Ingeniería de Servidores (2025-2026)
Grado en Ingeniería Informática
Universidad de Granada

Práctica 1 : Virtualización e instalación de Sistemas Operativos

---

Índice

Índice de figuras

Índice de tablas

OBJETIVOS MÍNIMOS

1. Familiarizarse con distintos Sistemas Operativos (SOs) usados en servidores.
2. Conocer alternativas comerciales para tener un servidor.
3. Configurar unidades de disco RAID y LVM.
4. Configurar una red local de máquinas virtuales.

Lecciones

1. Instalación y configuración de Debian
2. Configuración de LVM con Alma Linux
3. Configuración de RAID1 con Alma Linux

Competencias que se trabajarán

Competencias Básicas

1. CB2. Que los estudiantes sepan aplicar sus conocimientos a su trabajo o vocación de una forma profesional y posean las competencias que suelen demostrarse por medio de la elaboración y defensa de argumentos y la resolución de problemas dentro de su área de estudio.

Específicas de la Asignatura

1. R1. Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente.
2. R2. Capacidad para planificar, concebir, desplegar y dirigir proyectos, servicios y sistemas informáticos en todos los ámbitos, liderando su puesta en marcha y su mejora continua y valorando su impacto económico y social.
3. R5. Conocimiento, administración y mantenimiento de sistemas, servicios y aplicaciones informáticas.

Competencias Específicas del Título:

1. E4. Capacidad para definir, evaluar y seleccionar plataformas hardware y software para el desarrollo y la ejecución de sistemas, servicios y aplicaciones informáticas.

Competencias Transversales o Generales:

1. T2. Capacidad de organización y planificación así como capacidad de gestión de la Información.

1. Introducción

En esta práctica el alumno podrá realizar todos los pasos para la instalación de un servidor real. Para ello, una vez que se posee la máquina, es necesario instalar el Sistema Operativo (SO) que proporcionará y sobre el que se ejecutarán los servicios. Dado que no es posible tener la infraestructura necesaria para poder trabajar con servidores físicos, recurriremos a la virtualización de los servidores. Este método cada vez es más popular y se presenta como una gran alternativa a tener desplegado un servidor físico, también denominado como infraestructura *on-premise*.

1.1. CONCEPTO DE MÁQUINA VIRTUAL Y VIRTUALIZACIÓN

Un software de máquinas virtuales es, esencialmente, aquel que permite crear una capa de abstracción sobre el HW en el que se ejecuta de modo que pueden ejecutarse simultáneamente varias máquinas virtuales en el mismo servidor (o conjunto de servidores) físico. Hay una tecnología similar que está ampliamente adoptada, especialmente en entornos cloud que es denominada contenedores. Ésta permite compartir recursos entre los contenedores y el anfitrión. Una posible analogía es la diferencia entre proceso y hebra, dos máquinas virtualizadas completamente serían como dos procesos mientras que los contenedores serían como las hebras (que comparten "cosas") [10, 8, 3].

1.1.1. Método de trabajo

El estudiante podrá realizar el trabajo en los equipos de las aulas de la ETSIIT utilizando una memoria USB (o una memoria USB por máquina) para poder llevarse a casa las máquinas virtuales creadas y seguir trabajando con ellas.
También existe la posibilidad de que el alumno traiga su propio portátil y trabaje con él si bien es recomendable que éste tenga al menos 4 GiB de memoria RAM y la capacidad de ejecutar como mínimo dos hebras simultáneamente.
Por último, si el alumno desea trabajar con un puesto fijo ubicado en otro lugar, debe ser capaz de acceder a éste desde el aula de prácticas (mediante compartición de escritorio, teamviewer, tigherVNC, logmein, etc.).
Lo importante es que el alumno siempre tenga disponibles sus máquinas virtuales durante las sesiones de prácticas.
Adicionalmente, el progreso será ilustrado con capturas de pantalla e historiales de comandos que deben subirse a un repositorio de GitHub o GitLab (público o privado) y compartirse con el profesor tal y como se indica en la presentación de la P1-L0.

1.1.2. Software para virtualización: VMWare y VirtualBox

En el laboratorio se podría utilizar tanto VMWare (Player) como Virtual Box, pero para homogeneizar preguntas y problemas, nos ceñiremos al uso de VirtualBox. A continuación, para evitar confusiones y por economía del lenguaje, el software de virtualización se notará como VMSW (*Virtual Machine SoftWare*).

Usted debe conocer qué tipo o modo de virtualización utiliza Virtual Box así como tener una noción del concepto de "contendor". También debe ser consciente de las alternativas que tenemos a día de hoy en lo que se refiere al alojamiento de los servidores pasando por los tecnicismos hosting dedicado y virtual, housing y conocer sus ventajas e inconvenientes de cara a tomar decisiones para definir una infraestructura.

2. INSTALACIÓN DE SISTEMAS OPERATIVOS VIRTUALIZADOS

En esta práctica el alumno debe crear dos máquinas virtuales e instalar dos de los SOs Linux usados comúnmente en servidores: Debian y Alma Linux (que da continuidad al, todavía popular, proyecto de CentOS `https://rockylinux.org/` que fue extinguido por Red Hat).

Debian: Para su instalación, cree una máquina virtual e indique que instalará el SO más adelante. Una vez creada, seleccione el dispositivo CD, cargue la imagen correspondiente y arranque la máquina :

`http://atcproyectos.ugr.es/esriie/debian-13.0.0-amd64-netinst.iso`

Alma: En este caso, puede indicarle a VMSW que el SO es Fedora/CentOS/RHELinux cuando le pregunte qué SO instalará. En caso de no aparecer esa opción, indicaremos que es Fedora 64 bits. Al igual que con Debian, necesitará indicarle dónde está el archivo con la imagen:

`http://atcproyectos.ugr.es/esriie/AlmaLinux-10.0-x86_64-minimal.iso`

Para tomar decisiones antes de configurar un servidor, es importante conocer quién está detrás de cada distribución, su relación con otras distribuciones así como qué empresas dan soporte y apoyo ante posibles problemas e indagar cuánto podría llegar a costar un poco de ayuda.

Es obligado citar que Windows Server en sus distintas versiones es una alternativa con un porcentaje de uso no despreciable (20 % en 2022) [11] pero, desafortunadamente, no se utilizará en las prácticas. Los motivos son: 1) la falta de tiempo 2) la opacidad en comparación a un sistema de código abierto y 3) los requisitos a nivel de infraestructura para poder trabajar con él.

2.1. Configurando el disco duro virtual y RAID1

Cuando estamos preparando una máquina para su uso como servidor, se nos plantean tomas de decisiones cruciales tanto a nivel hardware como a nivel software desde el inicio del diseño de la solución. La instalación de un SO en una máquina implica a una serie de elementos que deben ser configurados desde el comienzo, cuya modificación implica la detención del servicio y un esfuerzo adicional, además de incrementar la posibilidad de cometer errores.

Uno de los elementos más importantes es el almacenamiento ya que el número de parámetros que se pueden configurar es elevadísimo y su impacto en el rendimiento, fiabilidad,

tolerancia a fallos, etc. es enorme.
Por tanto, usted debe ser capaz de tomar una decisión de a tener que elegir un sistema de archivos concreto así como tener nociones sobre cómo gestionar el almacenamiento.
Dada la importancia del almacenamiento, en servidores es normal aplicar soluciones RAID para que el acceso sea más eficiente y para preservar la información en caso de que haya roturas o problemas de disco.
Tal y como se mostrará en las lecciones, se espera que usted configure un RAID1 utilizando LVM y cifrando la información que los volúmenes contienen. Este proceso puede hacerse fácilmente durante la instalación de Debian. Para Alma deberá hacerlo una vez instalado el sistema por defecto, de ese modo, puede hacerse una idea del coste de no tomar ciertas decisiones en el momento correcto.

2.2. Configuración de la red

Otro de los aspectos importantes en la administración de puestos de trabajo y configuración de servidores es la configuración de la red [6]. Por tanto se espera que usted sepa configurar las interfaces de red a nivel de aplicación de VMSW así como para los dos SOs con sus particularidades.
La configuración que se utilizará será una que permita comunicarse a las máquinas virtuales entre sí y con el anfitrión así como tener conexión al exterior.
Siguiendo las consideraciones de la documentación de VirtualBox (cap. 6) así como la documentación de Debian, CentOS (y REHL) y Alma, así como las páginas del manual, establecemos el siguiente procedimiento para llegar a la configuración de red:

1. Nos aseguramos de que la red solo anfitrión que crea Virtual Box tiene los parámetros correctos (asignados manual o automáticamente) : 192.168.56.1

2. Hemos de añadir para cada MV una interfaz de red que conectaremos en modo "solo-afintrión"(Host-only). En caso de no tener una red local con el anfitrión, iremos al menú Preferencias de VBOX y crearemos una nueva red local (p.ej. vboxnet0) con el anfitrión (Ctrl+W, Ctrl+H en las nuevas versiones (usar cmd en vez de Ctrl en Mac) para acceder al menú de creación).

3. Comprobaremos que nuestra nueva interfaz está ahí mediante la ejecución de `lspci | grep Ether`

   A modo de registro y por aprender cómo ha ido evolucionando, mostramos la configuración con Ubuntu Server 16.04: editar el archivo `/etc/network/interfaces` y añadimos:

   ```
   auto enp0s8
   iface enp0s8 inet static
   address 192.168.56.105
   ```

   Desde la versión 12 de Debian hay un nuevo servicio denominado `netplan`[1] que nos permite establecer la configuración mediante archivos YAML [2] (lo veremos en las

[1] `https://wiki.debian.org/Netplan`

siguientes prácticas). Esta nueva herramienta permite analizar un archivo con una configuración clara y legible y generar sobre la marcha ("renderinzar") el archivo de configuración para la interfaz para el módulo correspondiente (normalmente NetworkManager o systemd).

Para ello, editaremos el archivo de configuración disponible en `/etc/netplan/` con nombre `00-installer-config.yaml` y escribiremos:

```
network:
    ethernets:
        enp0s3:
            dhcp4: true
        enp0s8:
            dhcp4: false
            addresses:
                [192.168.56.105/24]
    version: 2
```

Una vez editado, aplicaremos los cambios ejecutando `sudo netplan apply` .

Para más detalles sobre esta herramienta y su configuración visite [1].

