

```
//fires the appear event when appropriate
var check = function() {
    //is the element hidden?
    if (!t.is(':visible')) {
        //it became hidden
        t.appeared = false;
        return;
    }

    //is the element inside the visible window?
    var a = w.scrollLeft();
    var b = w.scrollTop();
    var o = t.offset();
    var x = o.left;
    var y = o.top;

    var ax = settings.accX;
    var ay = settings.accY;
    var th = t.height();
    var wh = w.height();
    var tw = t.width();
    var ww = w.width();

    if (y + th + ay >= b &&
        y <= b + wh + ay &&
        x + tw + ax >= a &&
        x <= a + ww + ax) {
        //trigger the event
        if (t.appeared) {
            t.trigger('appear');
        }
    }
}
```

UNIDAD FORMATIVA 5

Servidores

Sistemas operativos

Índice

Administración de sistemas	2
Objetivos	2
Definición de sistema informático	3
El rol del administrador de sistemas	4
Servidores	7
Elección de sistema operativo	9
Preparación entorno de pruebas	11
Administración de sistemas Linux	15
Administración de usuarios y grupos	15
Sistemas de archivos	20
Sistemas de permisos	23
Procesos y servicios	26
Herramientas del sistema	30
Programación de tareas	32

Administración de sistemas

Para continuar el curso, nos vamos a detener a estudiar la disciplina de la administración de sistemas informáticos, entendiendo como tales aquellas instancias o máquinas físicas que se encargan de alojar todo el software que da soporte a las capacidades de negocio de una organización:

- Bases de datos.
- Servidores web.
- Sistemas de logs, monitorización.
- Aplicaciones críticas.

Estas capacidades son, por definición, **críticas** para el negocio y como tal deben ser gestionadas con la mayor eficiencia, rigor y seguridad posibles. La figura que tradicionalmente se ocupa de la gestión de los sistemas IT en una organización, el *sysadmin*, tiene una relevancia total para el correcto desempeño de toda esa flota de servidores. Estudiaremos las tareas a las que tradicionalmente ha estado asociado este administrador, así como los retos que han ido apareciendo en los últimos años, motivados por la explosión de las tecnologías cloud y la aparición del movimiento y la cultura DevOps.

Además, se introducirán el concepto de **servidores**, los sistemas que como DevOps, a cargo de los sistemas físicos o virtuales, deberemos mantener funcionales, seguros y dando servicio el mayor tiempo posible. De especial interés es aprender más sobre el sistema operativo que nos encontraremos de manera más predominante en la industria: Linux. Nos detendremos a revisar los conceptos básicos de Linux y unos cuantos comandos que nos serán de utilidad para verificar el estado de procesos y servicios, así como para la operativa habitual de gestión del sistema.

Objetivos

- Definir el concepto de sistema informático.
- Aprender sobre los retos a los que se enfrenta un *sysadmin* en la era actual.
- Definir el concepto de servidor y preparar un entorno local de pruebas válido para la asignatura y el resto del curso.
- Estudiar los conceptos básicos de Linux.

Definición de sistema informático

Aunque existen muchas diferencias en los servicios que dan soporte a una organización, podemos trazar una línea y decir que un **sistema informático** consiste en uno o más servidores, desplegados en una o más redes, que ejecutan diferentes piezas de software con una configuración determinada para conseguir el objetivo que se propone.

En este caso, la administración de sistemas es la disciplina dentro de las tecnologías de la información que se encarga de **diseñar, configurar y mantener** esos sistemas informáticos, lo que implica responsabilidad en cada una de las piezas que hemos comentado:

- Elección de hardware, tanto físico como virtual, o recursos cloud.
- Diseño de las **redes** sobre las que correrá ese hardware, definiendo con cuidado los límites de esa red, dónde se conectará al exterior, subredes internas, etc.
- Instalación, configuración y mantenimiento del **sistema operativo** y del **software adicional** necesario para poder productivizar las aplicaciones que sean necesarias para cumplir los objetivos.
 - Esto implica **monitorizar** que el hardware y los servicios estén funcionando adecuadamente y establecer **alarmas** que nos avisen cuándo se cruzan umbrales peligrosos, como poco espacio en disco o uso excesivo de recursos de red.
- Tomar decisiones de **seguridad** para cada pieza, de modo que se pueda garantizar que el sistema se mantiene aislado de intrusiones no autorizadas, evita fugas de datos, se mantiene funcionando ante ataques externos o circunstancias internas, casuales o intencionadas.
- En general, se debe garantizar que los servicios tienen una **disponibilidad mínima** aceptable, usualmente formalizada a través de los **acuerdos de nivel de servicio** (*service level agreement, SLA*). Recordemos el concepto de los **nueves** que se introdujo en el tema de 'Cloud > S3'.
 - Objetivo 'Tres 9s' → El sistema está disponible el 99.9% del tiempo.
 - Cinco 9s → 99.999%...
 - Cada 9 que se añade a la disponibilidad se traduce en un mejor servicio de cara al cliente, pero con una cantidad de trabajo detrás que no sigue una progresión, en absoluto, lineal: <https://uptime.is>.

El rol del administrador de sistemas

Debido a la aparición de la computación en la nube, el trabajo de un administrador de sistemas (conocido habitualmente por el nombre en inglés, **sysadmin**) está cambiando y derivando, en muchos casos, en la reconversión hacia DevOps. A medida que crece la adopción de la nube en las organizaciones, se pasa de la administración de recursos **físicos** a la de sistemas virtuales o desacoplados y del mantenimiento de las máquinas a ser capaz de replicar los entornos para garantizar el servicio.

Las empresas buscarán administradores de sistemas y DevOps con la capacidad de liderar la transformación y el despliegue de entornos en la nube.

Tareas tradicionales

Cada organización define los roles y las responsabilidades de sus administradores. El ámbito de las tareas cambia de organización a organización y también cambia según evoluciona la organización, pero hay algunas tareas que cualquier administrador tiene:

- Instalación de servidores y clientes.
- Instalación y mantenimiento de aplicaciones.
- Creación de usuarios y grupos.
- Soporte a usuarios.
- Copias de seguridad y recuperación frente a desastres.
- Seguridad.
- Automatización de tareas.
- Instalación de periféricos como impresoras.
- Gestión de cambios.
- Configuración de equipos de red local.

Retos en la era DevOps

La función del administrador del sistema está cambiando. Hace solo unos años, la mayoría de los sistemas se ocupaban de granjas de servidores de hardware físico y realizaban una planificación detallada de la capacidad. Ampliar su aplicación significaba comprar nuevo hardware y tal vez pasar tiempo acumulándolo en el centro de datos.

Actualmente, hay un gran porcentaje de la industria que nunca ha tocado el hardware físico. Es posible desplegar servidores con una llamada API o haciendo clic en un botón en una página web.

A medida que aumenta la competencia en el espacio del mercado de la nube, su atractivo para los administradores de sistemas y los propietarios de negocios aumenta casi a diario. AWS continúa impulsando el mercado de la computación en la nube al introducir con frecuencia nuevas herramientas y servicios (no hay más que repasar la velocidad con la que publican las novedades en [What 's New?](#)).

Las economías de escala están constantemente bajando el precio de los servicios en la nube. Aunque los entornos como AWS o Google Compute Engine aún no son adecuados para todas las aplicaciones, cada vez es más claro que las habilidades en la nube se están convirtiendo en una parte necesaria de un conjunto de herramientas completo del administrador de sistemas.

Para las empresas, la nube abre nuevas vías de flexibilidad. Los equipos tecnológicos pueden hacer cosas que hubieran sido prohibitivamente caras hace solo unos años. Los juegos y las aplicaciones que tienen la suerte de convertirse en éxitos desbocados, a menudo, requieren una gran cantidad de capacidad de cómputo. Provisionar esta capacidad en horas en lugar de semanas permite a estas compañías responder rápidamente al éxito, sin entrar en inversiones y compromisos de gasto por varios años o gastos iniciales de capital.

En la era DevOps, los desarrolladores y la dirección saben lo importante que es iterar y mejorar rápidamente el código de aplicación. Los servicios de los proveedores de nube permiten tratar la infraestructura de la misma manera, permitiendo que un equipo relativamente pequeño administre infraestructuras de aplicaciones masivamente escalables.

