedor tendrá los servicios y librerías necesarias para poder hacer funcionar la aplicación web.

Crear y arrancar el contenedor de la base de datos

```
ruben@vega:~$ docker run -d -p 8069:8069 --name odoo \
--link odoo_db:db -t odoo
```

2.2. Docker Compose

Docker Compose es una herramienta para correr servicios multi-contenedor y se crea a través de un fichero en formato YAML. Es una manera de crear,parar,reconstruir una arquitectura de servicios de manera rápida y sencilla.

Se debe crear un fichero `compose.yaml` y lo ideal es que esté dentro de un directorio con el nombre del "stack de servicios", ya que coge el directorio como parte del nombre a la hora de crear los contenedores.

Contenido de fichero compose.yaml

```yaml
version: '3.1'
services:
  web:
    image: odoo:16.0
    depends_on:
      - db
    ports:
      - "8070:8069"
  db:
    image: postgres:15
    environment:
      - POSTGRES_DB=postgres
      - POSTGRES_PASSWORD=odoo
      - POSTGRES_USER=odoo
    ports:
      - "5433:5432"
```

Para arrancar los servicios se realiza, desde el mismo directorio donde se encuentra el fichero, con el siguiente comando

Levantar docker compose

```
ruben@vega:~$ docker compose up
```

¡Cuidado!

En Julio del 2023 se migró a Compose v2, tal como se indica en la web oficial. Dependiendo de la versión que tengamos instalada será `docker compose up` o `docker-compose up`

2.3. Herramientas extra

Para poder acceder a la base de datos podemos hacer uso de un cliente externo. De esta manera no tendremos que entrar al contenedor y tendremos un interfaz gráfico con el que poder administrarla.

Podemos utilizar:

- **DBeaver**: Es una aplicación de escritorio que permite conectarnos a distintos SGBDs. Existe versión community y otra con licencia que permite también conectarse a bases de datos NO-SQL.
- **pgAdmin**: Es una aplicación que permite administrar PostgreSQL a través del servidor web.

Crear y arrancar el contenedor pgAdmin

```
ruben@vega:~$ docker run -p 8090:80 \
-e 'PGADMIN_DEFAULT_EMAIL=user@domain.com' \
-e 'PGADMIN_DEFAULT_PASSWORD=SuperSecret' \
--name pgadmin4 -d dpage/pgadmin4
```

V

Uso de datos en informes

1. Dashboards

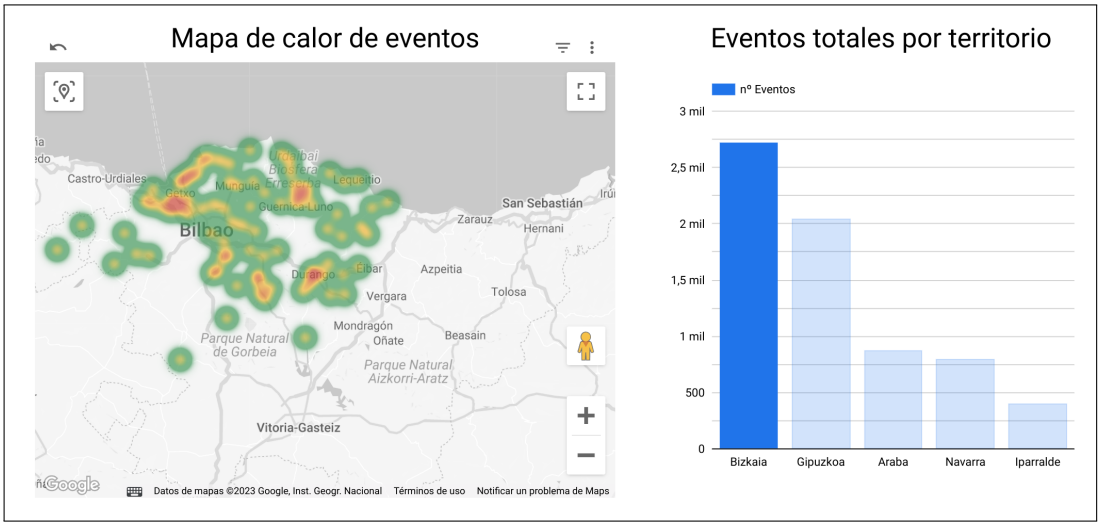
Los paneles de información (en inglés *dashboard*) son herramientas gráficas que nos permiten visualizar datos para convertirlos en información sin tener que realizar ningún tipo de programación.

Podríamos decir que los paneles son un interfaz gráfico previamente programado que obtiene datos de una fuente de datos (una hoja de cálculo, un fichero CSV, una base de datos,...) para posteriormente visualizarlo de una forma gráfica.

2. Características

Los paneles de información suelen contar con las siguientes características:

- Son **fáciles de usar**: no es necesario ser programador para hacer uso de ellos, **siempre y cuando los datos los tengamos en un formato correcto**.
- Permiten hacer uso de **diferentes tipos de gráficos**: tablas, mapas para datos geolocalizados, mapas de calor, gráficos tipo tarta, mapas de barras (horizontales y verticales), ...
- Son **interactivos**: podemos hacer que los gráficos estén enlazados entre sí, y de esta manera al seleccionar un apartado de uno, el resto se actualizan.



Gráficos enlazados

- Se pueden generar **filtros** para realizar una búsqueda en los datos.
- Podemos **compartir** el dashboard creado o generar una presentación.

Como contrapartida, también pueden existir algunos inconvenientes a la hora de utilizarlos, que debemos de tener claros antes de utilizarlos:

- Estamos **limitados a las características que ofrecen**. Por lo tanto, si nos interesa crear un gráfico distinto a los que ofrece, no vamos a poder hacerlo.
- Los datos deben de estar en el **formato que acepte la herramienta**. Si programamos nosotros, el

formato puede ser propio, o el diccionario de datos puede estar a nuestro gusto.

- **Dependencia** de la herramienta. Si deciden eliminar funcionalidades, hacerla de pago, ... deberemos migrar los datos y crear de nuevo el panel.

3. Entendiendo los datos

Antes de crear un panel de información debemos tener claro los tipos de datos que tenemos, cómo están organizados (o si debemos organizarlos), en qué formato están, ... es por eso que debemos realizar un análisis previo de los mismos.

3.1. Análisis de datos

A la hora de realizar el análisis de datos, debemos tener en cuenta, al menos, las siguientes características:

- **Fuente de los datos**: ¿son datos propios? ¿podemos recuperarlos en cualquier momento? ¿son fiables?
- **Formato de los datos**: Debemos tener en cuenta no sólo el formato del fichero en el que obtenemos los datos, si no si estos están normalizados. Por ejemplo, si los números están guardados en formato entero/real o en formato texto; si existen datos de geolocalización, si están separados o agregados; ...
- **Entender los datos**: Aunque pueda parecer obvio, es necesario **entender cada apartado de los datos** para posteriormente relacionarlos entre sí, y para hacer uso de ellos en el panel/gráfico.

3.2. Creación del esquema de datos

Teniendo en cuenta el punto anterior, quizá sea necesario normalizar los datos o crear un formato específico. Por lo tanto, debemos de conocer algún tipo de herramienta o de lenguaje de programación que nos permita hacerlo.

Si los datos en bruto obtenidos no cumplen las expectativas, entre las tareas que debemos realizar estarán:

- **Normalizar los datos**: A cada tipo de datos convertirlo al formato que debe ser. Por ejemplo, en algunos datos de geoposición de OpenData Euskadi, hacen uso de un formato llamado UTM, que es necesario convertir a latitud y longitud.
- **Diccionario de datos**: Crear el esquema/diccionario de datos que mejor se adecúe a los datos y al sistema de panel que vayamos a utilizar.

3.3. Crear el tipo de fichero

Por útlimo, será necesario generar el tipo fichero con el formato que la herramienta necesita: json, CSV, formato excel, ...

Para ello, dependiendo del esquema de datos, podremos generar el fichero con herramientas simples (por ejemplo excel, para generar un fichero en formato CSV) o deberemos hacer uso quizá de lenguajes de programación con librerías específicas que nos faciliten la labor (por ejemplo python).

3.4. Documentación del análisis

Siempre es recomendable documentar el proceso del análisis de datos para entender en etapas sucesivas, o tiempo después, el por qué se han modificado los datos tal como se ha hecho, o la razón de hacer uso de un tipo de fichero y no otro.

La documentación también facilitará el realizar modificaciones más adelante o para poder realizar adaptaciones a nuevas herramientas. De esta manera, en principio, tendremos parte del trabajo hecho y no habrá que volver a realizarlo.

4. Looker Studio como generador de paneles de información

Looker Studio, anteriormente DataStudio, es una herramienta de Google para visualizar y crear paneles informativos con datos que proporcionamos desde distintas fuentes.

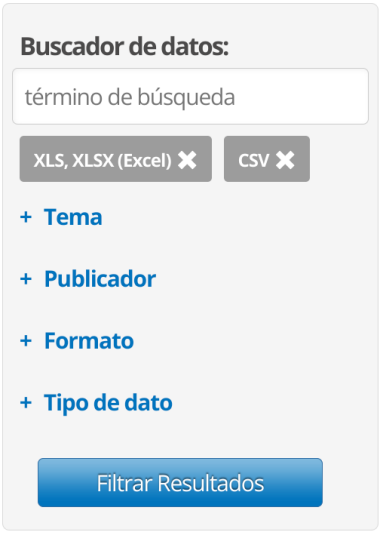
Es una herramienta gratuita, fácil de usar y que permite generar distintos tipos de paneles con los datos proporcionados. Aparte, también permite crear otros tipos de visualizaciones, por lo que la comunidad aporta nuevos paneles.

Para poder crear los paneles, necesitaremos hacer uso de datos, que podremos importar o usar desde distintas fuentes, como por ejemplo:

- Ficheros CSV (comma separated value)
- Hojas de cálculo de Google
- Bases de datos MySQL, PostgreSQL, SQL Server, ...
- Otros conectores creados por la comunidad

4.1. Obtener datos

En primer lugar vamos a necesitar crear una fuente de datos, donde deberemos subir los datos obtenidos. Tal como se ha dicho previamente, es necesario realizar un análisis de los mismos para posteriormente normalizarlos en caso necesario.



Realizar búsqueda enOpenData Euskadi

Para este ejemplo se va a hacer uso de una hoja de cálculo de Google con datos obtenidos de OpenData Euskadi. Buscaremos para que los datos estén en formato CSV y/o XLS de Excel, y de esta manera la conversión de los datos no resulte tan compleja.

En caso de estar en formato JSON, u otro, quizá deberíamos realizar una conversión previa. Aparte, de que en caso de querer realizar la manipulación de los datos, no sería tan directa.

Tras realizar la búsqueda obtendremos un listado de resultados, en el que podremos analizar si son de nuestro interés. En algunos casos los nombres son bastante representativos. Buscaremos datos que sean de nuestro interés y que tengan datos de geolocalización.

Una vez descargado los datos, y generado una hoja de cálculo en Google con ellos, deberemos realizar el análisis de los datos. Entre otras cosas, deberíamos:

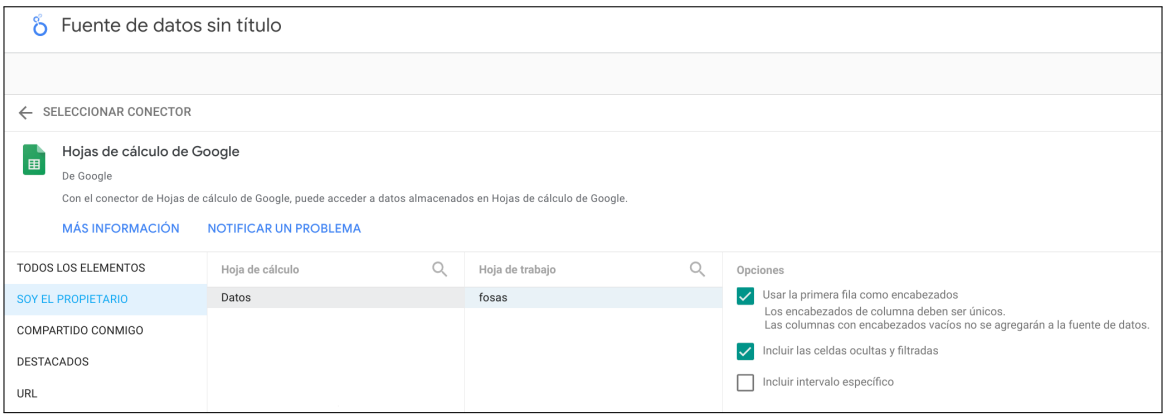
- Renombrar la primera fila, para que cada columna esté identificada de manera correcta.
- Comprobar si hay datos duplicados que se puedan borrar.
- Eliminar columnas que no nos interesen y/o que estén vacías
- Comprobar si los datos geográficos están en el formato adecuado. Para ello, podremos ver si al usarlos en Google Maps hacen referencia a la localidad correspondiente.
    - Los datos latitud y longitud deben estar en la misma celda y separado por una coma.
    - En caso de estar en formato UTM, habría que convertirlos (se puede usar este conversor).

Una vez realizado, deberíamos tener los datos listos para poder ser usados.

4.2. Crear fuente de datos en Looker Studio

Para poder hacer uso de los datos en los paneles de información, primero debemos agregar una “Fuente de datos”, por lo que para ello vamos a “Crear → Fuente de Datos”, y elegimos el conector específico que nos interese.

En el caso de utilizar una hoja de cálculo de Google, deberemos darle permisos para poder acceder, y elegiremos la hoja de cálculo que nos interese. También nos va a permitir seleccionar unas opciones durante la importación:



Crear fuente de datos

Tras realizar la importación, dentro de la fuente de datos deberemos identificar el tipo de fuente que es cada dato. En algunos casos Looker Studio identificará el tipo correcto, pero en otros no, como sucede con las posiciones de geolocalización.

Datos normalizados

Una vez realizada la normalización de los datos, podemos comenzar a crear el informe.

4.3. Crear informe

Una vez terminada la normalización de los datos, es momento de comenzar a crear el informe y convertir los datos en información. Para ello, debemos pensar qué tipo de información queremos mostrar y cómo.

Looker Studio nos permite hacer uso de distintos tipos de gráficos, cada uno con sus peculiaridades y su configuración, por lo que es interesante investigar cada uno de ellos y comprobar su funcionamiento.

A continuación se explican algunos de ellos:

- **Tabla**: Es la manera más sencilla de representar los datos, y tal como dice su nombre, en formato tabla, separado por filas y columnas. Podremos colocar las columnas en el orden adecuado, generar sistemas de ordenación, añadir colores a las filas pares o impares, ...
- **Cuadro de resultados**: Normalmente son utilizados para contabilizar el número de datos seleccionados o totales.
- **Gráfico de barras**: Nos muestra los datos agregados en columnas, pudiendo elegir en formato vertical, horizontal, ascendente, descendente...
- **Mapas**: Existen distintos tipos de mapas, que dependiendo de los datos nos permitirá visualizar datos de una manera u otra.
    - **Mapa de calor**: Nos crea un mapa de calor con los datos posicionados.

- **Mapa de burbujas**: Geolocaliza los datos añadiendo pequeños círculos. Estos pueden ser de distintos colores identificando algún aspecto destacable de los datos.

En el siguiente enlace se puede ver los aspectos más básicos de un informe.

VI

Alta Disponibilidad y Arquitectura de sistemas

1. Alta Disponibilidad

La alta disponibilidad en servidores se puede definir como el diseño de infraestructura (y su implantación) que asegura la continuidad del servicio y que no tiene un único punto de fallo.

Es lógico entender que un servicio debe de ser contínuo en el tiempo, ya que debe de dar servicio de manera continuada para que los usuarios puedan acceder a él. Pero para que esta premisa sea efectiva, y para asegurarnos que así sea, **la infraestructura debe de estar redundada y carecer de puntos de fallo únicos en su diseño**.

Esto quiere decir, que de cada servicio y para cada posible punto de fallo deberá haber al menos dos de ellos, para que en caso de que uno deje de funcionar el servicio siga funcionando (dos tomas eléctricas separadas, dos servidores que otorguen el servicio, dos conexiones a internet, ...).

Es habitual que un sistema en Alta Disponibilidad deba de estar pensado desde el diseño. Algunos tipos de servicios pueden empezar como un único servidor y posteriormente realizar un escalado horizontal, formando la alta disponibilidad, mientras que para otros **el diseño en alta disponibilidad debe de estar pensado desde el comienzo** (habitualmente en algunos tipos de clusters).

1.1. Importancia de un sistema en Alta Disponibilidad

Como se ha citado previamente, la alta disponibilidad nos va a asegurar al menos tres grandes ventajas:

- Una continuidad en el servicio
- Un diseño libre de puntos de fallos únicos, gracias a la redundancia.
- Mejorar el rendimiento global.

La redundancia permitirá que en caso de fallo de algún equipamiento/servicio, al estar redundando, no afecte al servicio. Gracias al sistema de monitorización seremos capaces de ver el problema y solventarlo lo antes posible. De estar el diseño correcto, el servicio mantendrá su actividad, mientras que por el contrario, si ha habido algún fallo en el diseño de infraestructura (o el problema es más grave de lo esperado) el servicio se verá afectado.

1.2. Tipos de Alta Disponibilidad

Existen muchos tipos de alta disponibilidad dependiendo de en qué capa de infraestructura estemos hablando. Por poner unos ejemplos:

- **Redundancia eléctrica**: Los servidores normalmente cuentan con doble fuente de alimentación, por lo que cada fuente de alimentación debe de estar conectada a una toma eléctrica completamente separada de la otra.
- **Redundancia de conectividad física**: El acceso a internet debe de ser redundado.
- **Redundancia de conectividad LAN**: El acceso a la LAN/DMZ/red de servicio debe de estar redundado (stacks de switches, LACPs configurados en switches y servidores, …).

- **Redundancia de servidores**: Debe de existir una redundancia de servidores para asegurar que el servicio funcione en más de un servidor físico.
- **Redundancia de servicio**: El servicio que se ofrece debe de estar redundado entre los distintos servidores.

La alta disponibilidad también se puede diferenciar como:

- **Alta disponibilidad real**: En caso de que haya algún problema el servicio continúa como si no hubiese pasado nada, gracias a la redundancia completa de servicios/dispositivos.
- **Alta disponibilidad pasiva**: En caso de error, los servidores activos serían los que reciben el impacto del problema y hay que escalar los servidores pasivos a modo activo para que comiencen a funcionar otorgando el servicio. Como se puede presuponer, esta modificación puede ser realizada de manera automática o de manera manual (lo que llevaría algo de tiempo, y por tanto el servicio se vería afectado).

2. Arquitectura de instalación

A la hora de realizar la instalación de un sistema de información, y teniendo en cuenta que es un pilar fundamental de la empresa, habrá que tomar ciertas decisiones desde el punto de vista de sistemas hardware.

De estas decisiones se encargará el ***sysadmin***, o administrador de sistemas, pero tendrá que tener ayuda de los especialistas de la aplicación de sistemas de información, así como de determinar una decisión desde el punto de vista empresarial.

Entre las tareas que hay que tener en cuenta, se podrían destacar las siguientes:

- **Hardware**: Determinar el hardware en donde se va a realizar la instalación. Hoy en día existen distintas alternativas, como son:
  - **Hardware dedicado**: Un servidor propio para el sistema, donde se realizará la instalación sólo para este servicio.
  - **Máquina Virtual**: El servicio será instalado en una máquina virtual a través de un sistema de virtualización profesional. El servicio es agnóstico al hardware, por lo que no sabrá si está virtualizado o no. Hoy en día suele ser la opción más común dadas las ventajas que ofrecen.
- **Elección del sistema de información**: Esta es una tarea importante y que no se puede dejar de lado, ya que la decisión de optar por una herramienta u otra puede suponer un problema a futuro.