Igualmente, indicamos cómo se haría en CentOS 7: Añadimos un nuevo archivo (script) a `/etc/sysconfig/network-scripts` con el nombre `ifcfg-enp0s8` con el siguiente contenido:

```
TYPE=Ethernet
BOOTPROTO=none
NAME=enp0s8
DEVICE=enp0s8
ONBOOT=yes
IPADDR=192.168.56.110
NETMASK=255.255.255.0
```

Realizamos la prueba de que todo está correcto ejecutando `ip addr` y haciendo `ping` a cada máquina y desde cada máquina al anfitrión. [2]

En la última versión de CentOS 8 disponemos de las herramientas `nmtui` y `nmcli` [9] que nos permitirá editar la configuración fácilmente. Alma Linux las tiene disponibles también.

Por tanto, para Alma, la distribución que utilizaremos, la configuración se las interfaces de red se realiza del mismo modo que con CentOS 8, es decir, mediante el NetworkManager.

[2] Si su anfitrión es Windows, asegurese de configurar el cortafuegos para que los mensajes de ICMP no sean bloqueados

Existe la posibilidad de hacer la adición de la interfaz de red antes de hacer la instalación del SO y, en ambos sistemas, se configurará automáticamente.

2.3. Creación del usuario para y otorgando permisos

En las instalaciones por defecto con algunas distribuciones, el instalador nos pregunta si deseamos crear un usuario y otorgarle privilegios así como si deseamos establecer la contraseña del usuario root.
Durante la instalación de Debian nos pregunta sobre la creación de un usuario y la contraseña de root. Tal y como explican en el manual de instalación [4], si especificamos contraseña para root, podremos autenticarnos como tal pero el usuario que creemos después no tendrá privilegios. Si, por el contrario, dejamos la contraseña en blanco, el usuario root no podrá autenticarse pero el usuario que creemos después sí podrá hacer administración mediante sudo.
Para poder darle privilegios a ese usuario o crear un usuario nuevo con éstos una vez tenemos el sistema instalado, podemos consultar la documentación de Rocky Linux [7] y de REHL [5].
Para añadir un usuario usaremos el comando `useradd nombre_usuario` que creará automáticamente un grupo con el mismo nombre del usuario, podríamos haber creado previamente el grupo. Le asignaremos la contraseña con `passwd nombre_usuario`.
De cara a proporcionar a este usuario con privilegios para realizar tareas administrativas tenemos la opción de incluirlo en el grupo *wheel* (con el comando:
`usermod -a -G wheel nombre_usuario`) pero, de cara a tener más control sobre nuestras acciones y que cada comando sudo se registre en /var/log/messages y /var/log/secure, lo añadiremos al conjunto de usuarios que pueden usar el comando sudo.
Para ello, debemos editar el archivo `/etc/sudoers` con el comando `visudo` y añadir la línea:

```
nombre_usuario ALL=(ALL)     ALL
```

que otorgará a `nombre_usuario` privilegio para ejecutar mediante la utilidad sudo cualquier comando desde cualquier máquina.
En Debian, puede verse contrariado al no encontrar ciertos comandos, considere añadir al PATH el directorio /usr/sbin: `export PATH=\$PATH:/usr/sbin`

Anexo I: Contextualización de Escenarios

2.1. Lección 1

Usted está trabajando en una empresa proveedora de servicios y recibe la solicitud de un cliente que sea tener un servidor para la implantación de un comercio electrónico mediante un CMS.
Sin tener más detalles por parte del cliente, le pregunta a un compañero qué configuración se suele aplicar en estos casos. Este le remite a su jefa de Dpto. que le recomienda la configuración de un RAID1 gestionado con LVM, cifrando toda la información para

cumplir con la legislación vigente. También le recomienda crear al menos 3 VL (hogar, raiz y swap) y una partición para el arranque.
Nota: El usuario que crearemos debe ser su primer nombre de pila acompañado de las iniciales de sus apellidos en mayúscula. Por ejemplo, Jose Manuel Fernan Pese ->JoseFP

2.2. Lección 2

En esta ocasión, en la empresa en la que le acaban de contratar tenían adquirido un servidor y su predecesor había realizado la instalación del S.O. Alma Linux, según le han comentado los compañeros, él solía hacer instalaciones por defecto y luego aplicar scripts de configuración. Sin más información, nuestro jefe nos informa que esa máquina va a alojar unos cursos con vídeos de alta calidad y relativamente largos. Por tanto, viendo la configuración del sistemas, prevemos que /var necesitará más espacio, incluso es conveniente asignarle un LV exclusivamente. Para ello, incluiremos un nuevo disco y configuraremos LVM para que /var se monte en el nuevo VL que crearemos para él.

2.3. Lección 3

Tras ver el éxito de los vídeos alojados en el servidor configurado en la práctica anterior, un amigo de su cliente quiere proceder del mismo modo pero va a necesitar alojar información sensible así que le pide explícitamente que cifre la información y que ésta esté siempre disponible. Por tanto, la decisión que toma es configurar un RAID1 por software y cifrar el VL en el que /var estará alojado.

Referencias

[1] Canonical. Netplan. https://netplan.io/?_ga=2.187511151.617163006.1600981856-1784695980.1593188495, 2020. [Online; consultada 24-Septiembre-2020].

[2] Canonical. Network configuration. https://ubuntu.com/server/docs/network-configuration, 2020. [Online; consultada 24-Septiembre-2020].

[3] Docker. What is a container. https://www.docker.com/what-container, 2017. [Online; consultada 14-Septiembre-2017].

[4] El equipo del instalador de Debian. Guía debian gnu/linux de instalación. https://www.debian.org/releases/stable/amd64/install.es.pdf, 2025. [Online; consultada 7-Septiembre-2025].

[5] Red Hat. Documentación rhel 7: Chapter 6. gaining privileges. https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system_administrators_guide/chap-gaining_privileges, 2022. [Online; consultada 24-Septiembre-2022].

[6] Thomas A. Limoncelli, Christina J. Hogan, and Strata R. Chalup. *Practice of System and Network Administration, The (2Nd Edition)*. Addison-Wesley Professional, 2007.

[7] Rocky Linux. User management. , 2022. [Online; consultada 24-Septiembre-2022].

[8] Oracle. Tipos de virtualización. https://www.virtualbox.org/wiki/Virtualization, 2017. [Online; consultada 14-Septiembre-2017].

[9] Inc. Red Hat. Red hat enterprise linux documentation. chapter 2 getting started with networkmanager. https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/configuring_and_managing_networking/getting-started-with-networkmanager_configuring-and-managing-networking, 2020. [Online; consultada 12-Septiembre-2022].

[10] VMWare. Tipos de virtualización. https://www.vmware.com/es/solutions/virtualization.html, 2017. [Online; consultada 13-Septiembre-2022].

[11] W3Techs. Usage of operating systems for websites. https://w3techs.com/technologies/overview/operating_system/all, 2017. [Online; consultada 12-Septiembre-2022].

Ingeniería de Servidores (2025-2026)
Grado en Ingeniería Informática
Universidad de Granada

Práctica 2 : Instalación y configuración de servicios

Índice

Índice de figuras

Índice de tablas

OBJETIVOS MÍNIMOS

1. Saber instalar nuevas aplicaciones y conocer los distintos sistemas de gestión de paquetes en Linux.
2. Poder configurar de manera sencilla el cortafuegos.
3. Poder acceder a un servidor de manera segura con SSH.
4. Conocer el concepto de pila *stack* software e instalar una (LAMP).
5. Conocer una interfaz web para administrar servicios.
6. Conocer herramientas para hacer copias de seguridad (backups) y saber usar control de versiones (git).

Lecciones

1. SSH
2. Instalación Servidor Web
3. Copias de seguridad y control de versiones

Competencias que se trabajarán

Competencias Básicas

1. CB2. Que los estudiantes sepan aplicar sus conocimientos a su trabajo o vocación de una forma profesional y posean las competencias que suelen demostrarse por medio de la elaboración y defensa de argumentos y la resolución de problemas dentro de su área de estudio.

Específicas de la Asignatura

1. R1. Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente.
2. R2. Capacidad para planificar, concebir, desplegar y dirigir proyectos, servicios y sistemas informáticos en todos los ámbitos, liderando su puesta en marcha y su mejora continua y valorando su impacto económico y social.
3. R5. Conocimiento, administración y mantenimiento de sistemas, servicios y aplicaciones informáticas.

Competencias Específicas del Título:

1. E4. Capacidad para definir, evaluar y seleccionar plataformas hardware y software para el desarrollo y la ejecución de sistemas, servicios y aplicaciones informáticas.

Competencias Transversales o Generales:

1. T2. Capacidad de organización y planificación así como capacidad de gestión de la Información.

1. Introducción

Una vez que el sistema operativo está instalado, podemos pasar a configurar la máquina para que dé servicios. Por ejemplo, se podría programar un servicio personalizado y escuchar por el puerto correspondiente pero, ¿está accesible? ¿Cómo se puede averiguar? Son preguntas básicas que se podrán contestar tras la realización de la práctica.
No obstante, antes de pensar en programar un servicio, en el mundo Linux hay muchas implementaciones que nos pueden servir (por ejemplo, un servidor web, FTP, etc.). Por tanto lo primero que haremos será aprender el uso de los gestores de paquetes más importantes.
Tras haber instalado el software necesario, pasaremos a configurar y aprender a usar el servicio de *Secure SHell* (SSH) que nos permitirá la administración remota de nuestra máquina, así como algunas utilidades prácticas para usarla adecuadamente y unos mecanismos de seguridad básicos para evitar ciertos problemas.
Por último, aprenderemos a instalar un servidor web que nos permita alojar sitios dinámicos.

2. Instalando software

Para poder instalar nuevos servicios, existen gestores de paquetes que permiten realizar esta tarea de una manera muy sencilla. Aunque estas aplicaciones tienen una Graphical User Interface (GUI), es recomendable saber utilizarlas en la consola. En el caso de Windows, muchos servicios proporcionados por Microsoft pueden gestionarse mediante una ventana de configuración. En esta sección se aprenderá cómo utilizar estas herramientas desde la línea de comandos (aunque también existen GUIs que permiten la gestión de paquetes) que nos permiten instalar los servicios de manera fiable y segura.

2.1. dnf y apt

Tanto dnf [5] como apt [3] son dos herramientas para gestionar paquetes que permiten conectarse a repositorios (servidores que suministran paquetes) e instalar software resolviendo las dependencias con otros paquetes. Al igual que es sencillo realizar la instalación, también nos permiten más opciones como buscar y desinstalar paquetes. Además, también podremos añadir nuevos repositorios siempre que queramos.
Los paquetes pueden tener un sistema de firma que nos garantiza la fiabilidad de que no estamos cogiendo el paquete equivocado o adulterado.