A continuación, se muestran algunos de los retos que un *sysadmin* puede tener que asumir en un era en donde la tecnología y la cultura DevOps se está extendiendo y donde los servicios en la nube son la norma en lugar de la excepción:

Migración a la nube

Uno de los primeros retos de un administrador será la migración de cargas de trabajo de infraestructuras físicas y virtualizadas de un centro de datos privado (puede ser en la misma organización o en un DC físico) a un proveedor de nube pública. ¿Tiene sentido reconstruir todo el sistema desde cero? ¿Es posible crear una imagen de sistema existente que ofrece el 80% más de la funcionalidad necesaria?

La solución más sencilla puede ser hacer una copia del sistema, copiarla al proveedor de la nube y arrancarla en la nueva ubicación. Esta opción permite, entre otras cosas, delegar la administración de la infraestructura física y virtual en el proveedor y mantener el funcionamiento familiar al que están acostumbrados los administradores. No obstante, esta opción no permitirá gestionar el ciclo de vida de las aplicaciones existentes.

En un entorno DevOps en el que se tienda a la automatización se debería apostar por automatizar el despliegue, al menos, de todas las aplicaciones nuevas y en lo posible de las existentes, para poder disfrutar de todas las ventajas de un entorno de nube.

Creación de entornos híbridos

La integración con los entornos existentes es casi inevitable cuando se emprende la migración de entornos tradicionales a la nube. Esto requiere el aprovisionamiento de recursos, tanto locales como de la nube, con protocolos compatibles. Un DevOps tendrá que combinar sus conocimientos de desarrollo y administración con conocimientos de red para establecer las conexiones adecuadas. Estos recursos de red serán en muchos casos virtuales y, por lo tanto, también deben ser automatizados.

Aprovisionamiento de recursos

La implantación del *cloud computing* permite finalmente que los DevOps dispongan de recursos renovables, reciclables y fáciles de proveer. Crear un nuevo servidor es tan simple como unos pocos clics en una consola web. El reemplazo de un servidor existente es igual de sencillo si está automatizado su despliegue y configuración. Además de la facilidad, el aprovisionamiento es mucho más rápido en la nube que en los entornos tradicionales. Si la carga de un servidor es muy alta, es suficiente con lanzar más instancias o, en el peor de los casos, lanzar un servidor nuevo de mayor tamaño, utilizando las automatizaciones.

Administrar un entorno en la nube significa más que mantener todos los sistemas funcionando. Los DevOps deben ahora vigilar el coste de funcionamiento del entorno. Los costes de hardware se han reemplazado con el pago por uso, a veces, pero no siempre, con el coste de licencia incluido.

Los contratos de soporte se han trasladado de los proveedores de hardware a los proveedores de infraestructura. La comprensión de este equilibrio ayudará a los DevOps a trabajar con otros para construir una solución rentable y confiable.

Seguridad del sistema

La seguridad es una preocupación importante en la nube. Los proveedores son responsables de asegurar el acceso físico al hardware. Sin embargo, la infraestructura debe ser accedida de forma remota por definición. Los DevOps deben estar familiarizados con el modelo de seguridad de su proveedor de nube.

Los roles, grupos, usuarios y políticas son mecanismos comunes para otorgar y restringir el acceso. La mayoría de los proveedores de nube incorporan estas posibilidades de granularidad entre sus herramientas de administración. Por ejemplo, una directiva puede conceder permiso para que un usuario cree una instancia en una región y la deniegue en otra. Al igual que los entornos tradicionales, la seguridad debe ser una consideración inicial.

Permitir o bloquear el acceso a la red es un trabajo tradicionalmente de los administradores de red. En la nube es habitual que este trabajo recaiga en los DevOps, que aprovecharán la automatización para reducir los riesgos de seguridad de red.

Garantizar la disponibilidad

Uno de los objetivos de contar con recursos en la nube será el que tengamos una disponibilidad (*uptime*) cercana al 100%. Esto no es algo que pueda lograrse de forma automática por el mero hecho de utilizar la nube. Si bien es cierto que se podría pensar que al usar la nube no habrá caídas en los sistemas por fallos hardware, no es menos cierto que las máquinas en la nube a veces necesitan ser movidas para tareas de mantenimiento o incluso apagadas por parte del proveedor.