  Es por eso que se debe realizar un estudio de mercado entre las distintas posibilidades y tener en cuenta, al menos, las siguientes situaciones:

  - **Estado actual de la herramienta**: Es importante saber si la herramienta analizada cuenta con un desarrollo continuado, si existe una empresa o grupo de desarrollo por detrás que la apoye; que no esté abandonada; que sea una herramienta con buena aceptación y críticas...

- **Coste de licencia**: Es una herramienta que cuenta con una licencia a perpetuidad bajo un coste determinado**,** tiene licencia por el número de usuarios que acceden a ella**,** es una herramienta de Software Libre ...

- **Seguridad, actualizaciones y parches**: La herramienta cuenta con actualizaciones de seguridad periódicas; no ha habido fallos de seguridad graves en las últimas versiones; cuando se detectan fallos las actualizaciones aparecen de manera rápida y efectiva; las actualizaciones y/o parches son gratuitos o de pago...

- **Coste de mantenimiento**: Existe un coste asociado al mantenimiento de la aplicación, pero este puede ser por parte del sistema (realizar actualizaciones, aumento de recursos...) o por pago de licencias anuales, por versiones...

- **Posibilidades de personalización**: Existe la posibilidad de personalizar la herramienta; parametrizar opciones propias que se ajustan a la empresa; creación de módulos/plugins propios para mejorar/expandir la funcionalidad de la herramienta, ...

- **Conocimientos sobre la herramienta**: Dentro de la organización se cuenta con conocimientos acerca del uso/instalación/administración de la herramienta, debe ser subcontratado o existe la posiblidad de adquirir conocimiento mediante cursos o manuales.

- **Sistema operativo**: Dependiendo del sistema de información elegido, se deberá instalar en un sistema operativo u otro. En este punto se pueden tener en cuenta también los puntos anteriores sobre el conocimiento para la toma de decisiones.

- **Método de instalación**: Hoy en día existen distintas posibilidades a la hora de instalar servicios, por lo que es importante realizar una buena decisión:

  - **Tradicional**: Vamos a llamar sistema tradicional a aquel que se realiza mediante un instalador que realiza la instalación en el sistema operativo, que no suele dar demasiadas opciones de configuración durante el proceso.

  - **Contenedores**: Hoy día existen servicios que podemos instalar a través de sistemas de contenedores (como puede ser Docker), los cuales suelen facilitar la instalación, así como la posibilidad de que también sea un sistema multicapa.

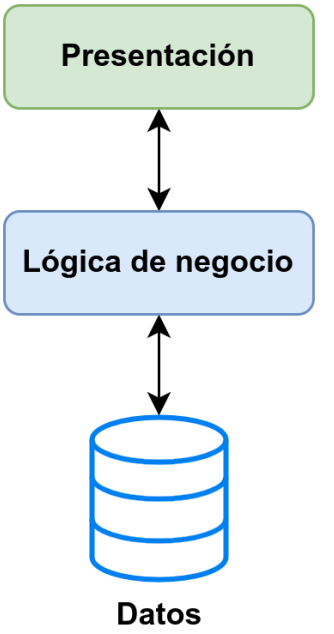
  - **Por capas**: La instalación multicapa puede resultar un poco más compleja y la aplicación/servicio debe poder permitir realizarlo. Aunque inicialmente pueda suponer un poco más de esfuerzo, pero a la larga puede suponer una gran ventaja como es la **alta disponibilidad**.

    Pasar de un sistema “monolítico” a un sistema por capas es posible, pero una vez más dependeremos de la aplicación. Por otro lado, si desde el inicio se ha creado un sistema multicapa, escalarlo será más sencillo que realizar la migración cuando ya esté en uso.

2.1. Arquitectura multicapa

Un sistema informático multicapa es aquel que hace uso de una arquitectura **cliente-servidor** en las que existe una separación lógica y/o física entre las distintas funciones que tiene una aplicación o servicio.

Normalmente se suele representar como una arquitectura en tres niveles, siendo estos:



- **Capa de presentación**: Es la que ve el usuario (también se la denomina «capa de usuario»), presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato).

  También es conocida como interfaz gráfica y debe tener la característica de ser «amigable» (entendible y fácil de usar) para el usuario. Esta capa se comunica únicamente con la capa de negocio.

  Hoy en día lo habitual es que hagamos uso de servicios web, por lo que la capa de presentación es **la web que estamos visualizando**. En el caso de aplicaciones móviles, es **la propia aplicación que tenemos instalada en el dispositivo**.

- **Capa de negocio**: es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse.

  Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él. También se consideran aquí los programas de aplicación.

  En este tipo de arquitecturas, esta capa es la que se denomina ***backend***, y lo habitual es que sea un sistema al que llamamos a través de una API (del inglés, *application programming interface*, o interfaz de programación de aplicaciones).

- **Capa de datos**: es donde residen los datos y es la encargada de acceder a los mismos. Está formada

por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

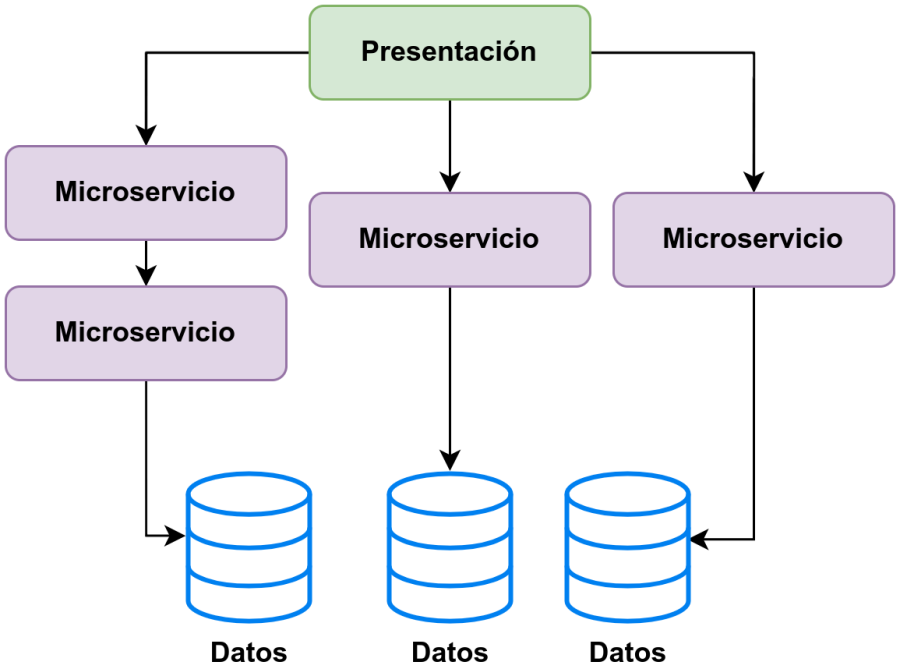
Información

Las aplicaciones web se pueden separar en dos capas: aplicación y base de datos.

En aplicaciones web es posible crear una arquitectura en capas aunque la aplicación no esté 100 % pensado para ello: la propia aplicación y la base de datos.

2.2. Arquitectura de microservicios

Para poder crear una arquitectura de microservicios el enfoque debe darse desde el primer momento del desarrollo de software. Es decir, antes de realizar ningún tipo de programación la aplicación se planteará como pequeños servicios que podrán interactuar entre sí.



Cada servicio se encargará de implementar una única funcionalidad. En caso de necesitar alguna característica que se repita en varios, se debería crear un microservicio que proporcione dicha característica o funcionalidad.

Información

Podríamos comparar una arquitectura de microservicios como una librería de programación, en la que existen funciones que realizan una única función.

Cada microservicio se desplegará de manera independiente, e incluso cada uno podrá estar programado en distintos lenguajes de programación. De esta manera **se puede hacer uso del lenguaje y la tecnología más adecuada para cada funcionalidad**.

3. Escalabilidad

Teniendo en cuenta todo lo dicho hasta ahora, cuando un sistema empieza a tener problemas de rendimiento deberemos abordar el problema y plantearnos cómo solucionarlo. De no hacerlo, se corre el peligro de que el servicio se vea interrumpido y por tanto perder tiempo de trabajo.

Antes de realizar ninguna modificación habría que analizar qué es lo que está sucediendo (para ello es importante tener un buen sistema de monitorización), y de esta manera saber en qué punto existe el problema y así poder solucionarlo.

Dependiendo de las decisiones tomadas durante la instalación, y tras lo visto previamente, podremos abordarlo de dos maneras diferentes.

Información

"Escalabilidad" no existe en el diccionario de la RAE, pero se usa como anglicismo de la palabra *scalability*.

3.1. Escalado vertical

Cuando se escala verticalmente un sistema lo que se va a realizar es **añadir más recursos al nodo que está teniendo problemas**. Tras el análisis previo realizado se añadirán los recursos necesarios (más RAM, discos duros más rápidos, aumentar el número de procesadores/cores).

Comúnmente también se dice "meter más hierro", porque antiguamente lo que se hacía era incrementar los recursos hardware del sistema. Hoy en día en sistemas virtualizados, estos recursos se pueden modificar, dependiendo del virtualizador, en "caliente", por lo que no sería necesario reiniciar el servicio.

Es el sistema más simple, ya que incrementando los recursos se espera que el problema se apacigüe o desaparezca, aunque esto no tiene por qué ser siempre así.

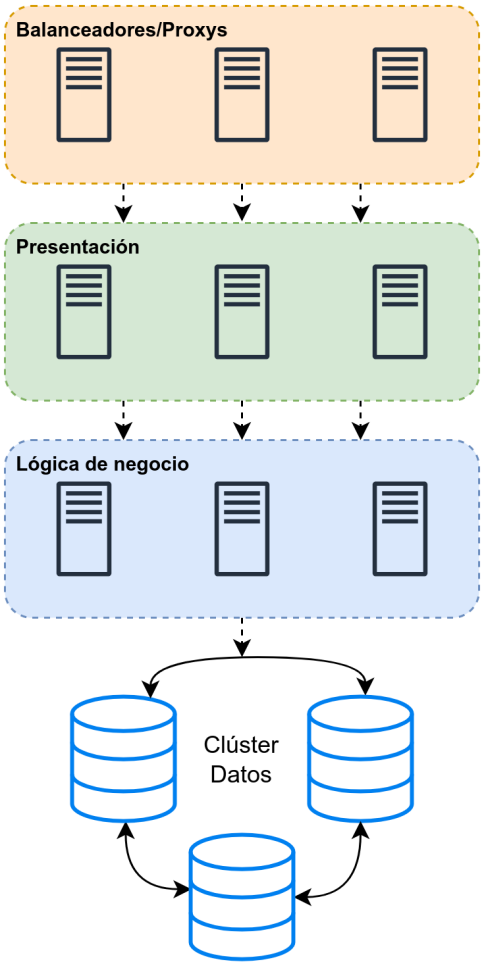
3.2. Escalado horizontal

El escalado horizontal trata de solventar el problema repartiendo la carga entre más nodos. Este proceso de escalado es más complejo y dependerá de la modularidad del servicio ofrecido. Es decir, la lógica de la aplicación debería haberse pensado desde el inicio para un sistema que escalará de manera horizontal en el futuro.

En principio, al realizar un escalado horizontal no existe una limitación de crecimiento, ya que siempre que se pueda repartir la carga entre servidores, no importará el número de ellos.

Dentro del escalado horizontal podríamos diferenciar dos tipos:

- **Escalado horizontal por capas**: Teniendo en cuenta lo visto previamente acerca de la arquitectura multicapa, en este modelo lo que se conseguirá es escalar cada capa de manera independiente. Podría realizarse de la siguiente manera:



- **Servidores frontales** (proxys) reciben las peticiones de la aplicación o del navegador y que balancean la carga y la mandan a servidores con la capa de presentación.
- Servidores con la **capa de presentación** que realizan las peticiones de negocio a los servidores correspondientes.
- Servidores de **lógica de negocio** que procesan las peticiones y piden los datos a un clúster de bases de datos.
- **Clúster de base de datos**.

- **Escalado horizontal de microservicios**: En el caso del escalado de microservicios, será similar al escalado horizontal, pero en este caso sólo se escalarán los microservicios necesarios.

  Debido a que todo es mucho más modular, es posible que quizá sólo sea necesario realizar el escalado de algunos microservicios, que quizá sean los más utilizados o los que requieren más tiempo de ejecución.

VII

Windows Subsystem for Linux

1. Introducción

WSL (del inglés *Windows Subsystem for Linux*) es una capa de compatibilidad que ha desarrollado Microsoft para correr ejecutables creados para sistemas Linux de manera nativa en Windows.

Desde el año 2019 la versión por defecto es la 2, que introdujo muchos cambios en el sistema, ya que esta versión corre dentro de una capa de virtualización creada a través de un **subconjunto del virtualizador Hyper-V**. Esto hace que el kernel que se está ejecutando sea mucho más compatible con los binarios de Linux que la versión 1. A pesar de usar virtualización, también mejora el rendimiento respecto a la versión anterior.

Por defecto WSL no viene instalado en el sistema en Windows 10, por lo que es necesario realizar la instalación para poder ejecutar las aplicaciones que deseemos (como Docker, por ejemplo). En versiones Windows 11 sí viene instalado.

2. Instalación

Para realizar la instalación necesitaremos una versión compatible de Windows (10 build 19041 o posterior). Hoy en día no debería ser problema si tenemos el sistema actualizado.

Para realizar la instalación necesitaremos de permisos de administrador, y lo realizaremos, para mayor comodidad, desde una consola de PowerShell o la nueva terminal de Windows. Para ello, abrimos la consola con permisos de administrador y ejecutamos:

Instalación de WSL en Windows 10

```
PS C:\Users\ruben> wsl --install
```

Tras la instalación es necesario reiniciar el sistema para que aplique cambios y levante los servicios necesarios. Una vez reiniciado nos aparecerá una ventana donde nos pedirá que introduzcamos el usuario y la contraseña para el sistema Linux recién instalado.

Información

Por defecto, la distribución que se instala es Ubuntu.

3. Configuraciones

Tras realizar la instalación podremos observar que Windows ha realizado una serie de configuraciones para adecuar la nueva instalación del servicio.

- Al tener las instancias levantadas, se genera un nuevo interfaz de red Hyper-V, con una red 172.25.240.0/20.

- Tal como se ha dicho, en WSL-2 las instancias realmente son máquinas virtuales Hyper-V. La configuración de las instancias se encuentran en el directorio `AppData` del usuario que las crea. Por ejemplo, para Debian, se encuentra el disco duro dentro de `./AppData/Local/Packages/TheDebianProject…/LocalState/ext4.vhdx`

  ¡Atención!

  El directorio AppData está oculto por defecto en el explorador de ficheros de Windows.

- Dentro de las instancias se puede acceder al disco duro de Windows a través de `/mnt/c`, o la unidad correspondiente.

- Desde Windows se puede acceder al sistema de ficheros de las instancias a través del explorador de ficheros, ya que nos aparecen las instancias que tenemos creadas.

Para poder realizar una configuración general de todo el ecosistema WSL se puede realizar desde una aplicación (disponible desde verano del 2024):

Aplicación de configuración de WSL

En esta aplicación se pueden modificar aspectos tan interesantes como:

- Procesadores lógicos dentro del WSL
- Tamaño máximo de la memoria
- Modo de red (NAT, mirrored, o proxy)

3.1. Configuración avanzada

Para realizar una configuración avanzada, existe una documentación desde dos puntos de vista:

- `wsl.conf`: fichero de configuración que se sitúa en el directorio `/etc` en las distribuciones. **Esta configuración sólo afecta en la distribución correspondiente** donde se haya realizado. Más adelante veremos un ejemplo para el uso de Docker.
- `.wslconfig`: **fichero en el directorio del usuario de Windows**. Este fichero tendrá la configuración que afectará a todas las distribuciones que hayamos instalado con WSL 2.

4. WSL con usuarios no privilegiados

WSL hace uso de ciertas características que necesitan de permisos de administrador. En caso de no tener permisos de administrador, por defecto sólo se hará uso de WSL versión 1, por lo que el rendimiento de los subsistemas es peor.

Por lo tanto, para poder utilizar WSL2 **se necesita tener acceso a los credenciales de administrador** y ejecutar los siguientes comandos:

Usar WSL2 en usuarios no privilegiados

```
PS C:\Users\usuario> wsl --update

PS C:\Users\usuario> wsl --set-default-version 2
```

5. Comandos útiles

Una vez realizada la instalación, existen ciertos comandos que nos pueden ser útil a la hora de hacer uso de WSL. No se van a detallar todos, ya que con `wsl -help` obtendremos la ayuda del comando y muchas más opciones.

Mostrar todas las distribuciones que se pueden instalar

```
PS C:\Users\ruben> wsl -l -o
```

Instalar una distribución Debian

```
PS C:\Users\ruben> wsl --install -d Debian
```

Mostrar las distribuciones instaladas

```
PS C:\Users\ruben> wsl -l -v
```

Ejecutar una distribución instalada y entrar en ella

```
PS C:\Users\ruben> wsl -d Debian
ruben@DESKTOP-1RVJ3UP:/mnt/c/Users/ruben$
```

Terminar/Apagar una distribución

```
PS C:\Users\ruben> wsl -t Debian
```

Apagar todas las instancias

```
PS C:\Users\ruben> wsl --shutdown
```

Eliminar una distribución instalada

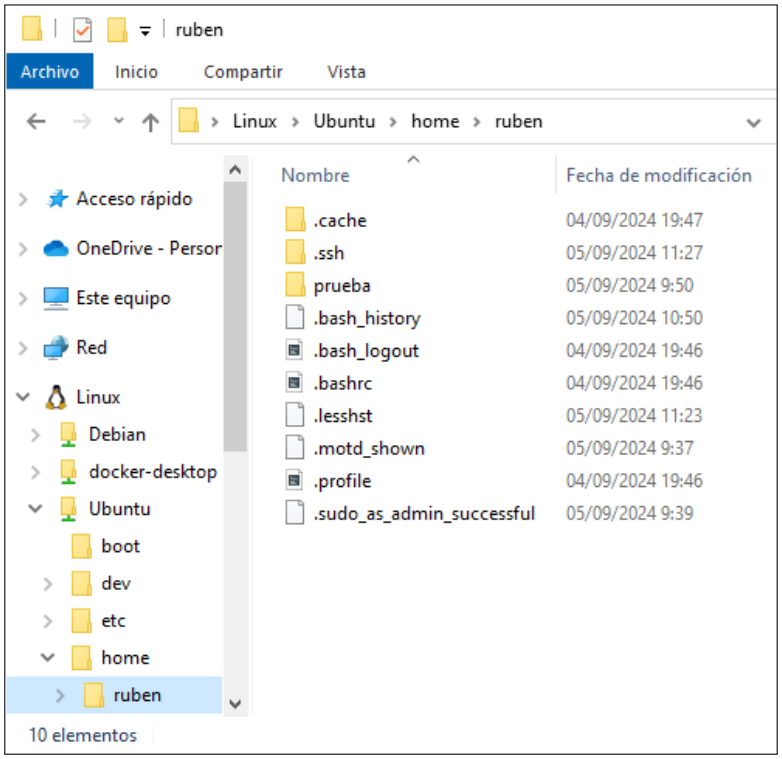
```
PS C:\Users\ruben> wsl --unregister Debian
```

Clonar una instancia de una distribución instalada

```
PS C:\Users\ruben> wsl --export Ubuntu ubuntu.tar

PS C:\Users\ruben> wsl --import Ubuntu2 ubuntu2_files ./ubuntu.tar
```