Por tanto, se considera necesario que usted sepa realizar las operaciones descritas anteriormente con los gestores de paquetes de las distribuciones que estamos usando.
Con el objetivo de converger a un gestor de paquetes que englobe las dependencias facilitando el mantenimiento de los paquetes hay dos propuestas que usted debe conocer: Flatpak [1] y Snap [8]. Se recomienda visitar el sitio de ambos enfoques para profundizar el conocimiento al respecto.
Aunque su uso no es necesario ponerlo en práctica, sí debe ser consciente de que algunas distribuciones como Fedora ya permiten usar una de estas alternativas.

3. Gestionando el cortafuegos

Tanto Debian como Alma Linux disponen de un *front-end* para el cortafuegos que permite definir cómodamente las reglas para nftables (el reemplazo del obsoleto iptables [7]).
Independientemente de que estudie o no el uso de nftables (o iptables) en otra asignatura, es muy útil conocer el uso adecuado del Uncomplicated FireWall (ufw) en Debian así como el firewall-cmd en Alma Linux. Se espera que usted sea capaz de abrir y cerrar puertos así como de comprobar su estado. Para ello, no solo usaremos las opciones de estos comandos sino que haremos un escaneo de puertos con el comando nmap.

4. SSH

Una vez que ya se disponen de las herramientas para instalar servicios y abrir la puerta para que presten servicio, vamos a trabajar con la administración remota.
Es importante ser conscientes de la ambigüedad de que ssh es tanto un cliente como un servicio [2]. En algunos sistemas es sencillo ya que para denotar al servicio, se utiliza una d (de daemon) al final. Debe prestar especial atención cuando edite los archivos de configuración.
Tras ver la lección correspondiente, se espera que usted sea capaz de instalar, configurar y "asegurar" el servicio SSH, limitando el acceso por contraseña, cambiando el puerto por defecto y prohibiendo que el usuario root tenga acceso al sistema.

4.1. screen, terminator y tmux

De cara a abrir una sesión con una shell y poder cerrar la conexión sin que la shell termine, podemos usar los comandos screen y tmux. Gracias a estos comandos, podemos tener varias sesiones abiertas y reconectarnos a ellas aunque la conexión se haya perdido.
Por otra parte, terminator es una consola que en modo gráfico (tmux para modo texto) permite tener varias terminarles abiertas en una única ventana así como hacer operaciones de *broadcast*, es decir, teclear en una ventana y que el texto aparezca en otras.
Usted deberá saber hacer uso de screen usando la opción más común como es la de recuperar una "ventana" y "desmarcarla" de otro sitio.

4.2. Un poco de seguridad: fail2ban y rkhunter

Para evitar ataques de fuerza bruta podemos usar fail2ban [4] que mete en una lista negra (encarcela) las IPs que han intentado iniciar sesión de manera errónea varias veces seguidas en un intervalo de tiempo predefinido.
RootKit Hunter (`http://rkhunter.sourceforge.net/`) es un software que permite analizar el sistema para ver si hay software malicioso instalado en la máquina (*rootkits, backdoors, exploits*, etc.).
Se espera que usted sea capaz de instalar, configurar y gestionar la funcionalidad básica de fail2ban así como de ejecutar rkhunter y determinar si hay alguna amenaza en el sistema.

5. Instalando un servidor web básico

Uno de los servicios más comunes que se suelen instalar en un servidor es un programa para servir páginas web. Incluso hay entornos de desarrollo que se basan en éste (p.ej. jupyter).
Por tanto, se espere que usted sepa configurar un servidor web con la funcionalidad básica que consiste en tener un servidor web (Apache) un gestor de bases de datos (MariaDB) y los interpretes de lenguajes de uso común para la web como pueden ser PHP y Python, todo ésto conocido comúnmente por LAMPP.
Debe demostrar su funcionamiento mediante un script (en uno de los dos lenguajes citados) que involucre a todos los actores enunciados.

5.1. Webmin y cockpit

Como utilidad inmediata para la administración, mencionamos un ejemplo de una interfaz web que nos permite administrar el sistema: Webmin (`http://www.webmin.com/`).
Existen otras alternativas de paneles de administración: C-Panel, Parallels Plesk, DirectAdmin, etc. (C-Panel y Parallels Plesk son los más populares) pero suelen ser de código cerrado y comerciales.
Hay una alternativa de popularidad creciente desarrollada por Red Hat: cockpit [6]. Para su instalación puede consultar la información disponible en `https://cockpit-project.org/running.html`.
Usted debe saber instalar y dejar en funcionamiento Webmin y cockpit además de razonar ventajas y desventajas de usar este tipo de interfaces frente a SSH.

6. Copias de seguridad y control de versiones

El mantener la información del sistema así como la de los usuarios es algo crucial cuando estamos trabajando con servidores.
Los comandos esenciales para esta tarea son: tar y rsync (aunque también existen otros bastante usados como dd y cpio) [9].

Existen también otras soluciones más complejas (y completas) para gestionar las copias de seguridad como Amanda (`http://www.amanda.org/`) y Bacula (`https://blog.bacula.org/`).
Además de la copia de los datos, siempre que modifiquemos un archivo de configuración deberemos hacer un "backup" de éste por si introducimos algún error. Hay muchas formas de hacerlo pero es interesante considerar el control de versiones así como para mantener los scripts de administración.
Usted debe saber realizar una copia de seguridad usando tar y rsync así como las operaciones básicas de git para controlar versiones y restaurar un error.

Anexo I: Contextualización de Escenarios

6.1. Lección 1

Usted está trabajando en una empresa proveedora de servicios y recibe la solicitud de un cliente que desea tener un servidor para la implantación de un comercio electrónico.
Una vez diseñado y configurado el almacenamiento, debe enviarle las credenciales de acceso para que el equipo de desarrollo pueda desplegar la aplicación. Dado que FTP no es un protocolo seguro, decide configurar SSH para que el usuario sea capaz de acceder a la consola de manera segura y asegurada (*secure and hardened*). Para ello, configurará el servicio de modo que el usuario root no pueda tener acceso, que sea posible la autenticación por llave pública, restringirá el usuario que puede acceder acceso y limitará los ataques por fuerza bruta. Además cambiará el puerto por defecto para que no esté en la lista de barrido rápido de `nmap`.

6.2. Lección 2

El cliente del comercio electrónico desea implementar una solución de Backup por si el servidor es vulnerado o por si prefiere cambiar de proveedor de servicios. Sin tener más información, se negocia que se pondrá disponible una copia del volcado de la BD en `/home/<nombre_usuario/` que estará comprimida junto con el histórico de los comandos ejecutados por el usuario (`~/.bash_history` y `~/.mysql_history`).
Para ello, creará un script `backup_cliente` que llamará al comando tar para archivar esa información y comprimirla. Para el desarrollo del script (python, bash o lo que usted prefiera) se le requerirá usar Git como control de versiones (puede usar el repo de la bitácora).
Finalmente, se le solicita que rsync esté disponible para descargar la información así que usted proporcionará el comando que deben ejecutar para que el archivo se sincronice. Así que antes de dar comunicar "Hecho" al cliente, lo prueba con un directorio.

6.3. Lección 3

Recibe una nueva llamada, el equipo de desarrollo no tiene conocimiento de sistemas y debe hacer que el servidor tenga configurado un servidor web para poder ejecutar

Wordpress así que decide instalar y configurar la pila (el stack) LAMPP.

Referencias

[1] Flatpack. https://www.flatpak.org/, 2022. [Online; consultada 4-Oct-2025].

[2] Daniel J. Barrett, Richard E. Silverman, and Robert G. Byrnes. *SSH, The Secure Shell: The Definitive Guide, 2nd Edition*. O'Reilly Media, Inc., 2005.

[3] Debian. Apt interfaz de línea de comandos. https://wiki.debian.org/AptCLI, 2022. [Online; consultada 9-Oct-2022].

[4] Cyril Jaquier et al. Fail2ban. https://www.fail2ban.org/wiki/index.php/Main_Page, 2016. [Online; consultada 14-Feb-2018].

[5] Fedora. Dnf. https://fedoraproject.org/wiki/DNF?rd=Dnf, 2017. [Online; consultada 14-Feb-2018].

[6] Red Hat. Proyecto cockpit. https://cockpit-project.org, 2024. [Online; consultada 9-Sept-2024].

[7] Red Hat. Red hat documentation. cap. 6 introducción a nftables. https://docs.redhat.com/es/documentation/red_hat_enterprise_linux/8/html/securing_networks/getting-started-with-nftables_securing-networks, 2025. [Online; consultada 4-Oct-2025].

[8] Canonical Ltd. Sanpcraft. https://snapcraft.io/, 2018. [Online; consultada 4-Oct-2025].

[9] W. Curtis Preston. *Backup & Recovery*. O'Reilly Media, Inc., 2006.

Ingeniería de Servidores (2025-2026)
Grado en Ingeniería Informática
Universidad de Granada

Práctica 3 : Monitorización, Automatización y "Profiling"

Índice

OBJETIVOS MÍNIMOS

1. Conocer y saber usar las herramientas que permitan obtener datos sobre el sistema a nivel hardware y software (SO y servicios).
2. Saber interpretar los resultados proporcionados por las aplicaciones de monitorización.
3. Conocer los archivos que proporcionan información del sistema.
4. Tener conocimiento básico sobre automatización y orquestación
5. Ser capaz de configurar y utilizar un monitor de sistema

Contenido de las lecciones

1. Monitorización del RAID1, Monitores y Automatización
2. Zabbix

Competencias que se trabajarán

Competencias Básicas

1. CB2. Que los estudiantes sepan aplicar sus conocimientos a su trabajo o vocación de una forma profesional y posean las competencias que suelen demostrarse por medio de la elaboración y defensa de argumentos y la resolución de problemas dentro de su área de estudio.

Específicas de la Asignatura

1. R1. Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente.
2. R2. Capacidad para planificar, concebir, desplegar y dirigir proyectos, servicios y sistemas informáticos en todos los ámbitos, liderando su puesta en marcha y su mejora continua y valorando su impacto económico y social.
3. R5. Conocimiento, administración y mantenimiento de sistemas, servicios y aplicaciones informáticas.

Competencias Específicas del Título:

1. E4. Capacidad para definir, evaluar y seleccionar plataformas hardware y software para el desarrollo y la ejecución de sistemas, servicios y aplicaciones informáticas.