En este escenario, muy cercano a la realidad, la única forma de garantizar el *uptime* es contar con mecanismos que permitan desplegar flotas de servidores autogestionados.

El concepto de flota se puede concretar en un grupo de autoescalado en AWS o en un despliegue en Kubernetes, por citar algunos. Los objetivos detrás de un grupo de servidores autogestionados son:

- Si un servidor deja de funcionar debe ser posible apagarlo sin intervención humana.
- Si la flota actual no tiene capacidad suficiente, debe ser posible añadir un servidor nuevo sin intervención humana.
- Una actualización incremental no debe implicar una caída del sistema.

Esta idea de flota en la que los servidores aparecen y desaparecen sin intervención humana queda reflejada en el paradigma de las mascotas y el ganado ([pets vs cattle](#), lectura muy recomendable):

- Los **servidores tradicionales** eran tratados como **mascotas**: provisionar uno llevaba mucho tiempo y los administradores hacían lo posible para mantenerlo funcionando todo el tiempo posible.
- Los **servidores modernos** en la nube se parecen más a **ganado**: es posible tener tantos que no merece la pena preocuparse por cada uno en particular.

Un detalle muy importante: una disponibilidad del 100% no significa que los DevOps trabajan 24/7 para garantizarla. La **automatización** es la base sobre la que se construyen las aplicaciones de escalado y de despliegue.

Servidores

En un entorno tradicional, un **servidor** podría referirse tanto al equipo informático físico formado por CPU, memoria, discos e interfaces de red, como al proceso de software que ofrece una funcionalidad a un cliente a través de la red. En esta asignatura, el concepto de servidor se limita al de una instalación de un sistema operativo pensado para ofrecer ejecutar aplicaciones para clientes.

Los sistemas operativos usados como servidores no son muy diferentes de los que ejecutan los ordenadores de sobremesa o los portátiles. Ambos tipos tienen un núcleo, una arquitectura de procesador habitualmente compatible (por ejemplo, x86), controladores de hardware y utilidades de software. Muchos de los ejemplos descritos en los siguientes temas pueden ejecutarse también en un entorno de escritorio sin cambio alguno.

Sin embargo, un servidor tiene una característica diferencial con los equipos habituales de sobremesa y es que suelen estar aislados, en ubicaciones protegidas, apilados en *racks* y con condiciones especiales de refrigeración y organización que las hacen idóneas para el almacenamiento masivo de máquinas.

Evidentemente esto depende de la organización, ya que una empresa pequeña puede tener unos pocos servidores en una sala o puede decidir contratar servicios en un *data center* (DC) cercano. También puede estar totalmente migrada a la nube, donde los recursos subyacentes siguen siendo físicos y desplegados en alguna parte.

Imprescindible leer [el caso de la infraestructura global de AWS](#) para hacernos una idea de cómo funciona un proveedor global de servicios cloud.

Evidentemente, si un servidor está en una ubicación tan restringida, acceder mediante un teclado, un ratón y un monitor no parece muy cómodo. De hecho, **es el caso más extremo**, solo cuando no hay ninguna otra posibilidad de arreglar un problema y solo en aquellos casos en los que la infraestructura está tan acoplada al sistema que no salga más barato disponibilizar un servidor nuevo o que esto no sea una opción. Veremos que estos casos representan los **antipatrones** de los que queremos huir como administradores.

En general, el acceso es mediante acceso remoto a través de la red. Un servidor sin conexión a la red se considera una pieza inútil pues no es posible interactuar con ella.

Acceso remoto

Las maneras de conseguir acceso remoto más comunes son:

- Telnet

Telnet es un protocolo de capa de aplicación que proporciona acceso remoto mediante una conexión de terminal virtual. La utilización de Telnet está **desaconsejada** porque el tráfico no se transmite cifrado. Esto permitiría a un usuario malintencionado acceder al contenido del tráfico si fuera capaz de interceptarlo. Para poder utilizar Telnet es necesario que el servidor lo tenga activado y contar con un cliente de Telnet, que habitualmente está incluido en todos los sistemas operativos, aunque muchas veces desactivado por defecto.