6. Acceder al sistema de ficheros de los subsistemas

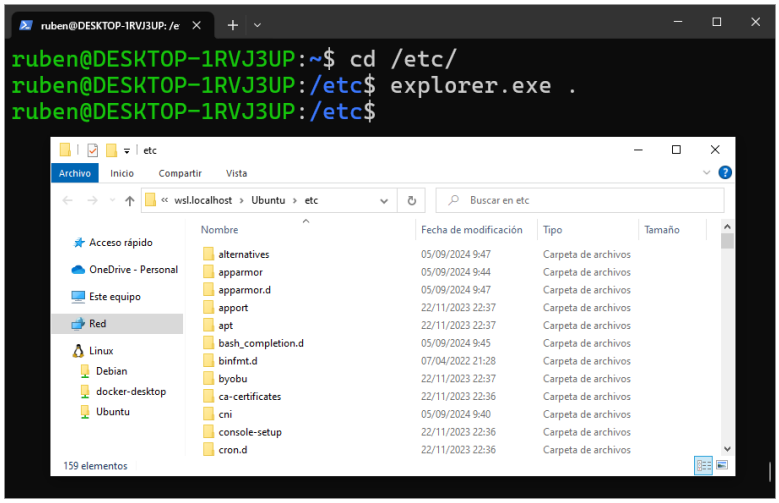
Microsoft ha creado la posibilidad de poder acceder al sistema de ficheros de los Linux que levantemos con WSL a través del explorador de ficheros de Windows. Esto permite copiar/pegar ficheros entre las distribuciones que hayamos creado y el propio sistema base. En la siguiente imagen se ve el explorador de archivos con tres subsistemas Linux creados:



Contenido de "/home/ruben" de Ubuntu desde el explorador Windows

Como alternativa, desde dentro del subsistema Linux podemos ejecutar aplicaciones Windows, por lo

que estando en cualquier ruta, podemos llamar al comando `explorer.exe .` que nos abrirá el explorador de Windows en esa misma ruta:



Abrir explorador Windows desde Linux

6.1. Rendimiento de los sistemas de ficheros en WSL

Por cómo funciona WSL y la gestión de sistemas de ficheros entre el sistema anfitrión Windows y el subsistema Linux, tenemos que tener claro que **existen dos sistemas de ficheros independientes, pero accesibles entre ellos**:

- **Sistema de ficheros de Windows**: Es el sistema de ficheros de nuestro equipo Windows. Hay que tener en cuenta, que cuando entramos al subsistema Linux, por defecto nos encontramos en ese mismo sistema de ficheros:

Al entrar al subsistema Linux, estamos en el sistema de ficheros de Windows

```
PS C:\Users\ruben\Desktop> wsl -d Ubuntu
ruben@DESKTOP-1RVJ3UP:/mnt/c/Users/ruben/Desktop$
```

Tal como se puede ver, al entrar en Ubuntu, la ruta en la que nos encontramos es `/mnt/c/Users/ruben/Desktop`, que es el sistema de ficheros de Windows (`C:`) **montado en la ruta de Linux** `/mnt/c`. Por eso, desde Linux tendremos acceso a todo el sistema de ficheros de Windows desde esa ruta.

- **Sistema de ficheros del subsistema Linux**: La máquina virtual de Linux que hemos creado tiene su propio sistema de ficheros, que como en cualquier Linux, está en `/`.

Pasamos al sistema de ficheros real de Linux

```
ruben@DESKTOP-1RVJ3UP:/mnt/c/Users/ruben/Desktop$ cd
ruben@DESKTOP-1RVJ3UP:~$ pwd
/home/ruben
```

Tal como se puede ver, con el comando `pwd`, ahora nos encontramos en el sistema de ficheros de

Linux real.

A la hora de hacer uso de aplicaciones en el susbsistema Linux, es recomendable hacerlo dentro del sistema de ficheros de Linux, no en el sistema montado, debido a que el rendimiento en el sistema montado es mucho peor. Por tanto, nos debemos asegurar que la aplicación está en la ruta correcta.

¡Cuidado!

Usar el sistema de ficheros de Windows montado en el subsistema Linux perjudica el rendimiento.

7. Docker dentro de WSL

Si queremos tener Docker dentro de un subsistema Linux, existen dos posibilidades completamente diferenciadas:

- Utilizar **Docker Desktop en Windows**. Docker Desktop usará WSL por debajo y tenemos la posibilidad de que los subsistemas hagan uso del Docker engine creado instalado en Docker Desktop. Este es el modo aconsejado por la documentación de Microsoft.
- Instalar el Docker Engine dentro de un subsistema Linux.

Este último método lo explicamos a continuación.

7.1. Instalar Docker Engine en WSL

En algunos casos nos puede interesar no hacer uso de Docker Desktop, porque lo que queremos es tener la posibilidad de un control total de Docker, como si de una instalación de máquina virtual completa de Linux se tratara. Es por ello que debemos realizar una pequña modificación en el funcionamiento del subsistema Linux correspondiente.

¡Atención!

Es más sencillo hacer uso de Docker Desktop en lugar de este sistema.

Supongamos que hemos creado el subsistema Linux de la distribución Ubuntu, deberemos entrar en ella, y tendremos que crear un fichero en `/etc/wsl.conf` con el siguiente contenido.

Configuración del fichero wsl.conf

```
[boot]
systemd=true
```

Salimos de la distribución y tenemos que forzar su reinicio. Una vez realizado estos pasos, si volvemos a entrar en la instancia, `systemd` estará funcionando y por tanto podremos instalar y hacer uso del Docker Engine como si fuese una máquina virtual creada al modo tradicional.

VIII

Introducción a Docker

1. Introducción

Hoy en día es muy habitual hacer uso de los sistemas de contenedores, siendo el más conocido Docker en el mundo del desarrollo de *software*. Este sistema trae consigo una serie de ventajas que veremos más adelante, que nos permite asegurar, entre otras cosas, que las versiones utilizadas en el entorno de producción son las mismas que durante las etapas de desarrollo.

En este documento se va a explicar cómo realizar la instalación y configuración de un sistema basado en contenedores Docker para poder arrancar servicios, y ciertas configuraciones que son necesarias conocer.

2. Sistemas de contenedores

Los sistemas de contenedores son un método de virtualización (conocido como "virtualización a nivel de sistema operativo"), en el que se permite ejecutar sobre una capa virtualizadora del núcleo del sistema operativo distintas instancias de "espacio de usuario".

Este "espacio de usuario" (donde se ejecutarán aplicaciones, servicios...) se les denomina **contenedores**, y aunque pueden ser como un servidor real, están bajo un mecanismo de aislamiento proporcionado por el *kernel* del sistema operativo, y sobre el que se pueden aplicar límites de espacio, recursos de memoria, de acceso a disco...

Información

Un contenedor es un espacio de ejecución de servicios al que se les puede aplicar límites de recursos (como la memoria, el acceso a disco…)

Desde el punto de vista del usuario, que un servicio se ejecute en una máquina virtual o en un contenedor es indistinguible. En cambio, desde el punto de vista de un administrador de sistemas o de un desarrollador, el uso de contenedores trae consigo una serie de ventajas que veremos en apartados posteriores.

2.1. Un poco de historia

Aunque está muy en boga el despliegue de aplicaciones haciendo uso de contenedores, no es un concepto nuevo, ya que lleva existiendo desde la década de los 80 en sistemas UNIX con el concepto de chroot.

Chroot, también conocido como "jaulas chroot", permitían ejecutar comandos dentro de un directorio sin que, en principio, se pudiese salir de dicha ruta. Tenía muy pocas restricciones de seguridad, pero era un primer paso al sistema de contenedores.

LXC nace en 2008 utilizando distintas funcionalidades del kernel Linux para proveer un entorno virtual donde poder ejecutar distintos procesos y tener su propio espacio de red. Con LXC nacen distintas herramientas para controlar estos contenedores, así como para crear plantillas y una **API que permite interaccionar con LXC** desde distintos lenguajes de programación.

Ha habido otras tecnologías en Linux, como OpenVz, pero nos centraremos en Docker, ya que es lo más conocido actualmente.

2.2. Qué es un contenedor y cómo se crea

Para entender qué es un contenedor dentro de la infraestructura Docker y cómo se crea, tenemos que diferenciar distintos conceptos:

- **Imagen Docker**
- **Contenedor Docker**

A continuación se van a detallar en profundidad.

2.2.1. Imágenes Docker

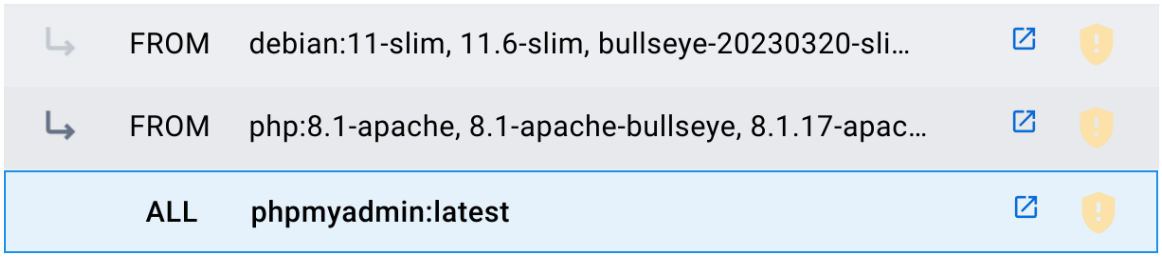
Para crear un contenedor necesitamos hacer uso de una **"imagen", que es un archivo inmutable (no modificable) que contiene el código de la aplicación que queremos ejecutar y todas sus dependencias necesarias**, para que pueda ser ejecutada de manera rápida y confiable independientemente del entorno en el que se encuentre.

Las imágenes, debido a su origen **sólo-lectura**, se pueden considerar como **"plantillas"**, que son la representación de una aplicación y el entorno necesario para ser ejecutada en un momento específico en el tiempo. **Esta consistencia es una de las grandes características de Docker**.

Información

Una imagen contiene el servicio que nos interesa ejecutar junto con sus dependencias, y son independientes del servidor donde se ejecuta.

Una imagen puede ser creada utilizando otras imágenes como base. Por ejemplo, la imagen de PHPMyAdmin empaqueta la aplicación PHPMyAdmin sobre la imagen **PHP** (versión 8.1-apache), que a su vez hace uso de la imagen **Debian** (versión 11-slim).



Jerarquía de imágenes usadas por PHPMyAdmin. Fuente.

A las imágenes creadas se les suele añadir etiquetas (**tags**) para diferenciar versiones o características internas. Cada creador determina las etiquetas que le interesa crear. Por ejemplo:

- **latest**: Se le denomina a la última imagen creada.
- php:**8.1-apache**: Indica que en esta imagen PHP la versión es la 8.1 y además cuenta con Apache.

Podemos utilizar imágenes públicas descargadas a través de un ***registry*** público, que no es otra cosa que un repositorio de imágenes subidas por la comunidad. El *registry* principal más utilizado es Docker Hub.

Información

Las imágenes Docker pueden ser públicas o privadas y se almacenan en un repositorio llamado *registry*, siendo el más conocido Docker Hub

Se pueden crear nuestras propias imágenes privadas, que pueden ser almacenadas en nuestros equipos o a través de un **registry privado** que podemos crear (también existen servicios de pago).

2.2.2. Contenedores Docker

Un contenedor Docker es un **entorno de tiempo de ejecución virtualizado donde los usuarios pueden aislar aplicaciones**. Estos contenedores son unidades compactas y portátiles a las que se les puede aplicar un sistema de limitación de recursos.

Información

Un contenedor se crea a través de una imagen, es la versión ejecutable de la misma que se crea en un entorno virtualizado

Un contenedor se crea a través de una imagen y es la versión ejecutable de la misma. Lo que se hace es crear una capa de escritura sobre la imagen inmutable, donde se podrán escribir datos. Se pueden crear un número ilimitados de contenedores haciendo uso de la misma imagen base.

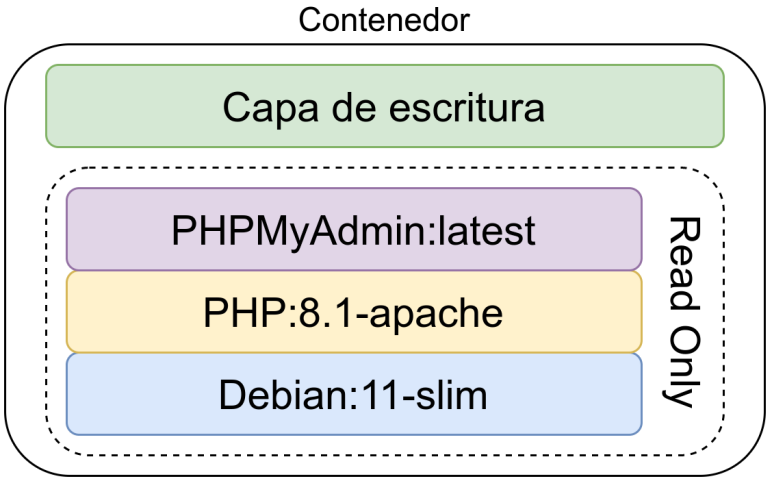


Imagen “interna” de un contenedor

La capa de escritura no es persistente y se pierde al eliminar el contenedor, es decir, **los datos de un contenedor se eliminan al borrar el contendor**. Para evitar este comportamiento se puede hacer uso de un volumen persistente de datos, de esta manera esos datos no se pierden.

¡Cuidado!

Los datos creados dentro de un contenedor se borran al eliminar el contenedor

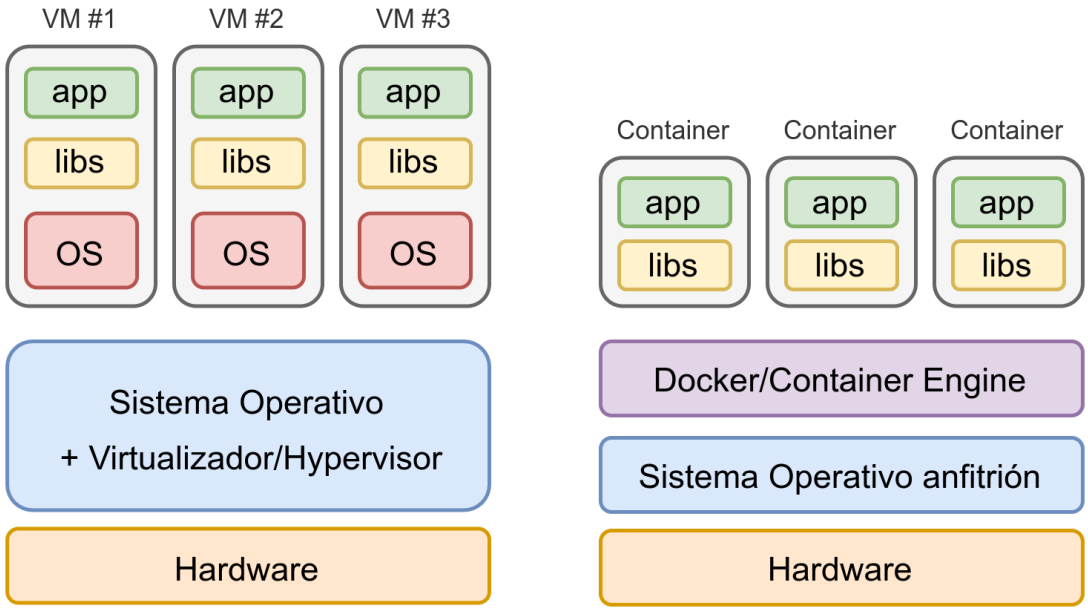
2.3. Contenedores vs. Máquinas virtuales

El uso de máquinas virtuales está muy extendido gracias a que cada vez es más sencillo crearlas. Esto no quiere decir que siempre sea la mejor opción, por lo que se va a realizar una comparativa teniendo en cuenta distintos aspectos a la hora de realizar un desarrollo con máquinas virtuales y con sistemas de contenedores.

2.3.1. Infraestructura

La creación de máquinas virtuales nos permite crear entornos aislados en los que poder instalar el Sistema Operativo que más nos interese y con ello poder instalar el software y los servicios que necesitemos.

Las máquinas virtuales se virtualizan a nivel de hardware, donde debe existir un Sistema Operativo con Hypervisor que permita dicha virtualización. Por otro lado, los contenedores se virtualizan en la capa de aplicación, haciendo que este sistema sea mucho más ligero, permitiendo utilizar esos recursos en los servicios que necesitamos hacer funcionar dentro de los contenedores.



Infraestructura Máquinas Virtuales vs Docker

En la imagen se puede apreciar una comparativa diferenciando cómo quedaría una infraestructura de 3 aplicaciones levantadas en distintas máquinas virtuales o en distintos contenedores.

Tal como se puede ver en la imagen, **al tener cada servicio en una máquina virtual separada**, se va a tener que virtualizar todo el Sistema Operativo en el que se encuentre, con el consiguiente **coste de recursos (memoria RAM y disco duro) y con el coste en tiempo de tener que realizar la configuración y securización del mismo**.

Información

Usando contenedores la infraestructura se simplifica notáblemente

Por otro lado, en un sistema de contenedores, cada contenedor es un servicio aislado, en el que sólo tendremos que preocuparnos (en principio) de configurar sus parámetros.

2.3.2. Ventajas durante el desarrollo

A la hora de desarrollar una aplicación es habitual hacer pruebas utilizando distintas versiones de librerías, *frameworks* o versiones de un mismo lenguaje de programación. De esta manera, podremos ver si nuestra aplicación es compatible.

Cuando se hace uso de una máquina virtual dependemos de las versiones que tiene nuestra distribución y es posible que no podamos instalar nuevas versiones u otras versiones en paralelo.

Por ejemplo, la última versión de PHP actualmente es la 8.4.11 y de Apache la 2.4.65:

- En **Debian 12** sólo se puede instalar PHP 8.2.29 y Apache 2.4.62.
- En **Ubuntu 24.04** la versión de PHP es la 8.3.6 y la de Apache la 2.4.58.

Con Docker, podremos levantar contenedores con distintas versiones del servicio que nos interese en paralelo para comprobar si nuestra aplicación/servicio es compatible.

Información

Con Docker es posible levantar servicios con distintas versiones en paralelo

Por otro lado, si un desarrollador quiere utilizar un sistema operativo distinto, no se tendrá que preocupar de si su distribución tiene las mismas versiones. O en el caso de usar Windows/Mac, no tener que estar realizando instalaciones de las versiones concretas.

2.3.3. Ventajas durante la puesta en producción

Ligado al apartado anterior, durante la puesta en producción es obligatorio hacer uso de las mismas versiones utilizadas durante el desarrollo para asegurar la compatibilidad.

¡Cuidado!

Para asegurar la compatibilidad en producción, siempre se debe usar la misma versión de los servicios que en desarrollo

Si tenemos un servidor que no está actualizado, o en el mismo servidor tenemos distintas aplicaciones que requieren utilizar distintas versiones de software, en un entorno de máquinas virtuales se hace muy complejo, ya que lo habitual será tener que instalar nuevas máquinas virtuales.

¡Atención!

No siempre es posible tener distintas versiones del mismo software en un mismo servidor

En un entorno con contenedores, al igual que se ha comentado antes, esto no es problema.

2.3.4. Rapidez en el despliegue

Ligado a todo lo anterior, realizar el despliegue de un entorno de desarrollo/producción es más rápido utilizando contenedores, sin importar el sistema operativo en el que nos encontremos.

Información

El despliegue con contenedores es más rápido.

Más adelante se verá cómo realizar el despliegue de distintos servicios haciendo uso de un único comando.