Competencias Transversales o Generales:

1. T2. Capacidad de organización y planificación así como capacidad de gestión de la Información.

1. Introducción

Esta práctica consiste en la utilización de herramientas de monitorización del sistema para visualizar cómo se comporta el sistema ante ciertas actividades que los usuarios u otros servicios generan.
Las herramientas de monitorización presentan medidas del sistema permitiendo generar informes e históricos que puedan ser de utilidad para un análisis a posteriori (off-line) o para tomar decisiones sobre la marcha (on-line).
En última instancia, el objetivo de esta práctica es monitorizar varios sistemas que tienen servicios instalados siendo consciente de cómo se realiza el proceso tanto a bajo nivel como usando herramientas que nos permiten trabajar a alto nivel.
A raíz de la solución basada en Grafana y la inclusión de IA en los stacks de monitorización, se está extendiendo el uso del concepto de *observabilidad* que difiere en matices filosófifcos de la monitorización añadiendo el componente de predecir y añadir acciones proactivas en base a las métricas y su análisis. Según Amazon " *El monitoreo recopila datos sobre los componentes individuales y la observabilidad analiza el sistema distribuido como un conjunto.*" [1]. En el caso de IBM, podemos encontrar una referencia [1] con un contenido bastante completo y correcto, no deje de consultarlo para aprender sobre el término y continuar con la disciplina fundamental del *Site Reliability Engineer* que puede ser uno de sus múltiples itinerarios profesionales. Le dejamos unas referencias sobre el tema por si desea ir profundizando: `https://sre.google/books/`.

2. Monitores para Hardware

Además del estado del software del servidor, también existen programas que nos permiten ver el estado del hardware de nuestra máquina. En primer lugar, muchas BIOS (*B*asic Input Output System) (ya en extinción debido a la aparición de los Universal Extended FIrmwares, UEFI) nos permiten acceder a cierta información sobre el estado del HW, sin embargo, para no tener que reiniciar, podemos utilizar otras herramientas. Concretamente, para Linux está: `hddtemp` para la temperatura del HD tenemos y el proyecto lm-sensors: `https://github.com/lm-sensors/lm-sensors` (con su correspondiente GUI: xsensors).
Al estar trabajando con máquinas virtuales, la ejecución y obtención de resultados de estos monitores no aporta nada ya que el hardware que monitorizan es virtual.
No obstante, debemos ser conscientes de que hay ciertos comandos que ya hemos visto que nos permiten listar el hardware disponible: `lspci, lsusb, lshw` nos muestran los dispositivos conectados a los buses e información general sobre el HW del equipo. Pruebe a ejecutarlos y ver qué muestran.

[1] https://aws.amazon.com/es/compare/the-difference-between-monitoring-and-observability/

2.1. Mensajes del kernel: dmesg

El kernel de Linux permite conocer qué actividad ha ocurrido gracias a los mensajes que proporciona el kernel. Esto es especialmente útil para detectar problemas con el HW o periféricos.

3. Monitores para Software

3.1. Subsistema de archivos

En linux (UNIX) todo se manipula a través de archivos de una manera cómoda y transparente. Existe un directorio especial: /proc (visto en clase de teoría) y /var (algo se ha comentado también al respecto) que nos pueden dar información tanto del hardware como del software.
En estos directorios se encuentran archivos fundamentales en la monitorización del comportamiento del sistema, de las aplicaciones y de los usuarios. Cabe destacar el subdirectorio `/var/log` que contiene los archivos en los que se van volcando los logs de los servicios y algunas aplicaciones.
Hay que tener ciertas precauciones ya que, una mala gestión de los archivos de bitácora puede resultar en un sistema caído. Para evitar que los logs crezcan indefinidamente, se realiza una rotación de estos archivos (logrotate).
Gracias a estos archivos podemos identificar errores y problemas, además esta tarea puede ser automatizada. Como ejemplo retomaremos los RAID1 en Ubuntu server de la primera práctica.
Usted debe conocer cómo identificar, monitorizar y reconstruir los discos RAID que tiene configurados en sus máquinas virtuales. Puede encontrar información imprescindible en las páginas del manual para mdadm y el archivo `/proc/mdstat` además de en [2, 3].

3.2. Monitorizando un servicio o ejecución de un programa: strace

Hay un conjunto de programas que permiten hacer una traza de las llamadas al sistema realizadas por un programa (servicio) en ejecución, p.ej. strace (system call tracer).
Este tipo de programas pueden ser útiles de cara a detectar problemas que no se muestran en los archivos de "log".
En `http://chadfowler.com/2014/01/26/the-magic-of-strace.html` tiene ejemplos de uso y podrá apreciar la utilidad de este tipo de programas.

4. Monitores generales

Además de los comandos integrados vistos en teoría y los que han usado en prácticas, existen otros programas muy populares que permiten monitorizar el sistema.

Hay algunos de entorno corporativo de grandes empresas como NetApp (http://www.netapp.com/es/products/management-software/), aunque los que se verán en las siguientes subsecciones también son usados por grandes instituciones y empresas.

4.1. Munin

Munin (significa memoria) está disponible en http://munin-monitoring.org/.
Para su instalación (en CentOS) puede hacerlo compilando el código fuente (como ha podido hacer con lm_sensors) alojado en la página o a través del paquete disponible en el repositorio EPEL (http://fedoraproject.org/wiki/EPEL/es). "¿Cómo puedo utilizar estos paquetes adicionales?" es el título de la subsección dentro de la página donde se explica cómo activar el repositorio que contiene los paquetes. Tan solo debe instalar un .rpm y podrá usar yum para instalar munin.
Para Ubuntu, está disponible sin tener que añadir ningún repositorio adicional.
En http://demo.munin-monitoring.org/ puede ver una demo del sistema de monitorización en funcionamiento.

4.2. Nagios y Naemon

Es otro software muy usado para monitorizar sistemas. Recientemente ha pasado por una transformación de proyecto de la comunidad a proyecto empresarial. Debido a esto, ha aparecido Naemon http://naemon.io que acaba de hacer una nueva *release*. Para ver una demo, podemos identificarnos a través de la demo que hace uso de la popular interfaz Thruk (https://demo.thruk.org/demo/thruk/cgi-bin/login.cgi?demo/thruk).

4.3. Ganglia

Es un proyecto alojado en http://ganglia.sourceforge.net/ que monitoriza sistemas de cómputo distribuidos (normalmente para altas prestaciones) y permite una gran escalabilidad.
Como puede verse en su página, importantes instituciones y compañías lo usan (p.ej. UC Berkely, MIT, Twitter, ...).

4.4. Grafana

Este software se autodefine como la pila de observabilidad (cualquier cosa que desee monitorizar) [4]. Para construir su pila tiene distintos componentes que se estructuran en complementos (plugins), bitácoras (logs), paneles (dashboards) y alertas (alerts). Dentro de esta visión amplia de la monitorización, incluyen un sistema de traza que se solapa con el concepto de *profiling*.
Es una solución de código abierto con posibilidad de contratar su servicio alojado en la nube y de manera autogestionada para empresas con mayor presupuesto.
Dentro de las empresas que han adoptado esta tecnología encontramos algunas muy populares y muchos casos de éxito de implantación tienen un artículo asociado explicando cómo

la han adoptado, se le anima a revisar algunos de éstos que suelen incluir algún vídeo explicativo (como, por ejemplo, el de una famosa empresa de citas: https://grafana.com/blog/2019/03/26/tinder-grafana-a-love-story-in-metrics-and-monitoring/?plcmt=footer.
Como suele ser habitual, tenemos la posibilidad de jugar con una demo disponible en: https://play.grafana.org/d/000000012/grafana-play-home?orgId=1.

4.5. Netdata

Otra empresa que ha ido desarrollando un producto un poco más enfocado a nicho ha sido Netdata. Su especialización ha sido la minimización del impacto de la solución de monitorización pudiendo observar los datos en tiempo real.
Recientemente han incorporado otros aspectos (ya presentes en otras soluciones de monitorización) como la inclusión de aprendizaje automático para generar alertas. Además, conscientes de la popularización de Grafana y viendo que es una solución que abarca más aspectos, han desarrollado una integración para esta otra solución de monitorización.
Es un proyecto opensource y miembro de la Linux Foundation. Pueden probar el producto en https://london.my-netdata.io/#menu_system_submenu_cpu;after=-480;before=0;theme=slate;help=true;utc=Europe/London.

4.6. Elastic (pila ELK)

Aunque no empezó su desarrollo como una herramienta de monitorización sino como una de búsqueda de información, el ecosistema ha crecido y se ha definido la pila ELK: Elastic + Logstash + Kibana. Gracias a estos tres componentes, se puede almacenar información (logstash) mediante lo que denominan *beats*, indexarla y hacer bśuquedas eficientes (Elastic) y visualizarla en paneles (Kibana).
Existen más plugins y es una solución bastante popular por sus integraciones, no obstante, es software privativo.

4.7. ZABBIX

Es otro programa que permite monitorizar el sistema también de código abierto. Su instalación es relativamente sencilla ya que solo hay que añadir los repositorios (visto en la práctica anterior) e instalarlo con el gestor de paquetes.
Este es el sistema de monitorización en el que nos centraremos en las lecciones. En [5] encontrará toda la información que necesitará para el seguimiento de las lecciones.
El objetivo es que usted sea capaz de instalar este sistema de monitorización y configurar éste para que se monitorice a sí mismo así como a Rocky (a través del agente) además de conocer cómo interactuar con el sistema de monitorización usando la API que proporcionar y conocer en qué consiste el protocolo SNMP.
Le proponemos el siguiente ejercico que debe realizar de manera obligatoria:

Ejercicio 1:
Realice una instalación de Zabbix 7.4 en su servidor con Debian 13 y configure para que se monitorice a él mismo y para que monitorice a la máquina con Alma Linux.
Puede configurar varios parámetros para monitorizar, uso de CPU, memoria, etc. pero debe configurar de manera obligatoria, como mínimo, la monitorización de los servicios SSH y HTTP.
Es recomendable, para reforzar el aprendizaje, que documente el proceso de instalación y configuración indicando las referencias que ha utilizado así como los problemas que ha encontrado.