- SSH

Secure shell (SSH) es un protocolo de red de cifrado para servicios de red de operación segura a través de una red no segura. Su aplicación más habitual es el de inicio de sesión remoto a servidores Linux. SSH proporciona un canal seguro mediante una arquitectura cliente-servidor. Los sistemas operativos Linux cuentan habitualmente con un cliente de SSH. En el caso de los clientes Windows, el cliente más utilizado es la aplicación puTTY. Se trata de un cliente de SSH de código abierto y libre (<http://www.putty.org/>).

SSH es la forma más habitual de acceder a los servidores, tanto aquellos alojados en la nube como servidores tradicionales en un centro de datos propio. SSH soporta el acceso con usuario/contraseña y el uso de claves públicas (opción por defecto en AWS, como vimos).

- VNC

Virtual network computing (VNC) es un sistema de control remoto que utiliza el protocolo *remote frame buffer* (RFB) para controlar de forma remota otro ordenador. Transmite los eventos de teclado y ratón de un ordenador a otro y las actualizaciones de pantalla de vuelta en la otra dirección. VNC es independiente de la plataforma, por lo que puede usarse para conectarse también a equipos Windows. El código fuente VNC original y muchos derivados modernos son de código abierto bajo la licencia pública general de GNU.

En el caso de VNC y otros métodos de acceso remoto mediante terminal gráfico, la principal limitación que tenemos es la imposibilidad de automatizar tareas o posibles limitaciones para asumir permisos elevados.

La **línea de comandos**, esa pantalla en negro con texto que tan bien han vendido las películas sobre hackers, es el entorno habitual de trabajo de un administrador. Incluso Windows, que mantiene el concepto de interfaz gráfica en su nombre, introdujo powershell en 2007.

La línea de comandos ofrece la posibilidad de escribir y ejecutar scripts: archivos con comandos y funciones propias de shell o powershell que se pueden ejecutar en bloque y de este modo facilitan la automatización y el mantenimiento de los sistemas.

Importante:

A partir de este momento, no existen las interfaces gráficas. El acceso y manejo de los sistemas será siempre mediante la línea de comandos, a la que accederemos conectándonos vía SSH.

Elección de sistema operativo

La discusión entre qué sistema operativo es mejor ha alimentado multitud de foros, blogs y artículos desde hace años. Los argumentos giran en torno al coste, la libertad de acceso al código fuente, la seguridad, la usabilidad, etc. Un argumento a favor en una situación (la usabilidad de Windows en un entorno de escritorio) puede no serlo en otra (en un entorno de servidor, la usabilidad del usuario puede pasar a un segundo plano).

La elección de uno u otro debe estar marcada por las necesidades de la aplicación, la organización y los usuarios. Por ejemplo, Linux es una gran opción para una aplicación nueva, diseñada con una arquitectura nativa en la nube, gracias al soporte nativo de contenedores. En el caso de una empresa con una gran base de usuarios que necesitan una suite ofimática y acceso a unas pocas herramientas corporativas, Windows parte con la ventaja de facilitar la administración de políticas de seguridad de manera centralizada.

En ambos ejemplos se pueden dar razones a favor y en contra de uno y otro. Cuantas más herramientas domine un administrador, más preparado estará para poder evaluar dichas razones y tomar una decisión.

En esta asignatura **nos centraremos en el sistema operativo Linux**, por ser el de mayor presencia en el mundo DevOps:

- La mayoría de las imágenes empleadas por máquinas virtuales son de alguna distribución Linux, o al menos aquellas cuyo uso no supone ningún coste.
- Para la creación de imágenes Docker, las imágenes base están basadas en Linux en la gran mayoría de los casos.
- El software empleado en la mayoría de las dimensiones de la ingeniería software está disponible para sistemas Linux por defecto y solo en algunos casos se crean las versiones para Windows.

Esto no quiere decir que los sistemas Windows no se usen o que no valga la pena aprender a administrarlos: **nada más lejos de la realidad.**

Distribuciones Windows

El ecosistema Windows no es tan extenso como el de las distribuciones de Linux, pero también puede dar lugar a confusión. La versión de sistema operativo dedicada a servidores se llama **Windows Server**.