3. Docker

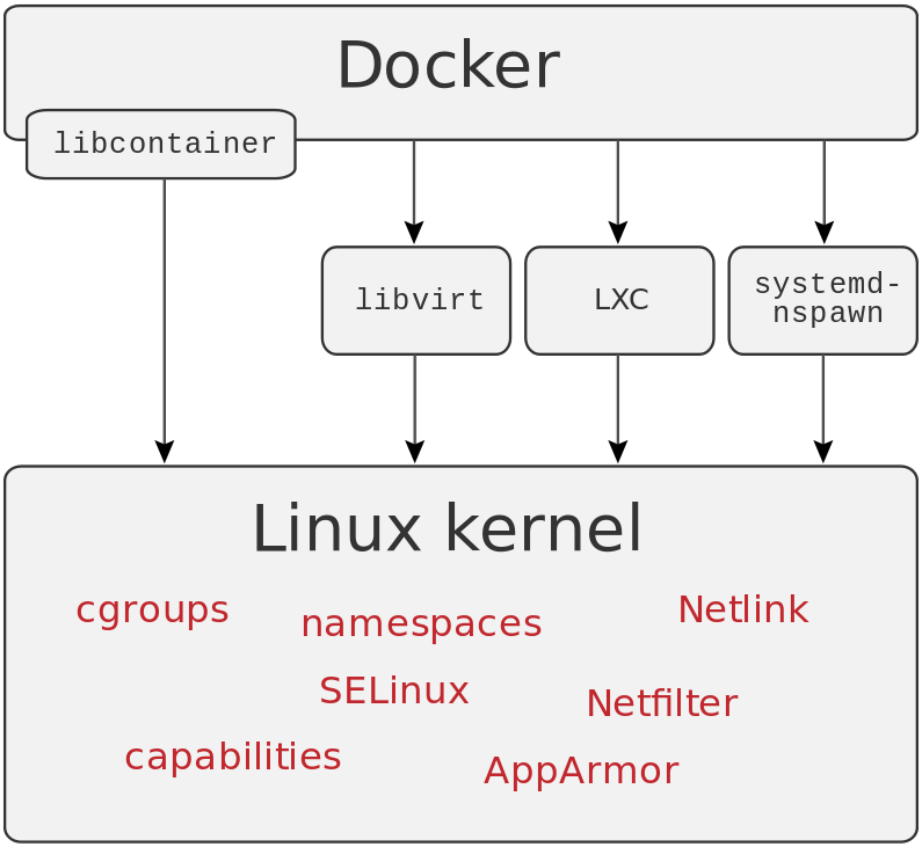
Docker es un proyecto de Software Libre nacido en 2013 que permite realizar el despliegue de aplicaciones y servicios a través de contenedores de manera rápida y sencilla, tal como veremos más adelante.

Estos contenedores proporcionan una capa de abstracción y permiten aislar las aplicaciones del resto del sistema operativo a través del uso de ciertas características del kernel Linux.

Dentro del contenedor, se puede destacar el aislamiento a nivel:

- Árbol de procesos
- Sistemas de ficheros montados
- ID de usuario
- Aislamiento de recursos (CPU, memoria, bloques de E/S...)
- Red aislada

Al igual que sucede con otro tipo de *software*, para que Docker haga uso de todas estas características, está construido haciendo uso de otras aplicaciones y servicios.



Tecnologías usadas por Docker. Fuente: Wikipedia

En el 2015 la empresa Docker creó la ***Open Container Initiative***, proyecto actualemente bajo la Linux Foundation, con la intención de diseñar un estándar abierto para la virtualización a nivel de sistema operativo.

3.1. Instalación

Dependiendo del sistema operativo en el que nos encontremos, Docker tiene la opción de instalarse de distintas maneras. En sistemas operativos GNU/Linux cada distribución tiene un paquete para poder realizar la instalación del mismo.

Instalación de Docker en Ubuntu

```
ruben@vega:~$ sudo apt install docker.io
```

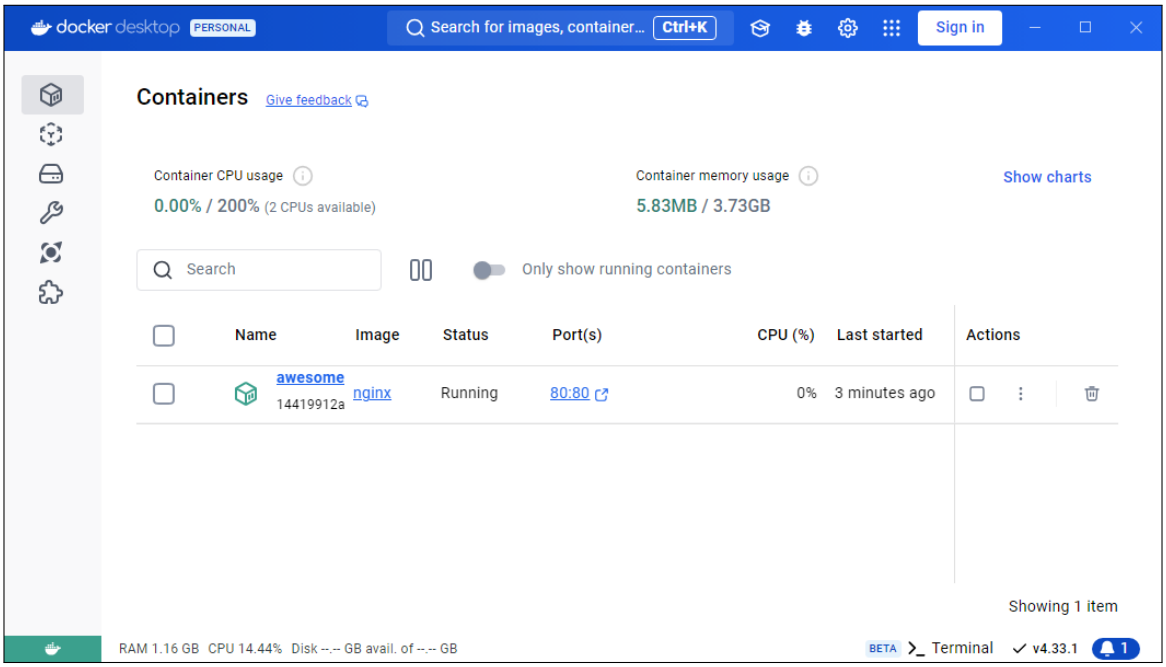
¡Cuidado!

El nombre del paquete en Ubuntu y Debian es “docker.io”.

3.1.1. Instalación en Windows y/o Mac

En sistemas Windows y MacOS existe la opción de instalar Docker Desktop, una versión que utiliza una máquina virtual para simplificar la instalación en estos sistemas. De todas maneras, también se instala el

CLI para poder usar los comandos que veremos a continuación.



Docker Desktop

En caso de Windows, se requiere tener las extensiones de virtualización habilitadas en la BIOS/UEFI y una de estas dos opciones, que habrá que configurar antes de instalar Docker Desktop:

- Usar WSL2.
- Usar Hyper-V y el sistema de contenedores de Windows.

3.2. Configuración

Tras realizar la instalación veremos cómo el servicio Docker ha levantado un interfaz nuevo en nuestra máquina, cuya IP es **172.17.0.1/16**, siendo el direccionamiento por defecto.

Nueva IP en el equipo

```
ruben@vega:~$ ip a
...
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
    link/ether 02:42:9c:1f:e2:90 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
      valid_lft forever preferred_lft forever
```

Esta IP hará de **puente** (similar a lo que sucede con las máquinas virtuales) cuando levantemos contenedores nuevos. Los contenedores estarán dentro de ese direccionamiento 172.17.0.0/16, por lo tanto, aislados de la red principal del equipo.

Información

Los contenedores que levantemos estarán en la red 172.17.0.0/16

3.3. Usar Docker con usuario sin privilegios

Para poder hacer uso de Docker con un usuario sin permisos de root/administrador, se debe añadir a los usuarios no privilegiados dentro de un grupo. Dependiendo de dónde usemos Docker, tendremos que realizarlo de una manera u otra.

3.3.1. Linux

En este caso, el grupo que debe tener el usuario es "**docker**", que se lo podemos añadir al usuario de distintas maneras:

- Editar el fichero `/etc/group`, y añadir el usuario al grupo
- Ejecutar estos comandos que ponemos a continuación:

Añadir el grupo docker al usuario correspondiente

```
ruben@vega:~$ sudo addgroup ruben docker
[sudo] password for ruben:
Adding user `ruben' to group `docker' ...
Adding user ruben to group docker
Done.
ruben@vega:~$ newgrp docker
```

A partir de ahora ya se puede hacer uso de Docker con el usuario al que hayamos añadido al grupo.

3.3.2. Windows

Para que un usuario en Windows pueda usar Docker Desktop tiene que pertenecer al grupo "**docker-users**". Para añadirlo, desde un PowerShell **con permisos de administrador**, ejecutaremos:

Añadir al usuario "usuario" al grupo docker-users

```
PS C:\Users\ruben> net localgroup "docker-users" "usuario" /add
```

3.4. Primeros pasos

El comando `docker` tiene muchas opciones, por lo que es recomendable ejecutarlo sin parámetros. De esta manera se pueden ver todas las opciones y una ayuda simplificada para cada una de ellas.

Algunas de las opciones del comando docker

```
ruben@vega:~$ docker
Usage:  docker [OPTIONS] COMMAND

Management Commands:
builder     Manage builds
completion  Generate the autocompletion script for the specified shell
config      Manage Docker configs
container   Manage containers
context     Manage contexts
image       Manage images
manifest    Manage Docker image manifests and manifest lists
network     Manage networks
node        Manage Swarm nodes
plugin      Manage plugins
secret      Manage Docker secrets
service     Manage services
stack       Manage Docker stacks
swarm       Manage Swarm
system      Manage Docker
trust       Manage trust on Docker images
volume      Manage volumes

Commands:
...
```

Para cada una de estas opciones, se le puede añadir el parámetro `--help` para mostrar la ayuda. Hay un segundo apartado que se ha cortado, en el que se incluyen más comandos.

Para asegurar que el servicio Docker está funcionando, podemos hacer uso de `docker info`, que nos mostrará mucha información acerca del servicio. Pero si lo que queremos es comprobar si tenemos algún contenedor corriendo, es más sencillo hacer `docker ps` (que es la versión simplificada de `docker container ls`):

Comprobar estado de Docker y contenedores levantados

```
ruben@vega:~$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES

ruben@vega:~$ docker container ls
```

```
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

En este caso, como no hay ningún contenedor levantado, sólo muestra las cabeceras de las columnas del listado.

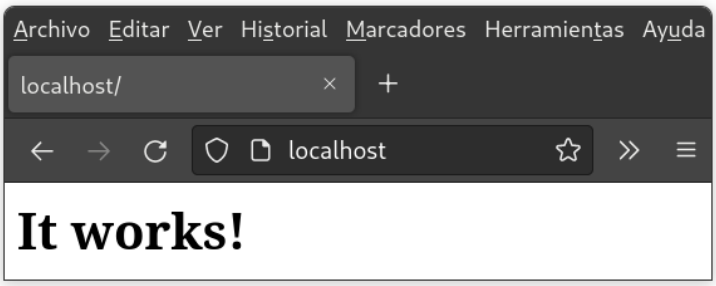
3.5. Levantando nuestro primer contenedor

Es momento de crear nuestro primer contenedor. Para ello, dado que se está usando la consola, hay que hacer uso del comando `>_ docker` con una serie de parámetros. En este caso se ha optado por levantar el servicio **Apache HTTPD**:

>_ Levantando el primer contenedor

```
ruben@vega:~$ docker run -p 80:80 httpd
AH00558: httpd: Could not reliably determine the server's ...
AH00558: httpd: Could not reliably determine the server's ...
[Fri Mar 24 18:25:14.194246 2023] [mpm_event:notice] ...
[Fri Mar 24 18:25:14.194347 2023] [core:notice] [pid  ...
172.17.0.1 - - [24/Mar/2023:18:25:41 +0000] "GET / HTTP/1.1" 304 -
```

Vemos los logs del servicio Apache al arrancar y si vamos al navegador a la dirección http://localhost muestra lo siguiente:



Y para entender lo que hace el comando, los parámetros son:

- **docker**: Cliente de consola para hacer uso de Docker.
- **run**: Ejecuta un comando en un nuevo contenedor (y si no existe lo crea).
- **-p 80:80**: Publica en el puerto 80 del servidor el puerto 80 utilizado en el contenedor. Se puede pensar que es como hacer un **port-forward** en un firewall.
- **httpd**: Es la **imagen** del contenedor que se va a arrancar. En este caso, la imagen del servidor Apache HTTPD.

Y si vemos qué muestra el estado de docker, vemos cómo aparece el contenedor levantado.

Comprobar estado de Docker y contenedores levantados

```
ruben@vega:~$ docker ps
CONTAINER ID   IMAGE     COMMAND       CREATED          STATUS          PORTS                NAMES
a1c3362b0d6c   httpd     "httpd-..."   3 seconds ago    Up 2 seconds    0.0.0.0:80->80/tcp   great
```

En la columna PORTS se puede apreciar cómo aparece que se ha levantado el puerto **`0.0.0.0:80`** (escucha en el puerto 80 para cualquier IP del sistema operativo) que es una redirección al puerto **`80/TCP`** interno del contenedor.

3.6. Contenedores en *background* y más opciones

Tal como se puede ver en el ejemplo anterior, el contenedor se queda en primer plano, viendo los logs del Apache. Esto para ver qué es lo que está sucendiendo durante el desarrollo puede ser útil, pero lo ideal es que el contenedor arranque en modo ***background***, y cuando necesitemos vayamos a ver los logs.

A continuación se va a arrancar un nuevo contenedor de Apache con nuevos parámetros:

Crear un contenedor Web en el puerto 8080

```
ruben@vega:~$ docker run --name mi-apache -d -p 8080:80 httpd
```

Los nuevos parámetros son:

- **`--name mi-apache`**: De esta manera se le da un nombre al contenedor, para poder identificarlo de manera rápida entre todos los contenedores creados.
- **`-d`**: Este parámetro es para hacer el ***detach*** del comando, y de esta manera mandar a ***background*** la ejecución del contenedor.
- **`-p 8080:80`**: Publica en el puerto 8080 del servidor el puerto 80 utilizado en el contenedor. Se puede pensar que es como hacer un **port-forward** en un firewall.

3.7. Parar, arrancar y borrar contenedores

Hasta ahora hemos aprendido a crear contenedores, pero en ciertos momentos nos puede interesar parar un contenedor que no estemos utilizando, o una vez haya cumplido su función, borrarlo.

3.7.1. Parar contenedores

Para parar un contenedor, debemos conocer el nombre del mismo o su ID (que es único). Estos datos los podemos conocer a través del comando `docker ps`.

Con esto, podemos ejecutar:

Parar un contenedor

```
ruben@vega:~$ docker stop mi-apache
```

3.7.2. Arrancar un contenedor parado

Una vez parado un contenedor, o al reiniciar el servidor, si queremos arrancar un contenedor parado, debemos conocer también su ID o nombre.

Para visualizar todos los contenedors (tanto los arrancados como los parados), lo podemos hacer a través del comando `>_ docker ps -a`.

Gracias a ese listado, podemos volver a arrancar un contenedor que esté parado con `>_ docker start mi-apache`, siendo "mi-apache" el contenedor que queremos arrancar.

3.7.3. Borrar un contenedor

Si queremos borrar un contenedor, éste debe estar parado, ya que Docker no nos va a dejar borrar un contenedor que está en ejecución.

Es interesante borrar contenedores que hayamos creado de pruebas o contenedores que ya no se vayan a utilizar más, para de esta manera liberar recursos.

Para borrarlo, similar a los casos anteriores, se hará con `>_ docker rm mi-apache`.

4. Variables de entorno

Algunos contenedores tienen la opción de recibir variables de entorno al ser creados. Estas variables pueden afectar al comportamiento del contenedor, o para ser inicializado de alguna manera distinta a las opciones por defecto.

El creador de la imagen Docker puede crear las variables de entorno que necesite para después utilizarlas en su aplicación. A modo de ejemplo, se va a utilizar la imagen de la aplicación PHPMyAdmin.

A continuación se van a crear 2 contenedores de PHPMyAdmin, diferenciados por el puerto, el nombre, y la variable de entorno **PMA_ARBITRARY**:

- El primer contenedor va a estar en el puerto 8081, se le va a dar el nombre "myadmin-1" y no va a tener la variable de entorno incializada.
- El segundo contenedor usará el puerto 8082, llamado "myadmin-2" y la variable **PMA_ARBITRARY** inicializada a "1", tal como aparece en la documentación de la imagen de PHPMyAdmin.

Para ello, se han ejecutado los siguientes comandos:

>_ Creación de dos contenedores PHPMyAdmin

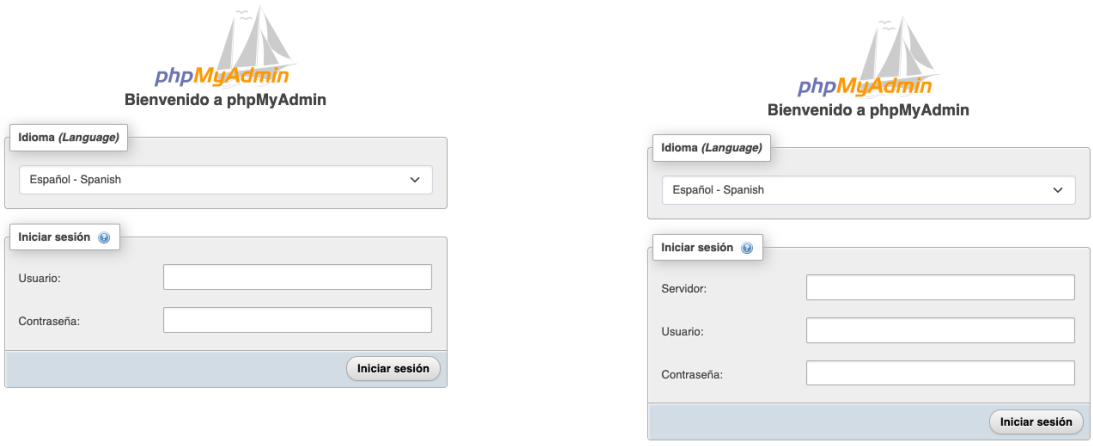
```
ruben@vega:~$ docker run --name myadmin-1 -d -p 8081:80 phpmyadmin