5. Automatización

5.1. cron y systemd

Tradicionalmente, el servicio cron ha permitido ejecutar cada cierto intervalo de tiempo una tarea concreta. Esto es muy útil de cara a recopilar información o monitorizar el sistema realizando una tarea concreta y lanzar alertas como, por ejemplo, enviar un correo electrónico cuando la carga esté por encima de un valor determinado (aunque algunos comandos de monitorización permitan hacer esa tarea directamente).
Con la introducción de systemd, cron puede seguir siendo usado pero su funcionamiento está basado en los *timers* que activan *services*. La documentación más recomendable sobre la gestión de systemd y los servicios está en los manuales de Red Hat, concretamente en `https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system_administrators_guide/sect-managing_services_with_systemd-unit_files`, aunque en `https://people.redhat.com/pladd/systemd_NYRHUG_2016-03.pdf` tiene un resumen con más enlaces a documentación.
Como ejemplo, podemos tomar este script escrito en Python que nos permite monitorizar si algún dispositivo de nuestro RAID ha fallado (puede ver más ejemplos en `http://echorand.me/site/notes/articles/python_linux/article.html`):

```
import re
f=open('/proc/mdstat')
for line in f:
  b=re.findall('\[[U]*[_]+[U]*\]',line)
  if(b!=[]):
     print("---ERROR en RAID---")
print("---OK Script---")
```

Podemos automatizar la ejecución de este script en systemd definiendo un *timer* dentro del directorio `/etc/systemd/system/` que se encarga de gestionar un servicio. Por tanto, hemos de crear dos archivos: mon_raid.timer y mon_raid.service

```
[Unit]
Description=Monitor RAID status
[Timer]
OnCalendar=minutely
```

```
[Install]
WantedBy=timers.target}

[Unit]
Description=Monitor RAID status
[Service]
Type=simple
ExecStart=/usr/bin/python3 /home/alberto/mon-raid.py
```

Para que pueda funcionar hemos de activar y habilitar el *timer* (de manera análoga a lo que hicimos con los servicios tras instalarlos en la Práctica 2). Una vez hecho esto, podremos monitorizar su ejecución gracias al comando journalctl,
p.ej. `journalctl -u mon-raid --since="yesterday"`.

5.2. Scripts

5.2.1. Shell y comandos del sistema: grep, find, awk y sed

Como ya han estudiado en Sistemas Operativos, el shell, en sus diferentes versiones (z, bourne, bash, c, ...) permite programar permitiendo hacer tareas de manera automática. Además de lo que ya saben, es importante mencionar tres comandos muy útiles de cara a automatizar tareas: grep, find, awk y sed, puede consultar algunos ejemplos en [6, 7].
Por ejemplo con sed puede buscar una cadena en un archivo y reemplazarla por otra, así podría dar acceso por contraseña durante unos instantes al servicio ssh para que los usuarios puedan copiar su llave pública.
Con awk puede programar la manipulación del texto así como generar salidas más completas a partir de la información en un archivo.
Con grep puede realizar filtrado de cadenas, útil cuando un archivo, listado, etc tiene unas dimensiones grandes, p.ej. `ps -Af | grep firefox` nos mostraría la información del proceso firefox (descartando el resto de información) Por último, con find, aunque probablemente ya lo haya usado en SOs, puede buscar archivos y, una vez encontrados, realizar acciones sobre ellos, p.ej. `find /home/alberto/docs -name '*pdf' -exec cp {} ~/PDFs \;` copiará todos los archivos cuyo nombre termine en pdf y los copia en la carpeta /home/alberto/PDFs

5.2.2. Python y PHP

Uno de los lenguajes más populares a día de hoy para programar servidores, concretamente páginas web dinámicas, es PHP. Existen numerosos y extendidos CMS (Content Managemente System) escritos en PHP, p.ej. Wordpress, Joomla, CakePHP, etc. Para poder programar sin tener que partir de cero existen varios frameworks como Synfony y Zend entre otros.
Python es un lenguaje de scripting con una creciente popularidad y se presenta como una gran alternativa a PHP (el framework django se basa en Python). También está ganando usuarios en el mundo de la investigación presentándose como una alternativa a Matlab (sin tener en cuenta Octave).

De cara a la asignatura, debido a su facilidad para manipular cadenas de texto mediante bibliotecas, nos puede permitir escribir tareas automatizadas que manipulen archivos de configuración. Como ejemplos muy sencillos que no requieren excesivo conocimiento del lenguaje tenemos los proporcionados en [8]. En [9] también hay otros scripts que muestran posibles casos prácticos además de todas las *recetas* disponibles en `https://github.com/ActiveState/code/tree/master/recipes/Python`.
Como último ejemplo, es oportuno citar la biblioteca para hacer uso de la API de Zabbix disponible en [10].

5.3. A nivel de Plataforma: Ansible

Además de todo lo visto, también existen interfaces que permiten programar scripts y visualizar su ejecución de una manera cómoda y visual. Por falta de tiempo, no los cubriremos en la asignatura con detalle, no obstante, sí que veremos en una lección el funcionamiento básico de Ansible y se le anima a visitar las webs de los proyectos y ver algún vídeo relacionado.
P.ej. Puppetlabs `http://puppetlabs.com/` Ansible `http://www.ansible.com/home` Rundeck `http://rundeck.org/screencasts.html`

Ejercicio 2: Usted deberá saber cómo instalar y configurar Ansible para poder hacer un ping a las máquinas virtuales de los servidores y ejecutar un comando básico (p.ej. el script de monitorización del RAID1). También debe ser consciente de la posibilidad de escribir acciones más complejas mediante playbooks escritos con YAML como, por ejemplo, asegurarse de que tenemos la última verisón instalada de httpd y que está en ejecución.

La plataforam Ansible también nos permite tener reactividad ante ciertos eventos, eso conlleva un nivel de automatización elevado que minimiza la intervención en las operaciones. Existen una serie de fuentes que, tras cumplir una serie de reglas permiten lanzar acciones de Ansible. En otras palabras, nuestra solución de monitorización, por ejemplo, Zabbix, puede ser considerada una fuente que envía al servidor un evento, tras hacer la comprobación en base a unas reglas, Ansible permite la ejecución automática de operaciones para reaccionar a ese evento [2].
Aunque posee la documentación oficial de Ansible, es recomendable leer la entrada de "Ansible Basics" dentro del libro disponible en la documentación de Rocky Linux denominado "Learning Ansible".

6. Profiling

6.1. Scripts

Para estudiar el comportamiento de los scripts, independientemente el lenguaje que usemos, podemos usar varios profilerers, p.ej. en Bash podemos usar la opción set -x y

[2] `https://www.redhat.com/en/blog/getting-started-with-event-driven-ansible`

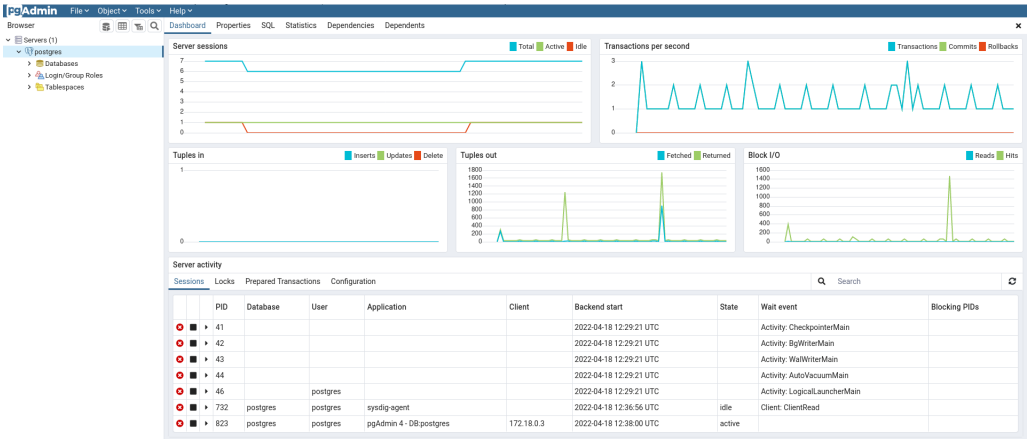

Figura 6.1: Figura de ejemplo de monitorización de la BD con pgadmin.

modificar la variable local PS4 para que muestre el tiempo `http://www.tldp.org/LDP/Bash-Beginners-Guide/html/sect_03_02.html`. En el caso de PHP tenemos el proyecto desarrollado inicialmente por Facebook:
XHProf: `http://pecl.php.net/package/xhprof` `https://github.com/facebook/xhprof`
Además de la documentación oficial, en la página de un prestigioso sistema de enseñanza virtual hay unos tutoriales interesantes sobre cómo realizar el profiling: `http://docs.moodle.org/dev/Profiling_PHP` .
En el caso de Python se puede utilizar el módulo cProfile. Para más información consulte en `https://docs.python.org/3/library/profile.html` .

6.2. SQL

El mismo MySQL (y MariaDB) que instalamos en la práctica anterior tiene un "profiler" para analizar cuánto tardan las consultas.
Puede ver la documentación en:
`http://dev.mysql.com/doc/refman/5.0/en/show-profiles.html` `http://dev.mysql.com/doc/refman/5.0/en/show-profile.html`
Además, es especialmente útil la herramienta MySQLWorkBench y otros comandos disponibles como mysqloptimize que optimizan la ejecución.
Pruebe a obtener un profile de una tabla cogiendo todos los atributos (`SELECT * ...`) y a escoger únicamente uno o dos (`SELECT ID,AGE ...`) y compare el tiempo que requiere consultar información extra que no necesitamos.
Por otra parte, para PostgreSQL también disponemos de pgAdmin (`https://www.pgadmin.org/`) que nos permite monitorizar cómodamente nuestra base de datos con independencia del SO en el que la estemos ejecutando.

Referencias

[1] IBM, "Observabilidad." https://www.ibm.com/es-es/topics/observability, 2024. [Online; consultada 31-Octubre-2024].

[2] David Greaves *et al.*, "Detecting, querying and testing." https://raid.wiki.kernel.org/index.php/Detecting,_querying_and_testing, 2006–2011. [Online; consultada 10-Noviembre-2021].

[3] David Greaves *et al.*, "mdstat." https://raid.wiki.kernel.org/index.php/Mdstat, 2007–2013. [Online; consultada 10-Noviembre-2021].

[4] Grafana Labs, "Granafa." https://grafana.com/, 2022. [Online; consultada 31-Octubre-2022].

[5] Zabbix SIA, "Manual de zabbix." https://www.zabbix.com/documentation/5.0/manual, 2020. [Online; consultada 14-September-2020].

[6] T. G. Stuff, "Ejemplos de sed." http://www.thegeekstuff.com/2009/10/unix-sed-tutorial-advanced-sed-substitution-examples/, 2017. [Online; consultada 10-Noviembre-2021].

[7] T. G. Stuff, "Ejemplos de awk." http://www.thegeekstuff.com/2010/01/awk-introduction-tutorial-7-awk-print-examples/, 2017. [Online; consultada 10-Noviembre-2021].

[8] A. Saha, "Linux system mining with python." http://echorand.me/site/notes/articles/python_linux/article.html, 2013. [Online; consultada 10-Noviembre-2021].

[9] J. K. (IBM), "Python for system administrators." https://www.ibm.com/developerworks/aix/library/au-python/index.html, 2013. [Online; consultada 10-Noviembre-2021].

[10] L. Cyca, "Pyzabbix." https://github.com/lukecyca/pyzabbix, 2017. [Online; consultada 10-Noviembre-2021].