Dentro de la oferta de Windows Server, Microsoft publica versiones de actualización a menudo. De manera similar a otros sistemas operativos, estas versiones son soportadas durante unos pocos años. Es decir, Microsoft publicará parches de funcionalidad y seguridad durante el periodo de soporte, además de ofrecer asistencia técnica a sus clientes. Por ejemplo, el [ciclo de soporte de Windows Server 2019](#) empezó el 15/10/2019 y terminará el 12/1/2027. Este tema se centrará en Windows Server 2019, la versión más reciente, salvo cuando se indique lo contrario.

Windows Server 2019 está disponible en dos ediciones, también conocidas como SKU: Standard y Datacenter. Las ediciones se diferencian en su precio y en las [funcionalidades que soportan](#).

La edición Datacenter está enfocada principalmente a entornos de virtualización y puede alojar un número ilimitado de máquinas virtuales (siempre que el servidor sea capaz de servir los requisitos de hardware virtual), mientras que la edición Standard solo puede ejecutar dos máquinas. Además de algunas diferencias específicas de Hyper-V (el hipervisor de virtualización de Windows), Datacenter soporta *software defined networking* ([SDN](#), redes definidas por software), una funcionalidad esencial de cualquier entorno de nube privada o pública.

Es habitual que Microsoft distribuya las ediciones en una misma imagen ISO, por lo que la elección de edición puede hacerse durante la instalación.

Ambas ediciones soportan, además, varios modos de ejecución:

- Escritorio o Server with Desktop Experience.
- Server Core.
- Nano Server.

El modo de **escritorio** es el más familiar, ya que convierte al servidor en un host similar a cualquier máquina Windows: tiene botón de inicio, ventanas y consolas de configuración. El modo Core, sin embargo, no tiene escritorio. Mantiene las funcionalidades para soportar aplicaciones tradicionales y permite la activación de cualquier rol, pero no tiene el entorno de escritorio tradicional. Está pensado para ser administrado remotamente a través de la línea de comandos, con powershell o con consolas MMC remotas. Hay otras diferencias, como que Server Core no incluye las herramientas de accesibilidad, soporte de audio o Internet Explorer.

Distribuciones Linux

En la grandísima mayoría de los casos, el sistema operativo sobre el que se ejecuten los procesos será algún tipo de sistema Linux. La razón principal es que Linux es un sistema operativo libre, lo que significa que no hay que pagar licencia por su uso: esto libera a las organizaciones de un problema de costes, aunque tiene sus contrapartidas.

¿Qué problemas puede traer el usar un sistema operativo libre para ejecutar nuestras operaciones? El **soporte técnico**, o lo que es lo mismo: ¿existe alguien que se haga responsable de solucionarme los problemas derivados del uso del sistema operativo? Existen empresas que ofrecen, a través de un sistema de licencias, distribuciones con diferentes opciones de soporte técnico, como pueden ser Red Hat o SUSE.

Una distribución de Linux consiste del núcleo (*kernel*) del sistema operativo junto a una colección de paquetes y aplicaciones. Dada una misma versión del núcleo, una aplicación escrita para Linux debería ser ejecutable en cualquier distribución con ese núcleo. No obstante, dado que la colección de aplicaciones abarca desde el instalador al entorno gráfico y el gestor de paquetes, la manera de interactuar con cada distribución puede ser muy diferente.

Algunas distribuciones están enfocadas al usuario final. A tal efecto incluyen entorno gráfico, aplicaciones ofimáticas, reproductores de audio y vídeo, etc. Esta asignatura se centrará en distribuciones orientadas a **servidores**. Algunas de ellas, como Ubuntu, mantienen ediciones de escritorio y de servidor para una misma versión de la distribución.

Entre las aplicaciones incluidas en casi todas las distribuciones están las utilidades GNU. Hay utilidades de manipulación de ficheros y del propio contenido de los ficheros, de información del sistema operativo, de la sesión, de búsqueda, editores de texto, etc. Estas utilidades son esenciales a la hora de administrar servidores en la línea de comandos y, como se verá en los temas siguientes, para automatizar tareas.