ruben@vega:~$ docker run --name myadmin-2 -e PMA_ARBITRARY=1 -d -p 8082:80 phpmyadmin
```

Tal como se puede ver, al segundo contenedor se le ha pasado un nuevo parámetro **-e**, que significa que lo que viene a continuación es una variable de entorno (en inglés **environment**). En este caso, la variable de

entorno es **PMA_ARBITRARY** que se ha inicializado a **1**.

Si ahora en nuestro navegador web apuntamos al puerto 8081 y al puerto 8082 de la IP de nuestro servidor, veremos cómo existe una ligera diferencia en el formulario que nos muestra la web.

En el formulario del puerto 8081 (donde no hemos inicializado la variable) sólo podemos indicar el usuario y la contraseña. Por el contrario, en el formulario del puerto 8082, al inicializar la variable **PMA_ARBITRARY**, y tal como nos dice la documentación de la imagen, nos permite indicar la IP del servidor MySQL al que nos queremos conectar.



A la izquierda formulario del puerto 8081, sin variable inicializada. A la derecha, puerto 8082 con variable inicializada.

Dado que una variable puede afectar al comportamiento (o la creación) del servicio que levantemos a través de un contenedor, es importante leer la documentación e identificar las variables que tiene por si nos son de utilidad.

Información

Es recomendable leer la documentación de las imágenes Docker para identificar las posibles variables de entorno que existen y ver si nos son útiles.

5. Volumen persistente de datos

Hasta ahora hemos levantado un contenedor a través de una imagen que levanta el servicio Apache, mostrando su página por defecto. Podríamos escribir en el contenedor la página HTML que nos interesase, pero hay que entender que **los datos de un contenedor desaparecen cuando el contenedor se elimina**.

Para que los cambios realizados dentro de un contenedor se mantengan, tenemos que hacer uso de los denominados **volúmenes de datos**. Esto no es más que **hacer un montaje de una ruta del disco duro del sistema operativo dentro de una ruta del contenedor**.

Estos volúmenes que le asignamos al contenedor pueden ser de dos tipos:

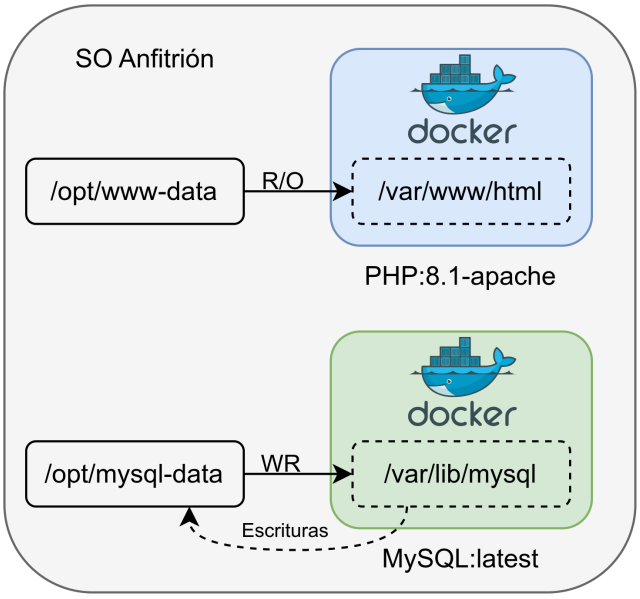
- **Sólo lectura**: Nos puede interesar asignar un volumen de sólo lectura cuando le pasamos ficheros de configuración o la propia web que queremos visualizar.

- **Lectura-Escritura**: En este caso se podrá escribir en el volumen. Por ejemplo, el directorio donde una base de datos guarda la información o una web donde deja imágenes subidas por usuarios.

De esta manera, tendremos que asignar el número de volúmenes necesarios a cada contenedor dependiendo de la imagen utilizada, el servicio que se levanta y lo que queremos hacer con los datos que le asignemos o generemos en el contenedor.

En la siguiente imagen se puede ver una infraestructura con dos contenedores y dos volúmenes:

- **Contenedor Web**: Se le asigna un volumen en modo **sólo lectura** cuya ruta original está en `/opt/www-data`, que dentro del contenedor está en `/var/www/html`.
- **Contenedor MySQL**: Dado que los datos de la base de datos deben ser guardados, en este caso se le asigna un volumen que permite escritura. Por lo tanto, lo que se crea dentro del contenedor en `/var/lib/mysql` realmente se estará guardando en el sistema operativo anfitrión en `/opt/mysql-data`.



Ejemplo de dos volúmenes asignados a distintos contenedores

5.1. Añadir volumen de escritura al crear un contenedor

Al añadir un volumen cuando creamos un contenedor hace que por defecto sea en modo lectura-escritura. Cualquier escritura realizada dentro del contenedor en la ruta especificada va a resultar en que el fichero se creará en la ruta indicada del sistema operativo anfitrión.

Añadir volumen de escritura al crear un contenedor

```
ruben@vega:~$ ls /opt/mysql-data
ruben@vega:~$ docker run -d -p 3306:3306 --name mi-db \
    -v /opt/mysql-data:/var/lib/mysql \
    -e MYSQL_ROOT_PASSWORD=my-secret-pw \
    mysql:latest
```

```
ruben@vega:~$ ls /opt/mysql-data
auto.cnf      client-key.pem      '#innodb_temp'      server-cert.pem   ...
```

Para este ejemplo se ha creado un contenedor usando la imagen de MySQL, al que se le ha asignado un volumen (**por defecto se asigna permitiendo la escritura**) y un parámetro necesario para realizar la posterior conexión con contraseña.

- **-v /opt/mysql-data:/var/lib/mysql**: A través del parámetro **-v** se le indica al contenedor que se le va a pasar un volumen. Posteriormente se le indica la ruta del sistema operativo anfitrión `/opt/mysql-data` que se montará dentro del contenedor en `/var/lib/mysql`.
- **-e MYSQL_ROOT_PASSWORD=my-secret-pw**: El parámetro "**-e**" sirve para pasarle al contenedor **variables de entorno**. En este caso, y tal como dice la web de la imagen MySQL, esta es la manera de asignar la contraseña del usuario **root** durante la inicialización de la base de datos.

Tras crear el contenedor, y asegurarnos que está levantado haciendo uso del comando `docker ps`, podemos realizar la conexión desde el sistema operativo anfitrión o desde cualquier otro lugar usando la contraseña indicada previamente.

Comprobar estado de MySQL

```
ruben@vega:~$ mysql -h127.0.0.1 -uroot -P3306 -p
    Enter password:
    ...
    MySQL [(none)]> show databases;
    +--------------------+
    | Database           |
    +--------------------+
    | information_schema |
    | mysql              |
    | performance_schema |
    | sys                |
    +--------------------+
    4 rows in set (0,007 sec)
```

5.2. Añadir volumen en modo sólo-lectura

A continuación se van a explicar los pasos para levantar un contenedor que contiene una web simple creada en PHP, que está alojada en la ruta `/opt/www-data` del sistema operativo anfitrión.

Añadir volumen sólo lectura al crear un contenedor

```
ruben@vega:~$ ls /opt/www-data
index.php
ruben@vega:~$ docker run -d -p 80:80 --name mi-web \
```

```
    -v /opt/www-data:/var/www/html:ro \
    php:8.2.4-apache
```

El parámetro nuevo asignado en la creación de este contenedor es:

- **`-v /opt/www-data:/var/www/html:ro`**: Para indicarle que le vamos a asignar un volumen siendo la ruta real en el sistema de ficheros del sistema operativo anfitrión `/opt/www-data` y la ruta destino **dentro del contenedor** y que va a ser en modo **read-only** `/var/www/html`.

Más adelante, cuando veamos cómo entrar dentro de un contenedor Docker, se podría usar el comando para ir a la ruta dentro del contenedor y comprobar que efectivamente está en modo sólo-lectura.

5.3. Entrar dentro de un contenedor Docker

Normalmente no suele ser necesario entrar dentro de un contenedor, ya que, tal como se ha dicho antes, cualquier modificación realizada dentro de él se perderá (salvo que sea dentro de un volumen persistente).

Aún así, para realizar pruebas o comprobaciones del correcto funcionamiento de una imagen puede ser interesante entrar dentro de un contenedor. Para ello, el comando a ejecutar es el siguiente:

Acceder a un contenedor

```
ruben@vega:~$ docker exec -it mi-db /bin/bash
```

Los parámetros utilizados son:

- **`exec`**: Indicamos que queremos ejecutar un comando dentro de un contenedor que está corriendo.
- **`-it`**: Son dos parámetros unidos, que sirven para mantener la entrada abierta (modo interactivo) y crear una TTY (consola)
- **`mi-db`**: Es el nombre del contenedor al que se quiere entrar. También se puede indicar el **ID** del contenedor.
- **`/bin/bash`**: el comando que queremos ejecutar. En este caso, una shell **bash**. En algunos casos esta shell no está instalada y debemos usar **`/bin/sh`**

Hay que tener en cuenta que dentro de un contenedor está el mínimo software posible para que la aplicación/servicio funcione, por lo que habrá muchos comandos que no existan.

6. Otros comandos útiles

Para obtener toda la información de un contenedor, incluido su estado, volúmenes utilizados, puertos, …

Obtener toda la información de un contenedor

```
ruben@vega:~$ docker inspect mi-db
```

Listar las imágenes descargadas en local. Al tener las imágenes en local, no hará falta volver a descargarlas, por lo que crear un nuevo contenedor que haga uso de una de ellas será mucho más rápido.

Listado de imágenes en local

```
ruben@vega:~$ docker image ls
REPOSITORY       TAG             IMAGE ID       CREATED         SIZE
php              8.2.4-apache    de23bf333100   3 days ago      460MB
httpd            latest          192d41583429   3 days ago      145MB
mysql            latest          483a8bc460a9   3 days ago      530MB
```

Borrar una de las imágenes que no se esté utilizando en ningún contenedor.

Borrar una imagen concreta

```
ruben@vega:~$ docker image rm httpd
```

Cuando un contenedor está en modo ***detached*** no aparecen los logs, por lo que para poder visualizarlos tenemos un comando especial para ello.

Ver los logs de un contenedor

```
ruben@vega:~$ docker logs mi-web -f
172.17.0.1 - - [26/Mar/2023:18:05:29 +0000] "GET / HTTP/1.1" 200 248
172.17.0.1 - - [26/Mar/2023:18:05:29 +0000] "GET / HTTP/1.1" 200 248
172.17.0.1 - - [26/Mar/2023:18:05:29 +0000] "GET / HTTP/1.1" 200 248
```

Listar los volúmenes existentes en el sistema. El listado muestra los que se están utilizando en contenedores (activos o parados) o los que se han utilizado en contenedores que ya no existen.

Listar volúmenes

```
ruben@vega:~$ docker volume ls
DRIVER    VOLUME NAME
local     0d6c400a6407f5cdea81a2f0158222fdd87d7f3b3e2b5969ca466d743fc71f5c
local     1d2f52018e17af0689e070a55337154c1dd68517c54435ecc24d597f7509d43c
local     6b72797227ef4708ca23ee1dfcb4b651b42eeacefd4166b898407ad4aadda10c
```

Si queremos realizar una limpieza de todos los recursos (contenedores, imágenes, volúmenes) que no se estén utilizando, se puede utilizar el siguiente comando.

Borrar recursos que no estén activos

```
ruben@vega:~$ docker system prune -a
WARNING! This will remove:
- all stopped containers
```

```
- all networks not used by at least one container
- all images without at least one container associated to them
- all build cache

Are you sure you want to continue? [y/N] y
```

¡Cuidado!

El comando anterior hace que se borren contenedores parados

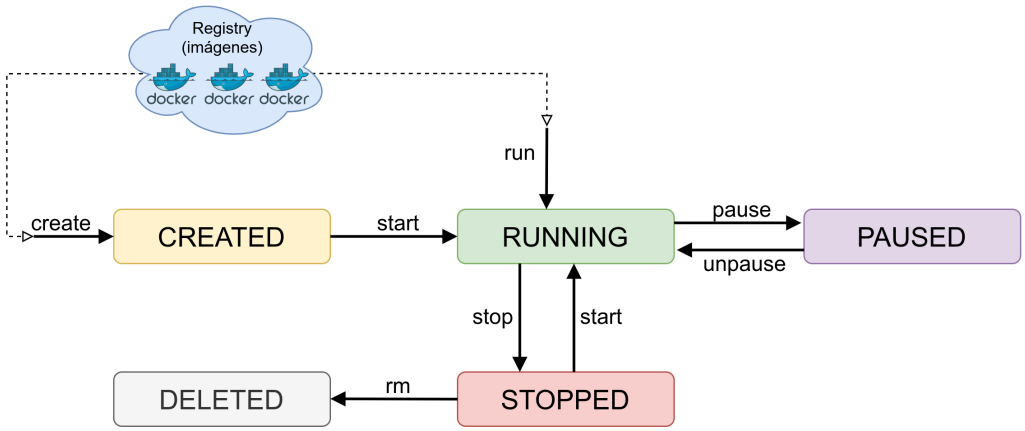
Para conocer las estadísticas de uso de cada contenedor.

Ver las estadísticas de los contenedores