Ingeniería de Servidores (2025-2026)
Grado en Ingeniería Informática
Universidad de Granada

Práctica 4 : Benchmarking y Ajuste del Sistema

Índice

OBJETIVOS MÍNIMOS

1. Conocer varios benchmarks para diferentes servicios.
2. Saber comparar distintas configuraciones (o implementaciones) de servicios en base a benchmarks.
3. Aplicar un test de carga a una aplicación basada en microservicios con API Rest.
4. Conocer y saber cómo modificar el valor algunos parámetros que pueden mejorar las prestaciones.

Contenido de las lecciones

1. Benchmarks suites y Jmeter

Competencias que se trabajarán

Competencias Básicas

1. CB2. Que los estudiantes sepan aplicar sus conocimientos a su trabajo o vocación de una forma profesional y posean las competencias que suelen demostrarse por medio de la elaboración y defensa de argumentos y la resolución de problemas dentro de su área de estudio.

Específicas de la Asignatura

1. R1. Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente.
2. R2. Capacidad para planificar, concebir, desplegar y dirigir proyectos, servicios y sistemas informáticos en todos los ámbitos, liderando su puesta en marcha y su mejora continua y valorando su impacto económico y social.
3. R5. Conocimiento, administración y mantenimiento de sistemas, servicios y aplicaciones informáticas.

Competencias Específicas del Título:

1. E4. Capacidad para definir, evaluar y seleccionar plataformas hardware y software para el desarrollo y la ejecución de sistemas, servicios y aplicaciones informáticas.

Competencias Transversales o Generales:

1. T2. Capacidad de organización y planificación así como capacidad de gestión de la Información.

1. Introducción

Como ejercicio final de prácticas, tras haber instalado, configurado y monitorizado servicios, vamos a aplicar una carga que nos pueda servir cómo indicador para ver cómo podría responder nuestro sistema. En base a los resultados obtenidos (y monitorizados) consideraremos también algunos parámetros que puedan mejorar las prestaciones ante cargas concretas.
En primer lugar estudiaremos un conjunto de benchmarks para luego centrarnos en uno concreto para un servicio que ya conocemos y que trataremos de optimizar su comportamiento.

2. Benchmarks

Hay muchos para cada tipo de servicio (p.ej. Para DNSs: NameBench y GRC's DNS Benchmark, `http://sourceforge.net/directory/os:linux/?q=benchmark`) pero puede resultar más interesante programar uno para analizar algún parámetro concreto que deseemos. Siempre debemos tener en cuenta un mínimo de cuestiones al hacerlo:

1. Objetivo del benchmark.
2. Métricas (unidades, variables, puntuaciones, etc.).
3. Instrucciones para su uso.
4. Ejemplo de uso analizando los resultados.

2.1. Phoronix

Phoronix es una plataforma que permite ejecutar un conjunto de benchmarks bajo la agrupación openbenchmarking.org. La aplicación puede instalarse a través de los gestores de paquetes ya vistos en los guiones anteriores.

Ejercicio propuesto: una vez que haya indagado sobre los benchmarks[a] disponibles, seleccione como mínimo uno de ellos y proceda a ejecutarlos en Debian y Alma Linux. Comente las diferencias de los resultados de ejecución.

[a] Cuidado, hacemos referencia a benchmark no a suite, que son más extensas y requieren más dependencias.

Ejercicio opcional: tras ver la parte de la práctica relacionada con contenedores, pruebe a ejecutar uno de los benchmarks de los seleccionados arriba y analice las diferencias en los resultados obtenidos en la MV directamente. Puede lanzar un contenedor con la imagen de phoronix, consulte: `https://www.phoronix.com/scan.php?page=article&item=docker-phoronix-pts&num=1` en su anfitrión (en la máquina virtual probablemente se quedará sin espacio). Otra posibilidad es ejecutar una imagen de contendedor de Debian y ahí instalar phoronix.

Otra funcionalidad interesante es la interfaz phoromatic que permite orquestar y automatizar la ejecución de benchmarks en múltiples máquinas.
Podemos hacer uso de una interfaz web mediante la instalación de un navegador, p.ej. firefox, y ejecutando el comando correspondiente indicado en la documentación [8]. La interfaz realiza una monitorización de la ejecución que podemos contrastar simultaneamente con la que se realiza mediante Zabbix como se aprecia en la Figura 2.1.

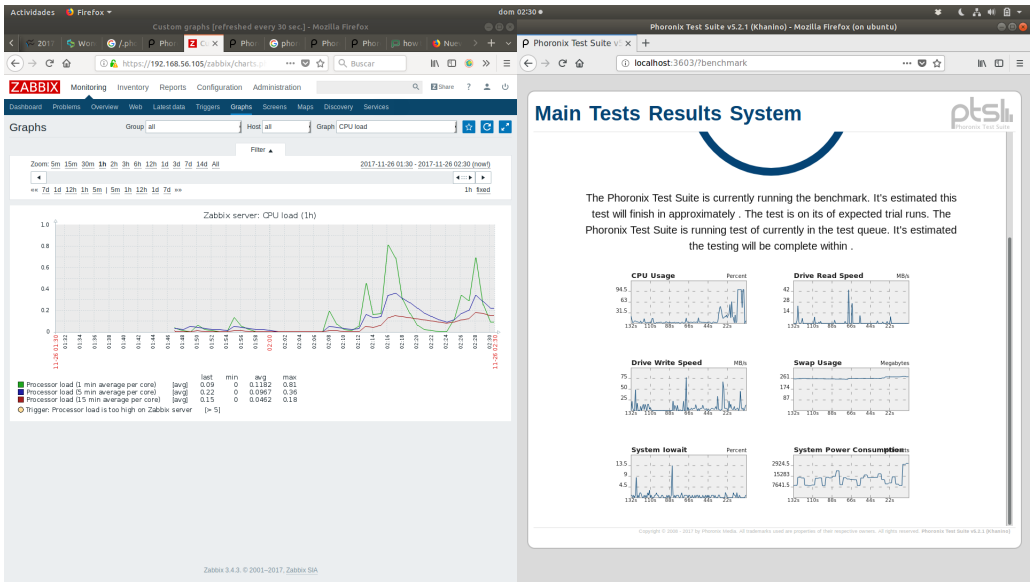

Figura 2.1: Ejemplo de monitorización con Zabbix y phoronix durante la ejecución de un benchmark

2.2. ab y Jmeter

Dentro de los benchmarks más populares para servidores web podemos encontrar la herramienta para "Apache HTTP server benchmarking "(comando ab) [2]. Concretamente, su misión es mostrar cuantas peticiones por segundo el servidor de HTTP (puede ser Apache, Nginx, IIS o cualquier otro) es capaz de servir.

> **Ejercicio propuesto**: Dentro de los parámetros básicos que usted debe conocer encontramos la opción que especifica la concurrencia así como el número de peticiones. Usted debe probar a ejecutar el benchmark (monitorizar su ejecución en el cliente y en el servidor) con sus máquinas virtuales (Debian y Alma Linux) y obtener conclusiones respecto al tipo de concurrencia que es capaz de generar ab además de comparar resultados.

Aunque tenemos muchas posibilidades usando ab de manera correcta, hay otras herramientas que nos permiten hacer tests más complejos, concretamente vamos a ver Jmeter `http://jmeter.apache.org/` [4]. Este software se autodefine como una aplicación

"*...designed to load test functional behavior and measure performance...*", es decir, no establecen ningún tecnicismo concreto ya que pueden medir y generar carga de forma genérica para varios elementos.
Ha sido comparado por la empresa Octoperf con Gatling (otra herramienta popular para realizar test de estrés [6]) obteniendo la conclusión de que ambos tienen un comportamiento y capacidades similares [11].
Algunas características interesantes de Jmeter es que puede crear concurrencia real debido a la posibilidad de usar varias hebras dentro de la mismas CPU así como distribuir la creación de carga en varias máquinas. La posibilidad de ejecutarse en modo de línea de comandos permite aligerar la carga de la máquina que está generando las peticiones al servidor además de permitir la automatización de ciertos test. También es muy interesante la funcionalidad que nos permite configurar como proxy de nuestro navegador a Jmeter para que vaya registrando nuestra navegación como usuarios y luego podamos replicar esas peticiones de manera automática y paralela.
Otras herramientas (por citar algunas más) que le puede ser interesantes seguir son:

- Locust `https://locust.io/` que permite escribir en código (con Python) los test sin necesidad de interfaces permitiendo comprobar la escalabilidad sin límites en lo que se refiere a la creación de hebras.
- Molotov `https://molotov.readthedocs.io/en/stable/index.html`, con la filosofía de mantener las operaciones simples pero veloces.
- Taurus `https://gettaurus.org/` para automatizar test de rendimiento.

Ejercicio propuesto: tras probar un test básico para una web [5], debe ser capaz de generar una carga a una aplicación o CMS. Como ejemplo de práctica utilizaremos Jmeter para hacer un test sobre una aplicación que ejecuta sobre dos contenedores (uno para la BD y otro para la aplicación en sí). El código está disponible en `https://github.com/aguillenATC/ISE-P4App` donde se dan detalles sobre cómo ejecutar la aplicación en una de nuestras máquinas virtuales (El repo, contiene una refactorización del un código programado originariamente por David Palomar con una mala praxis en lo que al uso de git se refiere, identifique los errores y aprenda para no repetirlos).
El test de Jmeter debe incluir los siguientes elementos:

- El test debe tener parametrizados el Host y el Puerto en el Test Plan (puede hacer referencia usando $param)
- Debe hacer dos grupos de hebras distintos para simular el acceso de los alumnos y los administradores. Las credenciales de alumno y administrador se cogen de los archivos: alumnos.csv y administrador.csv respectivamente.
- Añadimos esperas aleatorias a cada grupo de hebras (Gaussian Random Timer)
- El login de alumno, su consulta de datos (recuperar datos alumno) y login del administrador son peticiones HTTP.
- El muestreo para simular el acceso de los administradores lo debe coger el archivo apiAlumnos.log (usando un Acces Log Sampler)
- Use una expresión regular (Regular Expressión Extractor) para extraer el token JWT que hay que añadir a la cabecera de las peticiones (usando HTTP Header Manager)