```
ruben@vega:~$ docker stats
CONTAINER ID   NAME    CPU %   MEM USAGE / LIMIT     MEM %   NET I/O           BLOCK I/O   PIDS
2fcf97530766   mi-web  0.00%   54.62MiB / 15.47GiB   0.34%   9.44MB / 172kB    0B / 0B     6
413d6e9f590f   mi-db   0.55%   357.9MiB / 15.47GiB   2.26%   319kB / 280B      0B / 0B     38
```

7. Ciclo de vida de un contenedor Docker

Un contenedor tiene un ciclo de vida que puede pasar por distintos estados. Para pasar entre estados se debe realizar a través de distintos comandos de Docker.



Estados de un contenedor

En la imagen se representa los estados más básicos junto con los comandos para pasar entre ellos.

IX

Monitorización de servicios

1. Sistemas de Monitorización

El sistema de monitorización se encarga de recopilar información acerca del estado de los servidores de una infraestructura. Entre las métricas y datos que debe recopilar se encuentran:

- **Estado del servidor**: cantidad de RAM utilizada, estado de los discos duros, carga del servidor, sistema de ficheros, …
- Estado de los **servicios que tiene el servidor**: servidor web, número de procesos que tiene, bases de datos, conexiones existentes, cola de correos electrónicos para enviar si es un servidor de correo, … Dependiendo del servicio habrá que realizar unas comprobaciones u otras.
- **Infraestructura en la que se encuentran**: estado de la red, conexión a otros servidores, …
- **Estado del clúster**: en caso de que el servidor pertenezca a un clúster, hay que comprobar que el clúster se encuentra en perfecto estado.
- **Dependencias externas**: un servidor puede depender a su vez de otros, o de servicios externos, que deben de estar funcionando de manera correcta.

Normalmente en la monitorización actúa un **agente** (o servicio) instalado en el equipo monitorizado que obtiene la información requerida que se envía a un servidor central que recopila la información de toda la infraestructura monitorizada.

Esta información suele ser almacenada durante un periodo de tiempo determinado (un año, por ejemplo) para poder ser usada y comparar la situación de los servidores a lo largo del tiempo. Gracias a esta comparación temporal **se puede llegar a predecir el estado del servidor** a unos días/semanas vista y evitar problemas antes de que sucedan (no tener espacio en discos duros, mal funcionamiento de servicios por falta de RAM, …).

Información

La monitorización de servicios y equipos dentro de una infraestructura debe considerarse parte del proyecto, ya que es una parte muy importante de cara al mantenimiento del mismo.

1.1. Monitorización de servidores

Es habitual que los sistemas de monitorización funcionen en base a plantillas, que posteriormente se pueden asociar a los servidores monitorizados. Estas plantillas contendrán los servicios que deben ser monitorizados en cada tipo de servidor, ya que no es lo mismo monitorizar un servidor web o un servidor con un SGBD.

Para monitorizar un servidor lo habitual suele ser realizar las siguientes operaciones:

- **Comprobar conectividad con el servidor**: suele ser habitual que los servidores estén fuera de nuestra red (en un proveedor de Internet, en un cliente, en otra oficina…), por lo que es necesario

que exista conectividad de alguna manera para poder realizar la monitorización. En caso de no estar en nuestra red, el uso de una VPN es lo más utilizado.

- **Instalar un agente en el propio servidor**: Será el encargado de recopilar la información necesaria para mandarla al servidor central. Dependiendo del sistema de monitorización utilizado, necesitaremos un tipo de agente u otro. Algunos se encargan de realizar todas las comprobaciones y otros llaman a otros programas para realizar las comprobaciones y después mandar el resultado al servidor central.
- **Dar de alta el servidor en el sistema centralizado**: Tal como se ha dicho previamente, lo habitual es contar con un sistema centralizado en el que se tendrán todos los servidores y el estado de las comprobaciones realizadas. De ser así, habrá que darlo de alta, y para ello se necesitará:
  - **Nombre del servidor**: Un nombre que a simple vista identifique el servidor. Suele ser habitual poner el nombre del cliente también, y/o el tipo de servicio que preste.
  - **IP del servidor**: Para poder realizar la conexión al servidor.
  - **Plantillas asociadas**: En caso de utilizar un sistema que utilice plantillas, al dar de alta el servidor se le aplicarán las plantillas necesarias para que realicen todos los checks oportunos. Por ejemplo: plantilla de Servidor Linux + plantilla de Servidor web + Plantilla de MySQL.
  - **Puerto de conexión**: Los agentes de monitorización suelen contar con un puerto que queda a la escucha. Si hemos cambiado el puerto, habrá que indicarlo a la hora de dar de alta.
  - **Otras opciones**: Dependiendo del sistema de monitorización se podrán añadir muchas más opciones, como por ejemplo:
    - **Servidores de los que se depende**: Imaginemos que el servidor monitorizado depende a su vez de un router que también está monitorizado. Si el router cae, no llegaríamos al servidor, por lo que realmente es una caída por dependencia, aunque el servidor puede estar funcionando de manera correcta. Esto nos puede permitir crear “árboles de dependencias” de servidores.
    - **Periodos de monitorización**: Lo habitual es que un servidor esté monitorizado 24x7, pero quizá nos interese realizar cambios y que sólo se monitorice en unas horas determinadas (quizá el resto del tiempo está apagado).
    - …

1.2. Funcionamiento de la monitorización

Para conocer cómo funciona un sistema de monitorización lo mejor es que tomemos como ejemplo un tipo de servicio que queremos monitorizar. Como ejemplo se puede tomar las comprobaciones que queremos realizar a un SGBD (Sistema Gestor de Bases de Datos).

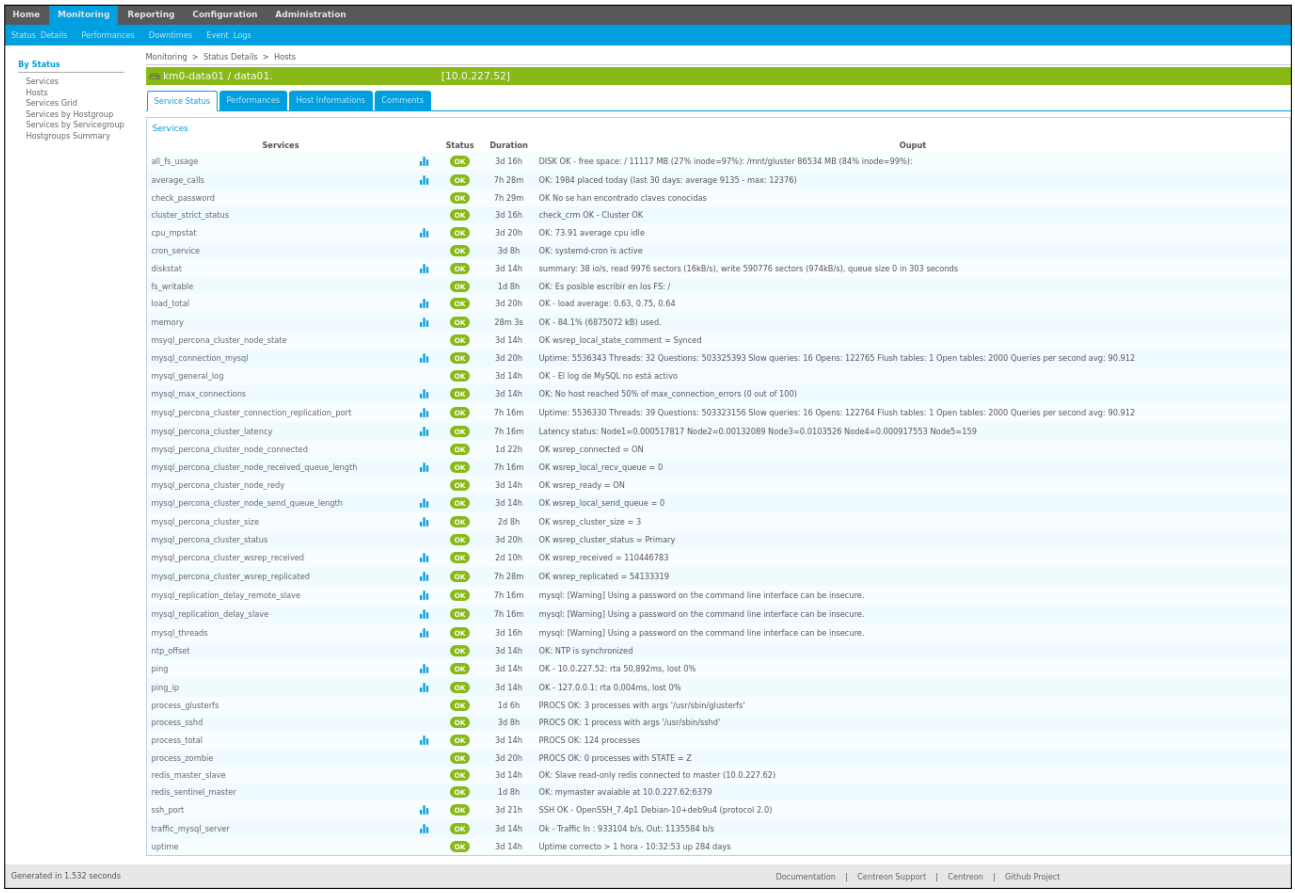
No será lo mismo realizar la monitorización de un servidor MySQL o de un Oracle, pero las comprobaciones

que queremos realizar en ellos deberían ser similares. Vamos a querer realizar la monitorización de las mismas comprobaciones: estado de las tablas en memoria, número de hilos en ejecución, número de *slow_queries*, …; pero los scripts ejecutados serán distintos.

Información

La monitorización dependerá del propio servicio que vayamos a monitorizar.

A continuación se puede ver el estado de un servidor monitorizado a través del sistema de monitorización Centreon:



Servidor monitorizado en Centreon

En la imagen anterior se puede comprobar un número de ***checks***, o comprobaciones, que se están realizando sobre un servidor concreto. Cada fila es una comprobación y contienen:

- **Nombre del check/servicio**: Un nombre para identificar qué es lo que se está comprobando con el check.
- **Icono para mostrar gráficas**: Algunos checks recibirán información que puede ser graficada para así poder observar patrones en el comportamiento del servidor. Por ejemplo: cantidad de RAM ocupada, número de procesos en el sistema, número de conexiones a un servidor, …
- **Estado del check**: Normalmente, tras realizar la comprobación, el check termina con uno de los

siguientes resultados:

- **OK**: El resultado obtenido es el correcto.
- **Warning**: El resultado obtenido está entre los márgenes de peligro. Es posible que de seguir así pase al estado siguiente:
- **Crítico**: El servicio devuelve un estado que es considerado crítico, lo que puede hacer que llegue a mal funcionamiento del mismo, o incluso que el servidor comience a dejar de funcionar (imaginemos que el servidor está con el 90 % de la RAM ocupada o de disco duro ocupado).
- **Indeterminado**: Por alguna razón, el *check* no se ha realizado, o el valor devuelto es indeterminado o no se puede saber en qué otro estado situarlo.

- **Duración del estado**: Para conocer cuánto tiempo lleva en el estado la comprobación obtenida. Lo ideal es que nunca haya estados que no sean OK y por lo tanto la duración de los mismos sea lo más alta posible.
- **Valor devuelto por la monitorización**: El valor real devuelto por la comprobación realizada. En base a este resultado se puede realizar las gráficas mencionadas previamente.

Información

El estado del servicio dependerá del valor devuelto por la monitorización.

Este resultado se cotejará con los valores que hayamos puesto para que sea considerado OK, Warning o Critical. Es decir, **en algunos casos el estado del servidor depende de los valores devueltos y de la baremación que le hayamos otorgado**.

Pongamos como ejemplo la monitorización de un SGBD:

- **El servicio del SGBD está funcionando**: Ahí no hay baremación posible. Si el servicio no está arrancado, es lógico pensar que el estado es crítico y que por tanto hay que ver qué ha ocurrido.
- **Número de conexiones en el SGBD**: El resultado devuelto será un número entero (que podremos graficar para obtener patrones). En este caso, podemos decidir los rangos para que el resultado sea OK, Warning o Critical. Es decir, si el resultado obtenido está por debajo del umbral de Warning, el sistema considerará que el estado es OK. Si está en dicho rango, será Warning y si está en el rango de Critical, así lo indicará.

Esta baremación y **estos rangos** se suelen aplicar también en las plantillas de los servicios. Hay que entender que también **pueden ser modificados y personalizados para un servidor concreto**. No es lo mismo que un SGBD tenga 500 conexiones simultáneas si tiene 8Gb de RAM o si tiene 128Gb (en el primer servidor se puede considerar que es un estado crítico mientras que en el segundo es lo esperado).

Cuando un *check* termina siendo un Warning o un Critical **es habitual que haya un sistema de alarmas configurado**. Dependiendo del sistema utilizado, notificará a los administradores mediante e-mail,

mensajería instantánea, SMS, … para que realicen un análisis lo antes posible y solucionen el estado del servicio.

Información

Los sistemas de monitorización suelen contar con un sistema de alarmas para que nos avise de los servicios caídos.

1.2.1. Monitorización básica

Tal como se ha comentado, en los servidores se suele realizar una monitorización del estado del mismo que suele ser común para todos, por lo que lo habitual suele ser tener una plantilla genérica para todos los servidores con la que se monitorizará:

- Cantidad de RAM utilizada
- Cantidad de memoria virtual utilizada
- Carga de la CPU
- Espacio libre en las unidades de disco duro
- Estado del sistema RAID del servidor (en caso de tenerlo)
- Cantidad de usuarios conectados a la máquina
- Estado de puertos de conexión (SSH, por ejemplo)
- Latencia hasta llegar al servidor
- …

Es cierto que no será lo mismo monitorizar un sistema GNU/Linux o un sistema Windows (ya que puede variar alguno de las comprobaciones a realizar), pero el estado general que queremos conocer es el mismo. Por lo tanto, lo habitual es tener dos plantillas, una específica para servidores Windows y otra para GNU/Linux.

1.2.2. Monitorización de Servicios

Aparte de la monitorización básica comentada previamente, necesitaremos monitorizar el estado de los servicios que pueda tener el servidor propiamente dicho. Para ello, de nuevo, se crearía una plantilla específica para cada tipo de Servicio que podamos tener en nuestro servidor.

No será lo mismo monitorizar un servidor que tenga un servidor web, un servidor de base de datos, un proxy… O puede que el servidor cuente con todos esos servicios.

Es por eso que a la hora de realizar la monitorización de un servidor **es muy importante conocer qué funciones desempeña cada servidor en la infraestructura a la que pertenece** y analizar los servicios que tiene arrancados para posteriormente ser monitorizados.

Información

Es muy importante conocer qué funciones desempeña cada servidor en la infraestructura a la que pertenece.

1.3. Tipos de monitorización

Existen varias maneras de realizar la monitorización de un servidor, y dependerá del gestor de monitorización que usemos (en caso de usar uno).

Es habitual que cuando nos referimos a sistemas de monitorización lo dividamos en dos grandes familias:

- Monitorización Activa
- Monitorización Pasiva

Estas dos maneras de monitorización suelen ser excluyentes, aunque algunos sistemas de monitorización permiten ambas, por lo que nos puede interesar usar una u otra dependiendo de la situación.

1.3.1. Monitorización pasiva

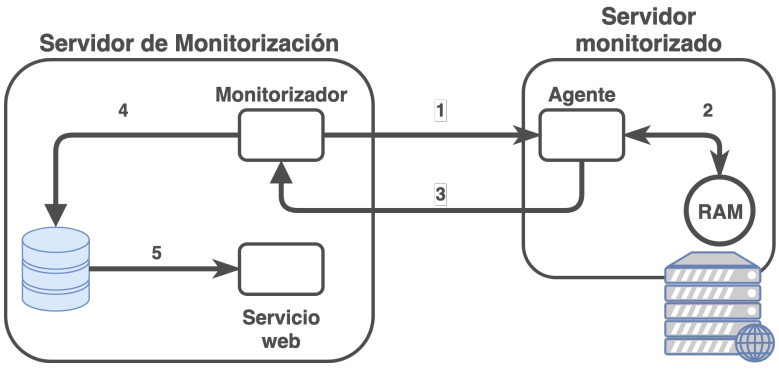
En la monitorización pasiva el servidor (u objeto monitorizado) es el encargado de mandar la información de manera periódica al servidor central. El agente instalado se ejecutará como una tarea programada cada cierto tiempo (habitualmente unos pocos minutos) e informará de la situación cambiante, de haberla, al servidor central.

Esta manera de monitorización es utilizada también cuando no hay un servidor central. En este caso, si la comprobación ha sido incorrecta, podría mandar un mail al administrador del servidor.

1.3.2. Monitorización activa

Suele ser la manera habitual de proceder de los sistemas que cuentan con un servidor centralizado de monitorización. El servidor de monitorización se encarga de preguntar al servidor, a través de la conexión con el agente, por la comprobación de alguno de los checks, y el agente devuelve la información.

A continuación se puede observar las etapas que existen en un sistema de monitorización activa utilizando un servidor de monitorización central:



Proceso de monitorización activa

Las etapas serían:

0. El sistema de monitorización tiene un scheduler (o planificador) que decide cuándo tiene que realizar cada comprobación (normalmente, cada pocos minutos).
1. El servicio encargado de monitorizar **establece conexión con el agente remoto** y le pide que compruebe un estado. En este ejemplo se ha optado por la RAM.
2. El agente en el servidor que se quiere monitorizar recibe la notificación y realiza una **comprobación local** (normalmente llamando a scripts locales) para obtener la cantidad de RAM ocupada, total y libre que tiene
3. El agente envía al sistema de monitorización el resultado obtenido en la ejecución de los scripts del paso anterior.
    1. El monitorizador al recibir el resultado, lo coteja con los rangos de baremación que tiene y decide si el check está en estado OK, Warning o Critical.
    2. Lo habitual es que si el resultado del servicio no es OK, se ejecute en el servidor de monitorización algún tipo de alarma (ya sea enviar un mail, sistema de mensajería, …) para notificar a los administradores.
4. El sistema de monitorización guarda en una base de datos los resultados obtenidos para así poder realizar posteriores análisis o comprobaciones temporales de los mismos.
5. Esos datos se suelen visualizar en una interfaz web, tal como hemos visto previamente.

Estos pasos son ejecutados de manera continuada en el servidor de monitorización para cada comprobación que se realiza en cada uno de todos los servidores que se monitorizan. Por lo tanto, se entiende que el propio servidor de monitorización también tiene que ser monitorizado ya que es de vital importancia que su estado sea óptimo.

1.3.3. Monitorización centralizada

Como ya se ha comentado, es el sistema habitual de monitorización. Las ventajas que podemos obtener al hacer uso de este sistema son muchas, pero se pueden destacar las siguientes:

- **Monitorización centralizada**: Aunque parezca obvio, el tener un único sistema en el que concentrar toda la información es muy útil y eficaz.
    - La alternativa sería tener una monitorización distinta en cada servidor.
- **Interfaz web**: Hoy en día suele ser habitual que los sistemas de monitorización tengan un servicio web en el que visualizar todos los datos obtenidos.
- **Sistema de plantillas**: De nuevo, es lo habitual, lo que hace que la gestión de monitorización de servidores sea más cómoda.
- **Gestión de usuarios**: Podremos tener usuarios que puedan ver unos servidores u otros, por lo que podemos tener equipos especializados en distintos grupos de monitorización y que sólo se enfoquen en ellos.

- Esto también es útil para dar acceso a los clientes a la monitorización de sus propios servidores.

1.3.4. Monitorización reactiva

La monitorización reactiva se puede definir como el sistema de monitorización que no sólo se encarga de comprobar y recibir el estado de los servidores, si no que también reacciona a los mismos para tratar de solucionar los problemas encontrados. Tras esta definición está la idea de que **existen ciertos fallos recurrentes que no siempre necesitan la intervención humana para solucionarse**, y que por tanto, se puede tratar de ejecutar antes de que sea considerado un problema real.

Como **ejemplo sencillo** se puede poner **el espacio libre en disco duro**. Imaginemos que se comprueba que apenas hay espacio en el disco duro de un servidor. En este caso, el sistema de monitorización recibirá que el servidor **está al 99.95 %** de espacio ocupado, y por tanto, en lugar de notificar a un humano indicando el estado crítico, **el sistema reacciona de manera automática tratando de liberar espacio**. Se habrá configurado previamente que en la reacción de este error trate de borrar ficheros temporales, vaciar papelera, limpiar ficheros de caché de ciertas rutas … Una vez hecho esto, se volverá a comprobar el estado del servidor. Si el espacio ocupado en disco duro ha bajado y está en modo OK no habrá que hacer nada más, y se habrá evitado que un administrador tenga que realizar dicha tarea. Si por el contrario el estado sigue siendo incorrecto, el sistema notificará el error para que se realice un análisis y se solucione el problema.

Como **ejemplo extremo** (que no suele ser habitual configurarlo así), imaginemos que **la RAM consumida por un SGBD es muy alta** y esté poniendo en peligro el estado del servidor, se podría configurar para que **el sistema reaccione reiniciando el SGBD para que libere la RAM** y vuelva a prestar servicio.

1.4. Gestores de monitorización

Hoy día existen muchos sistemas de monitorización, y dependiendo de nuestras necesidades deberemos optar por uno u otro. A continuación se expondrán varios ejemplos de gestores de monitorización basados en Software Libre, aunque la gran mayoría de ellos cuentan con un sistema dual. Es decir, se puede descargar y montarlo en tu propio servidor o puedes contratar a la empresa para que ellos tengan el servicio central:

- **Nagios**: Se puede considerar uno de los sistemas de monitorización más conocidos y del que se han basado otros. Generó mucha comunidad de administradores creando muchos scripts/plugins para hacerlos funcionar con él. Estos mismos scripts suelen ser utilizables en otros sistemas de monitorización.
- **Centreon**: Originalmente se creó como interfaz web para Nagios, pero poco a poco fue sustituyendo partes de Nagios hasta terminar siendo un sistema de monitorización completo. Existe la posibilidad de realizar la instalación por paquetes, descargar el sistema operativo en una ISO que te instala todo o incluso una máquina virtual con todo ya instalado y con configuración básica. (Demo).
- **PandoraFMS**: Sistema de monitorización creado por el español Sancho Lerena Urrea. Al igual que los anteriores, tiene sistema dual y la instalación se puede realizar por varios métodos.
- **Cacti**: Sistema más sencillo que los anteriores y habitualmente utilizado sólo en servidores sueltos,

es decir, no de manera centralizada.

- **Munin**: Igual que el anterior, ideal para monitorizar unos pocos servidores, ya que no se puede considerar un sistema centralizado como los primeros. Ver anexo de instalación de Munin.

Existen otros sistemas de monitorización basados “en la nube”, cuya funcionalidad es similar a lo expuesto previamente. Para hacer uso de estos sistemas nos descargamos un agente, lo instalamos y se encargará de mandar la información a los servidores de la plataforma contratada. Lógicamente, dependiendo del gasto realizado obtendremos más o menos servicios. Entre este tipo de servicios se pueden destacar:

- New Relic
- DataDog

X

Virtualbox

1. Virtualbox y adaptadores de red

1.1. Introducción

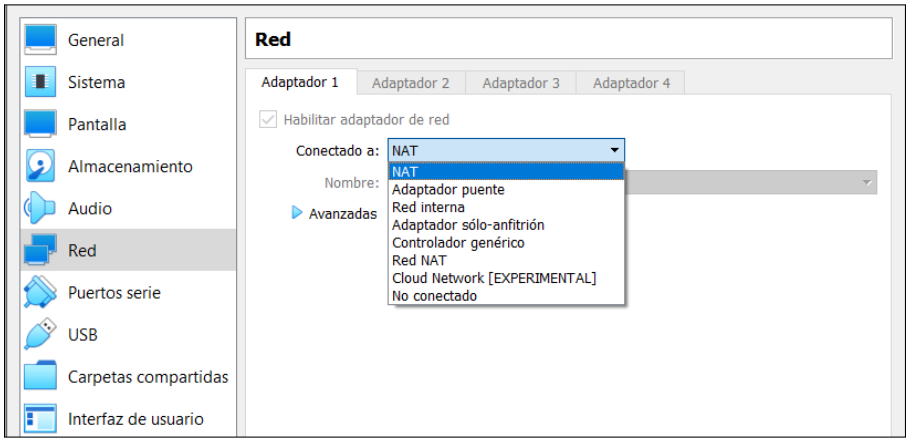
Virtualbox es una herramienta de virtualización para crear máquinas virtuales de manera sencilla. Es multiplataforma por lo que se puede utilizar en Windows, MacOS y Linux y aparte, es Software Libre.

Este documento no va a entrar en detalle en cómo se crean las máquinas virtuales, sino que va a explicar los distintos modos y adaptadores de red que puede tener una máquina virtual en este sistema de virtualización.

1.2. Adaptadores de red

Virtualbox permite que cada máquina virtual cuente con hasta cuatro adaptadores de red, lo que comúnmente se llaman interfaces o NIC (network interface controller).

Al crear las máquinas virtuales sólo tienen un único adaptador activo y suele estar configurado en modo NAT, pero tal como se ve a continuación, en el desplegable se puede ver que existen otras opciones:



En la documentación oficial aparece la explicación de los distintos modos, y es buena práctica leer y entender la documentación del software que utilizamos. También hay que entender que cada tipo de adaptador contará con una serie de ventajas y una serie de limitaciones que aparecen reflejadas en la documentación. Estos modos son comunes a otros sistemas de virtualización (VmWare, Proxmox, …), pero el nombre o el modo de uso puede variar así como las posibles limitaciones que puedan existir.

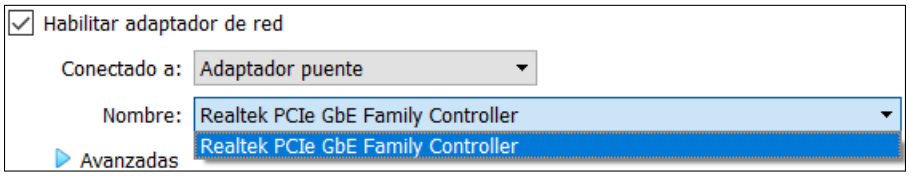
A continuación se va a dar una pequeña introducción a cada tipo de adaptador.

1.2.1. Adaptador puente

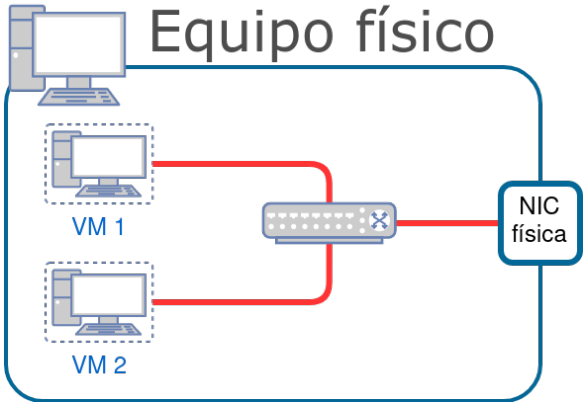
Es el tipo de adaptador que se usará si queremos que las máquinas virtuales aparezcan en la red física como si fueran un equipo más. Para poder entenderlo de mejor manera, podríamos pensar que este tipo de adaptador lo que hace es crear un “switch virtual” entre las máquinas virtuales y el interfaz físico, por lo que es como si fueran un equipo más en la red física.

Si el equipo físico anfitrión cuenta con más de un NIC (por ejemplo, en un portátil el NIC por cable y el NIC

wifi) tendremos que elegir en la máquina virtual sobre qué NIC queremos hacer el puente. En la siguiente imagen en el desplegable sólo se puede seleccionar un interfaz porque el equipo sólo cuenta con un NIC físico.



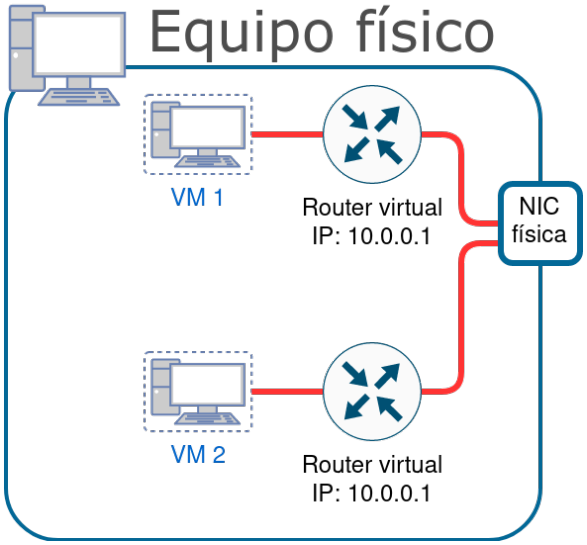
Es el método utilizado cuando virtualizamos servidores, ya que podrán dar sus servicios a toda la red.



Red como Adaptador Puente

1.2.2. NAT

Cada máquina virtual contará con su propio “router virtual” que hará NAT, y por eso todas las máquinas virtuales que usen este modo suelen tener la misma IP, pero no pertenecen a la misma red. Por defecto no se puede realizar conexión desde la red física al equipo virtualizado.



Red en modo NAT

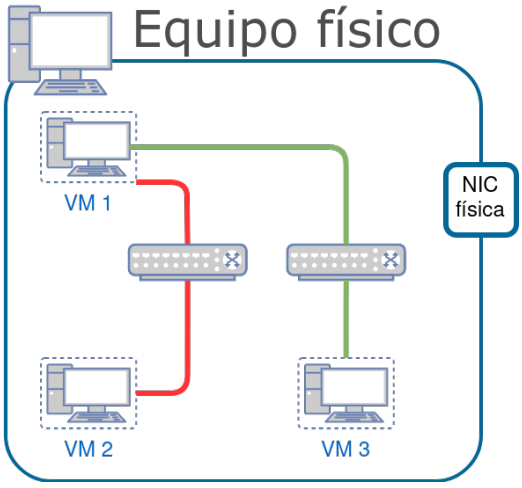
1.2.3. Red interna

Este tipo de adaptador lo que hace es “crear” un “switch virtual” que unirá las distintas máquinas virtuales que estén conectadas al nombre de esa red interna.

En el siguiente ejemplo la VM1 tiene 2 NICs, cada una con una red interna distinta. La VM2 tiene un NIC conectado a una de las redes internas creadas previamente y VM3 está conectada a la otra red interna.

Virtualbox no se encarga de dar IPs en estas redes, por lo que deberemos configurar cada interfaz de la máquina virtual con el direccionamiento que nos interese.

Este método se utiliza si queremos comunicar máquinas virtuales entre sí y que estén aisladas, ya que no podrán conectarse con el exterior, ni siquiera con el propio equipo físico.

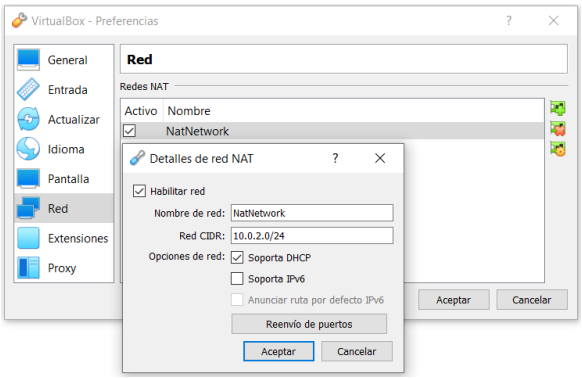


Red Interna

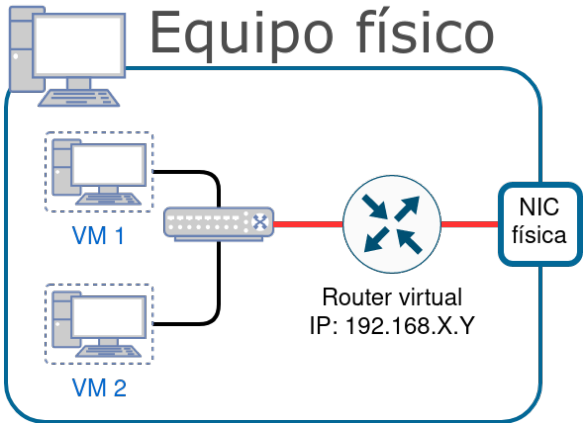
1.2.4. Red NAT

Podría definirse como una mezcla de NAT y red interna. Las máquinas virtuales podrán pertenecer a una única red, se podrán comunicar entre ellas, estarán detrás de un NAT de la red física y se podrán comunicar con el exterior.

Para poder usar ese modo hay que crear la “red NAT” en Virtualbox yendo a “*Archivo → Preferencias → Red*” y ahí se creará las redes NAT que queramos con el direccionamiento interno que nos interese.



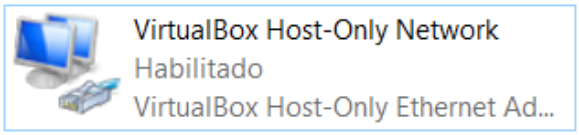
A la hora de crear la máquina virtual y elegir la opción “Red NAT” se podrá elegir entre las redes creadas en el paso anterior.



Red NAT

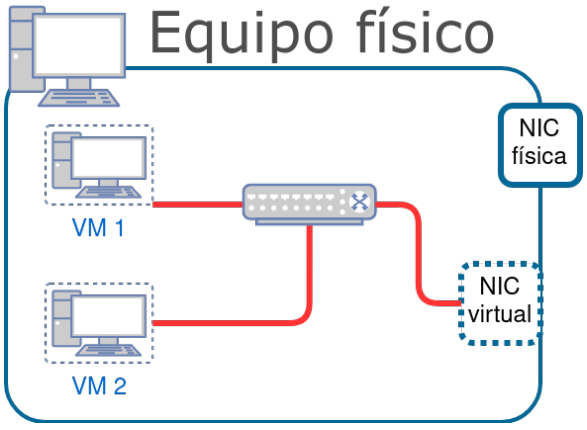
1.2.5. Adaptador sólo-anfitrión

Este tipo de adaptador es similar al de "red interna" pero con la posibilidad de comunicarse con el equipo físico anfitrión. En el equipo físico se crea un interfaz virtual y a través de él se podrá comunicar con las máquinas virtuales.



El direccionamiento que existe entre las VMs y el host se define en Virtualbox, dentro de "*Archivo* → ***Administrador de red de anfitrión***". Las máquinas virtuales podrán coger IP de ese direccionamiento por DHCP.

Mismo uso que "red interna" pero añadiendo la opción de comunicarnos con el host anfitrión.



Red en modo "Sólo Anfitrión"

1.3. Resumen de los adaptadores

A continuación se expone una tabla que resume los distintos tipos de adaptadores que existen y la conectividad posible entre las máquinas virtuales que usan esos adaptadores y el host anfitrión (fuente).

Modo \ Conectividad	VM → Host	VM ← Host	VM1 ← → VM2	VM → LAN real	VM ← LAN real
Adaptador puente	✔	✔	✔	✔	✔
NAT	✔	Redirección de puertos	✘	✔	Redirección de puertos
Red interna	✘	✘	✔	✘	✘
Sólo-anfitrión	✔	✔	✔	✘	✘

En la documentación se explica cómo realizar la redirección de puertos.

XI

Instalar Ubuntu Server 22.04

1. Instalar Ubuntu 22.04 LTS

En este anexo realizaremos la instalación de la distribución Ubuntu 22.04 LTS en su versión para servidores. En este anexo no se va a explicar cómo realizar la creación de una máquina virtual donde se aloja el sistema operativo, ya que existen distintos tipos de virtualizadores.

No se realizará una guía "paso a paso", sino que se centrará en las partes más importantes de la instalación y en las que más dudas puedan surgir.

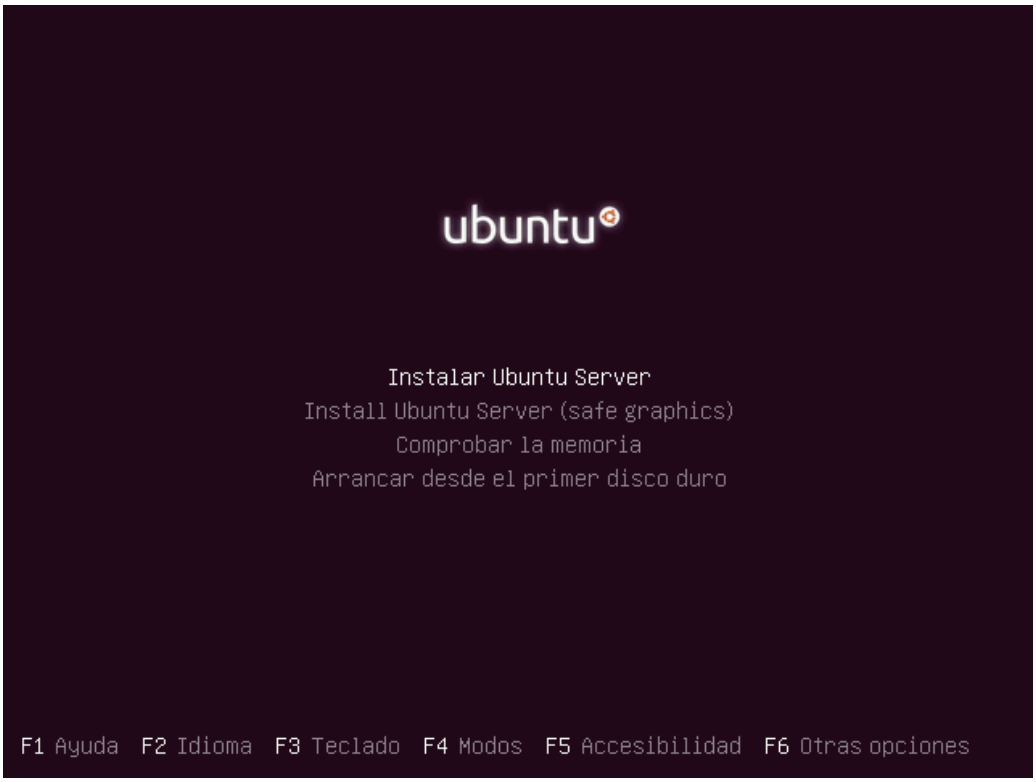
1.1. Descargar Ubuntu 22.04

La ISO la obtendremos de la web oficial y seleccionaremos la versión 22.04 LTS de Ubuntu Server. Esta ISO contendrá el sistema base de Ubuntu y nos guiará para realizar la instalación del sistema operativo.

Una vez descargada la ISO tendremos que cargarla en el sistema de virtualización elegido y arrancar la máquina virtual.

1.2. Instalar Ubuntu 22.04

Tras arrancar la máquina virtual nos aparecerá un menú para seleccionar el idioma durante la instalación y le daremos a "Instalar Ubuntu Server".



A partir de aquí comenzará el instalador y los pasos que nos aparecerán serán los siguientes (algunos de estos pasos puede que no estén 100 % traducidos al castellano):

1. Elegir el idioma del sistema
2. Actualización del instalador:

- Si la máquina virtual se puede conectar a internet, comprobará si existe una actualización del propio instalador de Ubuntu.
- Podemos darle a “Continuar sin actualizar”

3. Configuración del idioma del teclado
4. Configuración de la red
5. Configuración del proxy de red
6. Configuración del “mirror” o servidor espejo desde donde descargarse los paquetes de software para las actualizaciones posteriores.
7. Selección del disco duro donde realizar la instalación
8. Elegir el particionado de disco.
9. Configuración del perfil. Introduciremos el nombre de usuario, el nombre del servidor y la contraseña del usuario que vamos a crear.
10. Configuración de SSH Server. Aceptaremos que se instale el servidor SSH durante la instalación. En caso de no seleccionar esta opción, posteriormente podremos realizar la instalación.
11. “Featured Server Snaps”. En esta pantalla nos permite instalar software muy popular en servidores.

Una vez le demos a continuar, comenzará la instalación en el disco duro. Debido a que durante la instalación tenemos conexión a internet, el propio instalador se descarga las últimas versiones de los paquetes de software desde los repositorios oficiales.

Al terminar la instalación, tendremos que reiniciar la máquina virtual.

1.3. Post-instalación

Tras realizar el reinicio de la máquina virtual nos encontraremos con que el sistema arranca en el sistema recién instalado, y que tendremos que loguearnos introduciendo el usuario y la contraseña utilizadas en la instalación.

1.3.1. Actualización del sistema

Por si acaso, realizaremos la actualización del índice del repositorio, actualizaremos el sistema y en caso necesario realizaremos un nuevo reinicio:

Actualizar Ubuntu