2.3. Breve introducción a los contenedores: docker

Ya hicimos referencia a los contenedores en la práctica 1 cuando hablamos del uso de las máquinas virtuales usando la analogía de proceso y hebra. Con ese conocimiento es suficiente para realizar la práctica si bien vamos a ilustrar esa compartición la Figura 2.2
El contenedor presenta a la aplicación una nueva capa abstracción sobre el sistema operativo haciéndola homogénea, facilitando la portabilidad y, sobre todo, el encapsulamiento de información. La principal ventaja respecto al uso de máquinas virtuales es el ahorro en la función de Hypervisor (que es reemplazado por el S.O. de la máquina física) y el ahorro del S.O. invitado (que es reemplazado por la aplicación docker). Como consecuencias directas los contenedores ocupan menos espacio , requieren menos recursos y son más veloces en el arranque.
Aunque hay muchas tecnologías que implementan este nivel de abstracción (LXC, OpenVZ, RKT, Virtuozzo, etc.), Docker es una de las más populares a día de hoy. Hay dos implementaciones, una para la comunidad y otra bajo licencia con funcionalidades extras

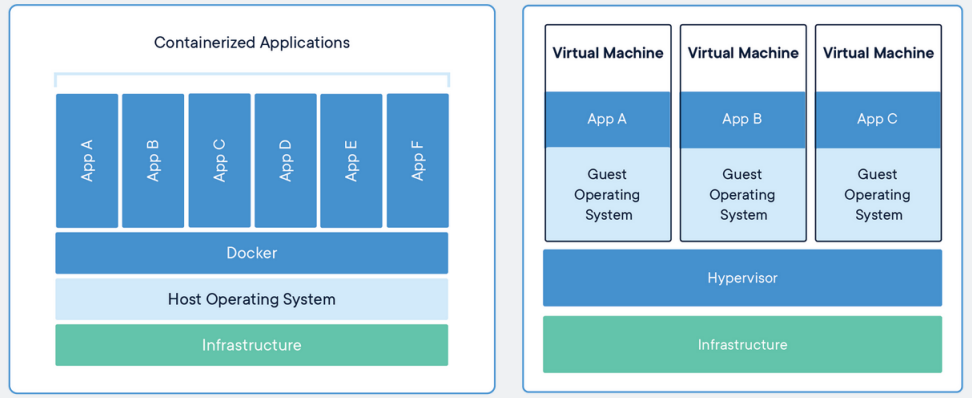

Figura 2.2: Nivel de virtualización de un contenedor. Fuente: https://www.docker.com/resources/what-container

[9], además puede ver una tabla comparativa en https://www.docker.com/products/container-runtime.
Actualmente, hay alternativas muy interesantes tecnologías como **podman** https://podman.io/ que nos permiten ejecutar contenedores con la misma sintaxis que docker y sin necesidad de tener un servicio como demonio con los riesgos de seguridad que eso conlleva[1]. Como artículo complementario a este hecho es interesante leer el artículo disponible en https://dev.to/martinheinz/it-s-time-to-say-goodbye-to-docker-386h.
En https://training.play-with-docker.com/ tiene cantidad de recursos para aprender los elementos básicos (y más avanzados). Concretamente, en https://training.play-with-docker.com/ops-s1-hello/ puede ejecutar el "hola mundo" sin necesidad de instalar nada en su máquina anfitriona e ir familiarizándose con esta tecnología. Se le recomienda que realice el tutorial para entender mejor lo que haremos a continuación.

2.3.1. Microservicios

La popularidad de los contenedores ha ido ligada al éxito de un tipo de arquitectura conocida como microservicios (*microservices*). En este tipo de arquitecturas tenemos distintos servicios que se ejecutan de forma independiente unos de otros y se comunican mediante APIs o sistemas de mensajería (RabbitMQ p.ej.).
Las ventajas de esta arquitectura son varias pero la flexibilidad y escalabilidad (vertical y horizontal) quizá sean las más destacables.
No obstante, hay que usarlos con precaución y siempre evaluando otras alternativas de implementación, prueba de ello es el caso de Amazon Prime y cómo abandonar los microservicios en favor del clásico monolito, les ha resultado beneficioso al ahorrarse todo el sobrecoste de cómputo: https://www.primevideotech.com/video-streaming/scaling-up-the-prime-video-audio-video-monitoring-service-and-reducing-costs-by-90

[1] Hasta el punto que se recomienda hacer un alias docker podman

2.3.2. Instalación de la aplicación para el test con Jmeter

Una vez está tenemos el software de contenedores instalado con docker o podman, podemos desplegar la aplicación en nuestro Debian, para ello lo primero que debemos hacer es clonar el repositorio de:

```
git clone https://github.com/aguillenATC/ISE-P4App
```

Tras esto, tendremos un directorio nuevo: iseP4JMeter al cual podremos acceder y levantar la aplicación con docker compose:

```
cd iseP4JMeter
docker-compose up
```

o si hemos instalado podman-compose:

```
cd iseP4JMeter
podman-compose up
```

En pantalla tendremos mostradas las peticiones que vayamos haciendo, podemos dejar docker como demonio o background con la opción -d.

2.3.3. Detalles sobre el dockerfile y compose

El *dockerfile* es el archivo donde indicamos todos los comandos necesarios para crear una imagen de docker y las acciones que debe llevar a cabo sobre esta (copiar archivos, instalar paquetes, modificar configuraciones). Desde un punto de vista muy abstracto podríamos decir que es el *playbook* de Ansible que aplica docker al levantar el contenedor.
En este dockerfile, dos elementos muy relevantes son la configuración del contenedor en lo que a red y almacenamiento se refiere.
En la aplicación sobre la que aplicaremos la carga tenemos dos contenedores que se configuran con los respectivos dockerfiles:
Contenido del dockerfile de la máquina que contiene la aplicación (en node.js):

```
FROM node:8
RUN mkdir -p /usr/src/app
COPY . /usr/src/app
EXPOSE 3000
WORKDIR /usr/src/app
RUN ["npm", "update"]
ENV NODE_ENV=production
CMD ["npm","start"]
```

En este archivo, indicamos la imagen a partir de la que crear el contenedor (node) con la etiqueta 8. Tras descargar la imagen (en caso de que no esté descargada ya) ejecutará un comando (con RUN) para crear un directorio y copiar los archivos del repositorio

en el contendor. Posteriormente, indicamos el puerto en el que queramos que escuche el contenedor (EXPOSE), esto está relacionado con la opción -p al ejecutar el contenedor ya que podemos hacer un mapeo del puerto del anfitrión ejecutando los contenedores con los que están exponiendo éstos. Por último se indica que la aplicación comience su ejecución tras configurar una variable de entorno.
Contenido del dockerfile de la máquina que contiene la base de datos (mongodb):

```
FROM mongo
COPY ./scripts/* /tmp/
RUN chmod 755 /tmp/initializeMongoDB.sh
WORKDIR /tmp
CMD ./initializeMongoDB.sh
```

En este caso, la imagen de partida es de mongo, una vez disponible, se copian unos scripts del repositorio al contenedor y se ejecutan (tras cambiar los permisos correspondientes). Por último se lanza un script para rellenar la base de datos.
Como ya hemos comentado, la herramienta para configurar varios contenedores de manera que den un único servicio es docker-compose. Cuando la invocamos, ésta lee el archivo *compose* que especifica los siguientes elementos:

```
version: '2.0'
services:
    #MongoDB based in the original Mongo Image
    mongodb:
      image: mongo
      ports:
        - "27017:27017"

    # Initialize mongodb with data
    mongodbinit:
      build: ./mongodb
      links:
        - mongodb

    # Nodejs App
    nodejs:
      build: ./nodejs
      ports:
        - "3000:3000"
      links:
        - mongodb
```

Como siempre, le remitimos a la documentación oficial para ver los numerosos detalles sobre estos archivos: https://docs.docker.com/get-started/part2/#dockerfile https://docs.docker.com/engine/reference/builder/ https://docs.docker.com/compose/compose-file https://docs.docker.com/develop/develop-images/dockerfile_best-practices/

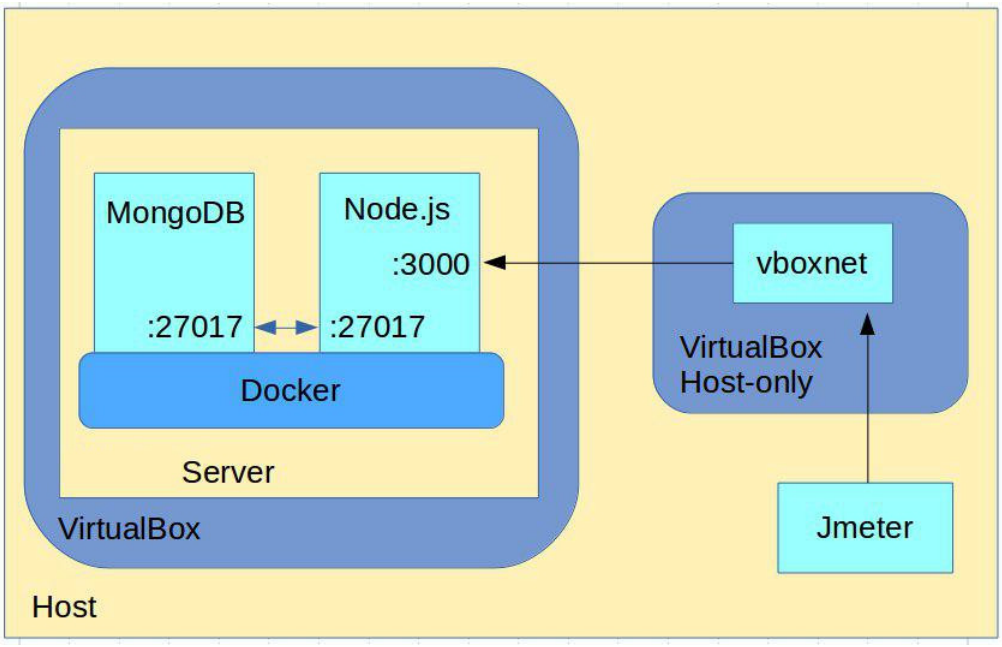

Figura 2.3: Descripción de los niveles de abstracción y los distintos elementos en ejecución.