```
root@ubuntu:~# apt update
...
root@ubuntu:~# apt upgrade
...
```

Con estos comandos nos aseguramos que el sistema está actualizado a los últimos paquetes que están en el repositorio.

1.3.2. Poner IP estática

Debido a la configuración de red de nuestro servidor, la IP está puesta en modo dinámica, esto quiere decir que nuestro equipo ha cogido la IP por configuración de DHCP de nuestra red. Debido a que un servidor debe de tener IP estática, tenemos que realizar la modificación adecuada para ponerle la IP estática que mejor nos convenga. Para ello editaremos el fichero de configuración situado en la siguiente ruta: `/etc/netplan/00-installer-config.yaml`

Lo modificaremos para que sea parecido a (siempre teniendo en cuenta la IP y gateway de nuestra red):

Configurando IP estática en Ubuntu

```
network:
  ethernets:
    enp0s3:
      dhcp4: false
      addresses: [192.168.1.199/24]
      routes:
      - to: default
        via: 192.168.1.1
      nameservers:
        addresses: [8.8.8.8,1.1.1.1]
  version: 2
```

El fichero de configuración que hemos modificado es de tipo YAML, que es un formato de texto que suele ser utilizado en programación o en ficheros de configuración. Este tipo de ficheros tiene en cuenta los espacios para el uso de la identación, y no suele permitir el uso de tabuladores.

Para aplicar los cambios realizados en el fichero de configuración deberemos ejecutar el siguiente comando que aplicará los cambios:

Aplicar configuración de IP

```
root@ubuntu:~# netplan apply
```

2. Administración remota

En informática no siempre tenemos los equipos que administramos en nuestra oficina. Pueden estar en otro edificio, la oficina de un cliente, en internet … por lo tanto no siempre es posible acceder de manera física a ellos, y por tanto entra en juego la **administración remota**.

Podemos definir la administración remota como el sistema que nos permite realizar ciertas acciones "lanzadas" desde nuestro equipo local pero que serán ejecutadas en un equipo remoto.

Se pueden diferenciar varios tipos de sistemas dentro de la administración remota, pero nos vamos a centrar en los siguientes:

- **Cliente remoto**: Lanzamos una acción a ejecutar desde un equipo remoto a través de algún tipo de interfaz o comando (que viajará a través de un protocolo securizado) y esperaremos a la respuesta.
- **Acceso remoto**: En este caso lo que hacemos es conectarnos al equipo a través de un protocolo que nos va a permitir administrarlo como si estuviésemos delante de él.

Todos estos sistemas pueden ser complementarios, y puede que podamos administrar un mismo servicio de todas estas maneras, por lo que queda a nuestra disposición elegir el mejor método en cada momento.

Por otro lado, dependiendo de qué tipo de administración vayamos a llevar a cabo, o el protocolo que utilice, tendremos que tener acceso al servidor de alguna manera (ya sea conexión directa o mediante VPN).

Información

Dependiendo de la administración remota que realicemos, necesitaremos conexión directa o mediante VPN al equipo que nos queremos conectar.

Por último, también debemos de conocer el tipo de protocolo que vamos a utilizar al realizar la conexión remota y por dónde va a pasar esa comunicación. Siempre hay que premiar la seguridad de la comunicación, y más cuando esta puede pasar por redes no controladas. Por lo tanto, deberemos asegurar que el protocolo utilizado es seguro, o en caso contrario, securizarlo de alguna manera.

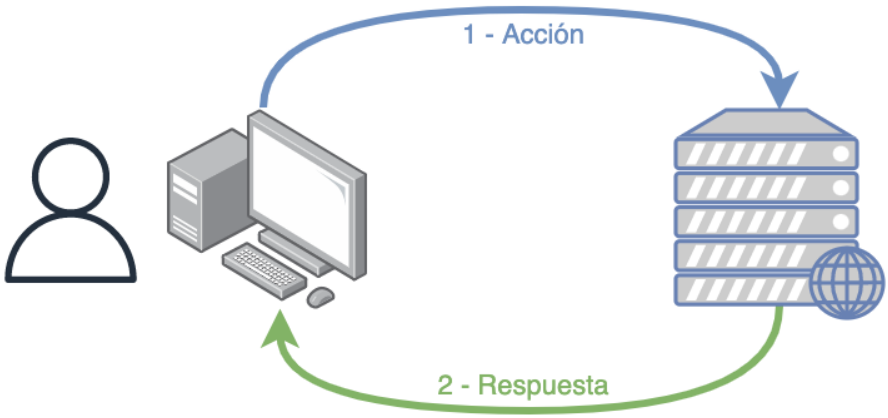
¡Cuidado!

Siempre debemos confirmar que la comunicación que se realiza para la administración remota viaja cifrada.

Más adelante veremos cómo securizar una comunicación no segura realizando un túnel mediante el protocolo SSH en entornos GNU/Linux.

2.1. Cliente remoto

Este sistema de administración permite enviar acciones al equipo remoto a través de un protocolo establecido, y dependiendo de la acción ejecutada se esperará una respuesta o no.



Hoy en día es muy habitual este tipo de sistemas a través de disintos **CLI** (*client line interface*) o **GUI** (*graphic user interface*) que nos permiten administrar servicios remotos. Por ejemplo:

- **https://www.mysql.com/MySQL**: El sistema gestor de base de datos MySQL cuenta con un cliente para realizar la conexión, ya sea desde el propio equipo o desde un equipo remoto.

  Este cliente se puede ejecutar desde línea de comandos, aunque también viene integrado en distintos interfaces gráficos como MySQL Benchmark, Dbeaver, ...

- **AWS CLI**: Es el interfaz de línea de comandos para poder administrar de manera remota los servicios contratados en la nube AWS de Amazon.

- **Gcloud CLI**: Similar al caso anterior pero esta vez para Google Cloud.

- **Remote Server Administration Tools for Windows 10**: En este caso se trata de un interfaz gráfico (**GUI**) que nos permite administrar un Windows Server desde un equipo Windows 10.

Antes de poder realizar la conexión remota con el cliente **debemos configurar un sistema de autenticación** para que el servicio remoto acepte las peticiones enviadas. En algunos casos será usando unos sistemas de certificados y en otros introduciendo un usuario y contraseña que establecerá una sesión temporal.

En el caso de AWScli y GCloud no nos estamos conectando directamente a nuestros servidores alojados en esas nubes, si no que lanzamos la orden a un "proxy" que verificará nuestros credenciales, verá los permisos que tenemos y después realizará la acción solicitada.

2.2. Acceso remoto

Este sistema permite acceder al sistema y podremos administrarlo como si nos encontrásemos delante. Dependiendo del sistema la conexión nos permitirá interactuar de alguna de las siguientes maneras:

- **CLI**: Mediante una conexión de línea de comandos. Es el caso más habitual en servidores GNU/Linux y la conexión se hace a través del protocolo seguro **SSH**.

- **GUI**: Podremos obtener un interfaz gráfico con el que veremos lo que está ocurriendo en pantalla en ese momento. En este caso, dependiendo del sistema, existirán distintas opciones, pero vamos a nombrar dos de ellas:

- **RDP**: Es el protocolo de escritorio remoto de Microsoft que transmite la información gráfica que el usuario debería ver por la pantalla, la transforma en el formato propio del protocolo, y la envía al cliente conectado. El problema es que este sistema desconecta al usuario que está logueado para poder hacer uso del escritorio remoto.
- **VNC**: En inglés *Virtual Network Computing*, es un servicio con estructura **cliente-servidor** que permite visualizar el escritorio del servidor desde un programa cliente. En este caso, no existe desconexión del usuario que está logueado y por tanto podrá ver lo que le están realizando de manera remota.

  Es muy habitual que los equipos de usuarios ya tengan la instalación del servidor hecha, para que de esta manera, en caso de incidencia, poder realizar la conexión remota sin que el usuario tenga que realizar ninguna acción.

A continuación se va a detallar algunos de los métodos mencionados.

2.3. SSH

SSH es un protocolo de comunicación segura mediante cifrado cuya función principal es el acceso remoto a un servidor. La arquitectura que utiliza es la de cliente-servidor.

Aunque el uso más habitual de SSH es el acceso remoto, también se puede utilizar para:

- Securizar protocolos no seguros mediante la realización de túneles.
- Acceder a un equipo saltando a través de otro.

Estas funcionalidades las veremos más adelante.

2.3.1. Servidor SSH

En el servidor al que nos queramos conectar deberá estar instalado el demonio/servicio SSH, conocido como **sshd**. Es habitual que ya esté instalado en sistemas GNU/Linux, pero de no ser así deberemos usar el sistema de paquetes de nuestra distribución para hacer la instalación. El nombre suele ser **openssh-server**.

Este servicio por defecto se pondrá a la escucha en el puerto 22/TCP:

SSHd escuchando en puerto 22

```
ruben@vega:~$ sudo ss -pntaln
State   Recv-Q   Send-Q   Local Address:Port   Peer Address:Port   Process
LISTEN  0        128      0.0.0.0:22           0.0.0.0:*           users:(("sshd",pid=1122,fd=3))
LISTEN  0        128      [::]:22              [::]:*              users:(("sshd",pid=1122,fd=4))
```

La configuración del servicio se realiza a través de un fichero de configuración que está situado en la ruta `/etc/ssh/sshd\_config`. Las distribuciones de GNU/Linux ya traen una configuración predeterminada que suele constar de las siguientes directivas (aunque hay muchas más):

- **Port**: Normalmente viene comentada, ya que el puerto por defecto es el 22. En caso de querer cambiar el puerto, podremos modificar esta línea, asegurando que no esté comentada.

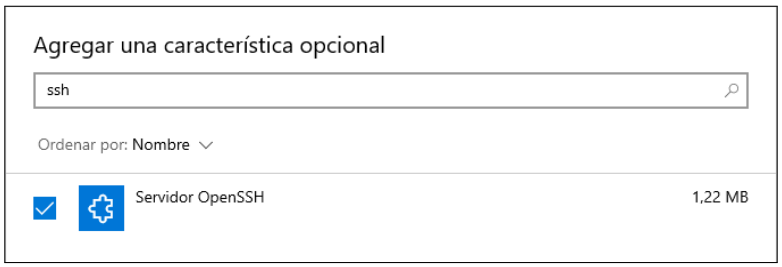
- **ListenAddress**: Por defecto SSH se pondrá a la escucha en todos los interfaces que tengamos configurados. Si sólo nos interesa escuchar en alguna de las IPs que tengamos configuradas, deberemos modificar esta configuración.
- **PermitRootLogin**: Para evitar problemas de seguridad, esta directiva suele estar configurada a "**No**", para evitar que se puedan usar los credenciales de root para hacer el login.
- **PubkeyAuthentication**: Esta directiva permite realizar la conexión a través de unas claves públicas/privadas que podemos crear. Se explicará más adelante.

Hoy día también se puede instalar en Windows 10 y posteriores a través de un comando, siendo administrador de PowerShell:

Instalando OpenSSH Server en Windows 10

```
PS C:\Windows\System32> Add-WindowsCapability -Online -Name OpenSSH.Server~~~~0.0.1.0

PS C:\Windows\System32> Start-Service sshd

PS C:\Windows\System32> Set-Service -Name sshd -StartupType 'Automatic'
```

O desde el interfaz gráfico a través de las "**Características opcionales**", buscando por ssh:



2.3.2. Cliente SSH

El cliente SSH es aquel programa que a través del protocolo SSH se puede conectar a un servidor SSH. Existen distintos tipos de clientes que podemos utilizar:

- **CLI**: El cliente de consola es el más habitual. Está instalado de forma habitual en todas las distribuciones de GNU/Linux (normalmente el paquete se llama **openssh-client**). También lo encontramos instalado por defecto en MacOS.

Hoy día también está instalado en Windows 10, y de no estar, se puede instalar a través de las "**Características Opcionales**".

- **GUI**: Existen distintos interfaces gráficos que nos puede interesar utilizar:
  - **Putty**: Un cliente muy habitual en entornos Windows.
  - **Kitty**: Una versión mejorada del anterior.
  - **Termius**: Cuenta con versión de escritorio y móvil.

Para realizar la conexión al servidor SSH debemos conocer:

- **Dirección del servidor**: Ya sea mediante IP o nombre FQDN (*fully qualified domain name*) que se resuelva.
- **Puerto**: Ya hemos comentado que por defecto el puerto es 22.
- **Usuario**: Para realizar el sistema de autenticación, necesitamos un usuario que exista en el sistema.
- **Contraseña**: Los credenciales de acceso del usuario.

Para realizar la conexión desde un cliente de consola ejecutaremos:

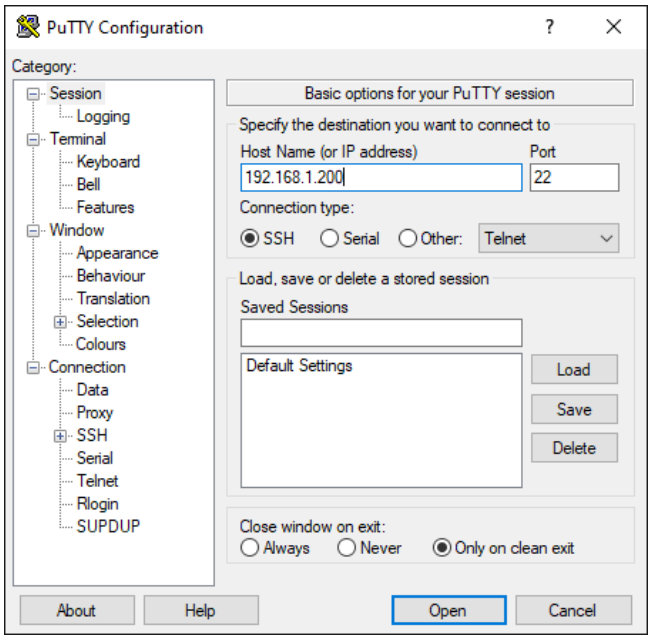
Conexión SSH

```
ruben@vega:~$  ssh usuario@192.168.1.200 -p 22
```

En el comando anterior podemos identificar:

- **ssh**: el cliente de consola
- **usuario**: el nombre del usuario con el que nos queremos conectar al servidor remoto.
- **@**: la arroba en inglés significa “at”, que indica “usuario en el servidor X”.
- **192.168.1.200**: La IP del servidor al que nos queremos conectar.
- **-p 22**: Estos dos parámetros van juntos, “-p” indica que vamos a indicar el puerto de conexión y “22” que nos queremos conectar a ese puerto. Debido a que 22 es el puerto por defecto, podríamos no poner estas opciones si sabemos que el servidor escucha en el puerto 22.

Si realizamos la conexión a través de un cliente de interfaz, como es putty, el aspecto será el siguiente, donde sólo podremos introducir la IP del servidor. Cuando se comience con la conexión nos pedirá los credenciales de acceso.



Si es la primera vez que nos conectamos a un servidor mediante SSH nos saldrá un mensaje como el siguiente:

Conexión SSH

```
ruben@vega:~$  ssh usuario@192.168.1.200 -p 22

The authenticity of host '192.168.1.200 (192.168.1.200)' can't be established.
ECDSA key fingerprint is SHA256:uK9MOl0gLDhTtCrlcafc1zVObVA/vnOMn6TWFsQb23o.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

Este "key fingerprint" es un identificador que está relacionado con el fichero de "**clave pública**" del servidor. Es como el DNI del servidor. La primera vez se nos guarda ese *fingerprint*, y en caso de que en una próxima conexión varíe, nos avisará. No suele ser habitual que este identificador cambie.

2.3.3. Conexión mediante certificados de clave pública/clave privada

Existe una alternativa a la hora de realizar una conexión SSH para que no nos pida la contraseña del usuario, y es **hacer uso de los certificados de clave pública y clave privada**. Este concepto de "clave pública y clave privada" viene de la **criptografía asimétrica**.

Este sistema de criptografía asimétrica hace uso de dos claves que están asociadas entre sí:

- **Clave privada**: Es la base del sistema criptográfico, y como su nombre indica, se debe de mantener en privado. **Nunca se debe de compartir**, ya que entonces se podrían hacer pasar por nosotros.
- **Clave pública**: Asociada a la clave privada, la clave pública puede ser compartida y enviada a otros ordenadores para poder realizar la conexión.

Para generar el par de claves se realiza con el siguiente comando:

Crear par de claves pública/privada

```
ruben@vega:~$  ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ruben/.ssh/id_rsa):

Enter passphrase (empty for no passphrase):
Enter same passphrase again:

Your identification has been saved in /home/ruben/.ssh/id_rsa
Your public key has been saved in /home/ruben/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:SPqPOYBmPb8PCFhcZgqcWZPZzaL5RNfMeKmHqebvC7E ruben@vega
The key's randomart image is:
+---[RSA 3072]----+
|o +oB o = .      |
| * B.+ = *       |
```

```
|  + + + =        |
| o o + = .       |
|. .o+.o S        |
|  +.+*o          |
| o  +Eo          |
|     .+=         |
|      *B+        |
+----[SHA256]-----+
```

El comando muestra los siguientes pasos:

1. Creación de la pareja de claves pública/privada haciendo uso del sistema criptográfico **RSA**.
2. Lugar donde se va a guardar la clave privada. Por defecto en `\~/.ssh/id\_rsa`.
3. Contraseña para securizar la clave privada. De esta manera, para poder usarla habrá que introducir dicha contraseña. Dado que nosotros queremos evitar introducir contraseñas, lo dejaremos en blanco.
4. Lugar donde se va a guardar la clave pública. Por defecto en `\~/.ssh/id\_rsa.pub`

Una vez tenemos nuestro par de claves, podemos copiar la clave pública al usuario del servidor que nos interese mediante el siguiente comando:

Crear par de claves pública/privada

```
ruben@vega:~$  ssh-copy-id user@servidor_remoto
```

Para ello es imprescindible conocer previamente la contraseña del usuario en el servidor. El comando `ssh-copy-id` realizará una conexión SSH y copiará el contenido de la **clave pública**, `\~/.ssh/id\_rsa.pub`, dentro del fichero `\~/.ssh/authorized\_keys` del usuario en el servidor remoto. Este paso se puede realizar a mano (con un editor de texto).

¡Cuidado!

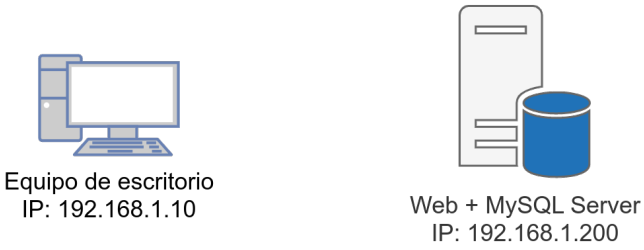
Windows no tiene el comando "ssh-copy-id", por lo que deberemos hacer el paso a mano, tal como se ha explicado.

Al realizar la siguiente conexión, ya no necesitaremos introducir la contraseña del usuario, ya que el sistema remoto podrá autenticarnos a través del sistema clave pública/privada.

2.3.4. Crear túneles SSH

Una de las funcionalidades extra de SSH es la posibilidad de crear "túneles" para securizar protocolos no seguros, o poder acceder a servicios que sólo escuchan en *localhost*.

Pongamos como ejemplo el siguiente escenario:



Tenemos un servidor web con Apache y MySQL al que queremos acceder. Por seguridad MySQL **sólo está escuchando en localhost (127.0.0.1)**, por lo que el acceso al servicio MySQL no es posible. Para administrarlo nos tenemos que conectar al servidor, y realizarlo de manera local.

En este punto es donde entra en juego la creación de un túnel seguro al servidor, para poder acceder al servicio remoto. **Para ello es imprescindible poder realizar una conexión SSH** (ya sea mediante usuario o clave pública/privada).

Para crear un túnel, desde el equipo de escritorio, lanzaremos el siguiente comando:

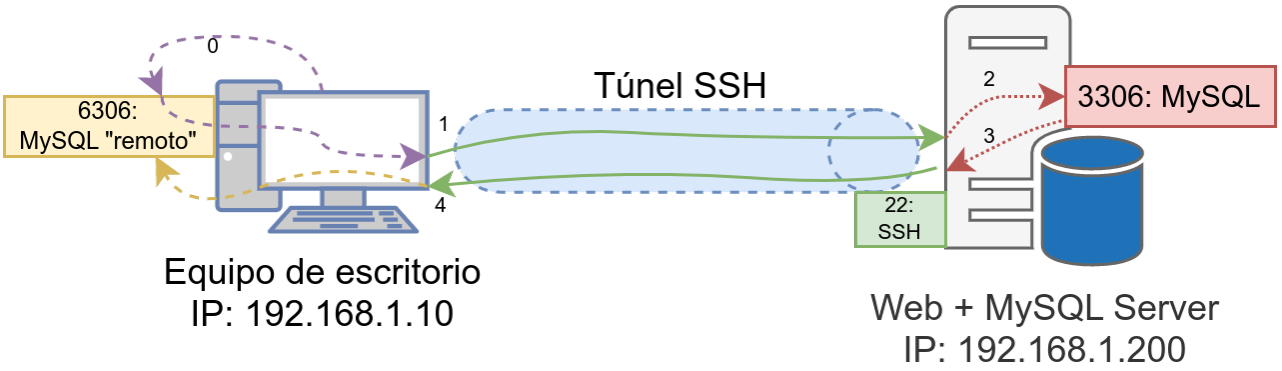
Crear par de claves pública/privada

```
ruben@vega:~$  ssh usuario@192.168.1.200 -L 6306:127.0.0.1:3306 -N
```

Al ejecutar este comando, habremos creado un túnel que enlaza el puerto remoto 3306 (que sólo se escucha en el "127.0.0.1" del servidor), con el puerto local 6306 del equipo de escritorio. A continuación la explicación del comando y sus parámetros:

- "**ssh usuario@192.168.1.200**": es como una conexión SSH normal. Lo que estamos indicando es que queremos conectarnos con "usuario" al servidor remoto 192.168.1.200 a través de SSH.
- **-L 6306:127.0.0.1:3306**: Especifica que el puerto local especificado se va redirigir al puerto e IP remota. Para entender esto hay que separar dos partes de los parámetros:
  - **6306**: Especifica la IP y el puerto local. En este caso, antes del puerto no hemos especificado IP, por lo que se creará un puerto 6306 que sólo se pone a la escucha en ***localhost*** en el equipo de escritorio
  - **127.0.0.1:3306**: Esta es la dirección y puerto remoto al que nos queremos conectar. En este caso, es el puerto 3306 que está escuchando en la IP 127.0.0.1 del servidor.
- **-N**: Sirve para que no ejecute ningún comando en el servidor remoto, y por tanto no nos abrirá conexión de terminal.

A nivel visual, y para entender de mejor manera lo realizado, sirva la siguiente imagen y los pasos que se pueden dar en un escenario real:



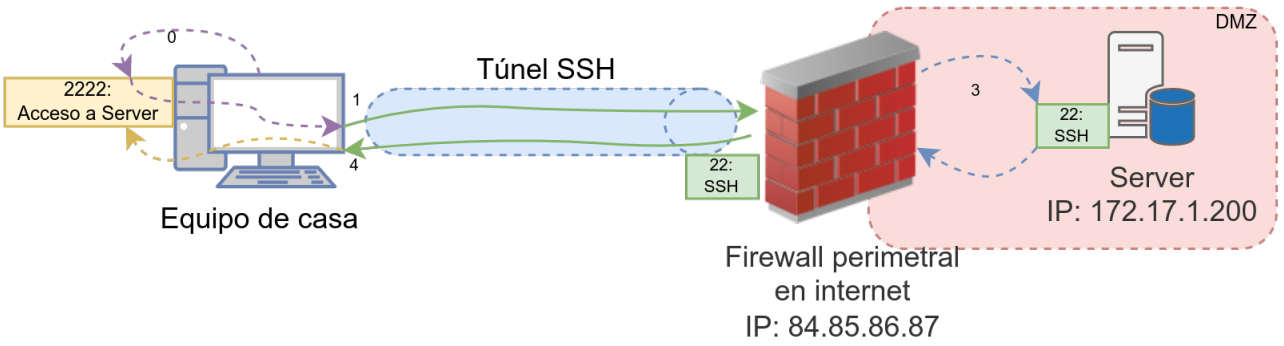
Con una aplicación en el equipo de escritorio queremos conectarnos al servidor MySQL que sólo escucha en el servidor remoto. Ejecutamos el túnel visto anteriormente, y los pasos que podremos realizar son los siguientes:

0. Mediante una aplicación nos podemos conectar al puerto local 6306, que ha sido creado mediante el comando anterior. Este puerto local está redirigido al puerto del servidor remoto. Por lo tanto la conexión se securiza a través del túnel
1. Como el túnel está establecido, la conexión viaja de manera segura a través de él.
2. Al llegar al servidor remoto, sabe que la conexión debe ir al puerto 3306 de la IP 127.0.0.1, que es lo establecido en el comando.
3. La conexión vuelve al túnel.
4. Viaja por el túnel hasta llegar a la comunicación que se había establecido previamente.

De esta manera, hemos podido realizar una conexión a un servicio remoto a través de SSH y completamente seguro.

2.3.4.1. Acceder a un equipo saltando a través de otro.

Este es un caso especial de túnel, similar a lo explicado previamente. En lugar de querer realizar una conexión a un servicio del equipo al que nos conectamos, en este caso lo usaremos de salto para acceder a otro servidor.



En este caso estando en casa nos queremos conectar a un equipo de la oficina. No tenemos VPN, y no hay redirección de puertos directa, pero podemos acceder al firewall perimetral. Por lo tanto, lo podemos utilizar para saltar al servidor que nos interesa.

En este caso, el comando a ejecutar sería:

Crear par de claves pública/privada

```
ruben@vega:~$  ssh usuario_firewall@84.85.86.87 -L 2222:172.17.1.200:22 -N
```

De esta manera, ahora desde nuestro equipo podremos realizar una conexión SSH al puerto 2222 que realmente será una redirección que viaja a través del túnel hasta el firewall, y que a su vez redirige al puerto 22 del servidor 172.17.1.200.

¡Atención!

El servicio remoto al que nos queremos conectar debe escuchar en la IP del equipo al que nos queremos conectar. El equipo que usamos para saltar debe poder conectarse a él.