2.3.4. La aplicación iseP4JMeter

La aplicación presenta una API Rest para obtener información sobre los alumnos tras autenticarse. Los detalles están descritos en el correspondiente README.md aunque, para más claridad, se ilustra el funcionamiento con el diagrama de la Figura 2.3
Además, se incorpora un script en bash que comprueba el correcto funcionamiento de la aplicación. En la Figura 2.4 está una descripción de la función de cada bloque.
Existe otra forma de comprobar que la aplicación está ejecutándose correctamente: podemos instalar un plugin a nuestro navegador (POSTMAN, RESTED, etc.) y generar las dos peticiones recuperando el token tras la autenticación y luego haciendo la petición tal y como muestra la Figura 2.5.
Respecto al mecanismo de autenticación, la API incorpora un tipo básico del protocolo HTTP que se usa introduciendo credenciales en la cabecera de la petición [1]. El fin de este mecanismo es reducir el ruido de Internet. Además de esta autenticación, la aplicación incorpora otro mecanismo basado en JSON Web Tokens (JWT) [12, 13]. Tiene varias ventajas frente a otros sistemas (claridad de JSON vs xml, tamaño, forma de cifrado) además de resolver el problema de *cross-domain*.
Para ilustrar el funcionamiento de la aplicación se muestra el diagrama de comunicaciones entre el cliente y el servidor en la Figura 2.6.

```
#!/bin/bash
SERVER=localhost
TOKEN=$(curl -s \
-u etsiiApi:laApiDeLaETSIIDaLache \
-d "login=mariweiss@tropoli.com&password=anim" \
-H "Content-Type: application/x-www-form-urlencoded" \
-X POST http://$SERVER:3000/api/v1/auth/login)
resultado=$?
if test "$resultado" != "0"; then
echo "ERROR: Curl fallo con resultado: $resultado"
fi
curl \
-H "Authorization: Bearer $TOKEN" \
http://$SERVER:3000/api/v1/alumnos/alumno/mariweiss%40tropoli.com
```

Definimos la IP (o hostname) del servidor

Hace un POST al servidor para recibir un token jwt Haciendo un Basic Auth (opción -u) y pasando credenciales en el cuerpo con -d

Comprueba que ha habido un resultado

Realiza la petición GET incluyendo el token recibido en la llamada POST anterior y muestra el resultado

Figura 2.4: Descripción de la funcionalidad de cada bloque del script.

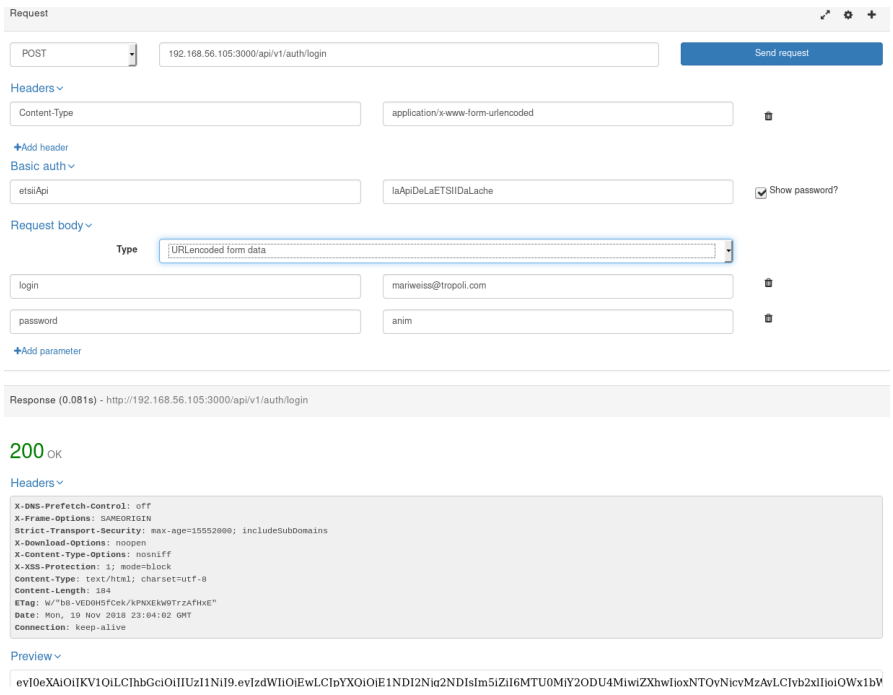

Figura 2.5: Ejemplo de petición HTTP usando el cliente Rested `https://addons.mozilla.org/en-US/firefox/addon/rested/`.

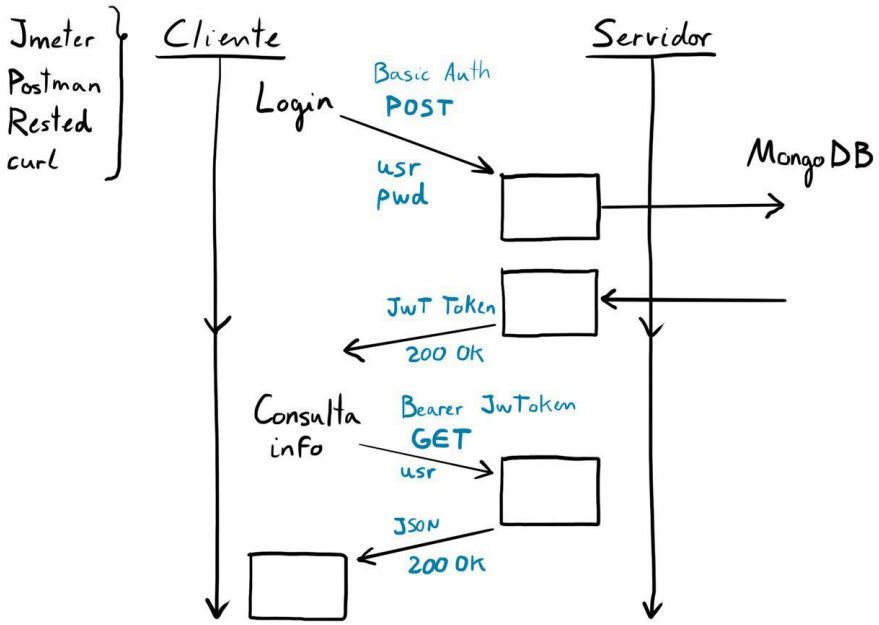

Figura 2.6: Descripción de los pasos de comunicación entre el cliente y el servidor.

3. Ajuste del sistema y de servicios: *Performance engineering*

Independientemente de la infraestructura que tengamos por debajo (HW, contenedores, máquinas virtuales, etc.) debemos ser conscientes de que cada capa tiene sus propios parámetros y ajustes y que, éstos, afectarán al rendimiento de la aplicación.
La modificación de los parámetros del sistema es una tarea compleja pues existen una gran cantidad de éstos para otra gran cantidad de subsistemas que, además, están relacionados. Tan solo en lo referente a los parámetros del kernel (el del anfitrión o el del invitado si usamos MV), tenemos un gran abanico de variables cuyos valores pueden ser números naturales, siendo muy difícil establecer o estimar una aproximación al óptimo (teniendo en cuenta que éste puede variar dependiendo de la carga). Algunos de estos elementos ajustables del kernel pueden ser accedidos y modificados a través del sistema de archivos (/proc/sys). Modificando los valores dentro de los archivos, es posible cambiar el comportamiento del sistema sin tener que reiniciarlo. Dentro de /proc/sys encontramos una estructura de directorios que separa los archivos de los distintos subsistemas (fs: sistema de archivos, kernel: kernel, vm: memoria virtual, dev: dispositivos, etc.)
Para evitar errores o valores inválidos, es recomendable usar el comando sysctl para modificar los parámetros del kernel.

3.1. Optimizando un servidor web

Como ya estamos familiarizados con la pila LAMP, vamos a considerar algunos parámetros y configuraciones interesantes que nos permitan obtener un mayor rendimiento.
No solo hay que tener en cuenta los elementos del sistema operativo y de los servicios en ejecución sino que cada código tiene unas características que nos obligan a monitorizar durante un tiempo la actividad e ir ajustando los valores en base a los resultados. Por ejemplo, si queremos tener una plataforma para enseñanza como puede ser moodle, ellos mismos ya tienen algunas sugerencias que, a posteriori, podremos modificar, pero son un buen punto de partida [10] y que incluyen las recomendaciones de la documentación de Apache [3]. Otro artículo en la misma línea es el incluido en la documentación de Wordpress, un Content Management System (CMS) de los más populares [14] y, además, en la documentación de MariaDB [7] tiene varios ejemplos al respecto.

Ejercicio opcional: con esta información usted podría modificar los parámetros de configuración de Apache, PHP o MariaDB para observar un cambio en el comportamiento del servidor (Alma Linux o Debian server) mediante la aplicación de un benchmark y analizando el cambio en las prestaciones o mediante el análisis de datos de monitorización ante una carga aplicada.
Puede configurar el servicio de la aplicación anterior o bien probar con otro CMS, como, por ejemplo, WordPress: `https://hub.docker.com/_/wordpress` que tiene su Docker compose disponible o el citado Moodle : `https://github.com/moodlehq/moodle-docker` .

Referencias

[1] . `https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication`. [Online; consultada 14-Nov-2019].

[2] The Apache Software Foundation. ab - apache http server benchmarking tool. `https://httpd.apache.org/docs/2.4/programs/ab.html`, 2017. [Online; consultada 14-November-2022].

[3] The Apache Software Foundation. Apache performance tuning. `http://httpd.apache.org/docs/current/misc/perf-tuning.html`, 2017. [Online; consultada 14-November-2022].

[4] The Apache Software Foundation. Best practices. `http://jmeter.apache.org/usermanual/best-practices.html`, 2017. [Online; consultada 14-September-2017].

[5] The Apache Software Foundation. Building a web test plan. `http://jmeter.apache.org/usermanual/build-web-test-plan.html`, 2017. [Online; consultada 14-September-2017].

[6] Gatling. Gatling. `http://gatling-tool.org/`, 2017. [Online; consultada 14-September-2017].

[7] MariaDB. Optimization and tuning. https://mariadb.com/kb/en/optimization-and-tuning/, 20. [Online; consultada 14-Nov-2022].

[8] Phoronix Media. Phoronix test suite. https://www.phoronix-test-suite.com/documentation/phoronix-test-suite.html, 2017. [Online; consultada 14-September-2017].

[9] Michael Friis (Docker Inc.). Announcing docker enterprise edition. https://www.docker.com/blog/docker-enterprise-edition/, 2017. [Online; consultada 14-Nov-2019].

[10] Moodle. Performance recommendations. https://docs.moodle.org/33/en/Performance_recommendations, 2017. [Online; consultada 14-September-2017].

[11] Jérôme Loisel CTO Octoperf. Jmeter vs gatling tool. https://octoperf.com/blog/2015/06/08/jmeter-vs-gatling/#gist23106684, 2015. [Online; consultada 14-September-2017].

[12] Okta, Inc. Introduction to json web tokens. https://jwt.io/introduction, 2019. [Online; consultada 14-Nov-2022].

[13] Okta, Inc. Get started with json web tokens. https://auth0.com/learn/json-web-tokens/, 2022. [Online; consultada 14-Nov-2022].

[14] WordPress. Wordpress optimization. https://codex.wordpress.org/WordPress_Optimization, 2017. [Online; consultada 14-September-2017].