Preparación entorno de pruebas

Vamos a ofrecer dos alternativas para crear una máquina Ubuntu, que nos servirán tanto para probar cualquier comando, técnica o script que se comente en esta lección (y cualquier otra), como para las actividades de desarrollo: una VM local o una instancia EC2.

No existe diferencia operativa entre ambas soluciones, es cuestión de gustos cuál usar. A tener en cuenta que el caso de AWS puede incurrir en costes si no se es precavido, pero si no tenemos un ordenador potente o preferimos compartir una instancia entre varios miembros de un equipo es la mejor opción.

VM local con Vagrant y VirtualBox

La opción más común, barata (por gratis) y directa es levantar una máquina virtual en nuestro ordenador personal, usando alguna de las opciones de software introducidas en la lección correspondiente: en este caso vamos a usar [VirtualBox](#).

A pesar de que existe la posibilidad de lanzar directamente una ISO de Ubuntu en VirtualBox, vamos a optar por un enfoque más apropiado para el curso que estamos dando: usaremos una tecnología que nos permite **describir** qué sistema operativo queremos lanzar, cuántas instancias y con qué características e incluso qué script de aprovisionamiento lanzar cuando esta se cree por primera vez.

El software que implementa esta especie de 'infraestructure-as-code' se llama [Vagrant by HashiCorp](#) y está disponible para los principales sistemas operativos. Una vez que lo tenemos instalado, crearemos un fichero llamado 'Vagrantfile', escrito en el lenguaje de programación Ruby, con el siguiente contenido:

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/focal64"
  config.vm.network "private_network", ip: "192.168.33.10"
  config.vm.provider "virtualbox" do |vb|
    vb.memory = "1024"
  end
end
```

Con este fichero estamos diciéndole a Vagrant que queremos levantar una máquina virtual usando la **box** 'ubuntu/focal64', correspondiente a Ubuntu 20.04 LTS. Una box de Vagrant es una adaptación de un sistema operativo objetivo, modificada para su integración con la herramienta. Como esto no pretende ser más que una leve introducción a la herramienta, se recomienda [leer más sobre las boxes](#) y sobre [Vagrant en general](#) en su web oficial.

Para lanzar la máquina virtual se usará el comando 'vagrant up', que lo primero que hará será descargarse la box, para después configurar SSH y dejar la VM lista. Podemos conectarnos usando la clave SSH que se nos acaba de crear o usar un atajo con el comando 'vagrant ssh':

```
$ vagrant up
==> vagrant: A new version of Vagrant is available: 2.2.19 (installed version: 2.2.16)!
==> vagrant: To upgrade visit: https://www.vagrantup.com/downloads.html

Bringing machine 'default' up with 'virtualbox' provider...
==> default: Box 'ubuntu/focal64' could not be found. Attempting to find and install...
    default: Box Provider: virtualbox
    default: Box Version: >= 0
==> default: Loading metadata for box 'ubuntu/focal64'
    default: URL: https://vagrantcloud.com/ubuntu/focal64
==> default: Adding box 'ubuntu/focal64' (v20211026.0.0) for provider: virtualbox
    default: Downloading:
https://vagrantcloud.com/ubuntu/boxes/focal64/versions/20211026.0.0/providers/virtualbox.box
Download redirected to host: cloud-images.ubuntu.com
==> default: Successfully added box 'ubuntu/focal64' (v20211026.0.0) for 'virtualbox'!
...
==> default: Machine booted and ready!
==> default: Checking for guest additions in VM...
==> default: Configuring and enabling network interfaces...
==> default: Mounting shared folders...

$ vagrant status
Current machine states:
```

default running (virtualbox)

The VM is running. To stop this VM, you can run `vagrant halt` to shut it down forcefully, or you can run `vagrant suspend` to simply suspend the virtual machine. In either case, to restart it again, simply run `vagrant up`

```
$ ssh -i .vagrant/machines/default/virtualbox/private_key The authenticity of host '192.168.33.10 (192.168.33.10)' can't be established.
```

```
ECDSA key fingerprint is SHA256:Qdb6UFvVN0mjMK/T/zaEhcsrARnn5kHwW/MYbimm9co.
```

```
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
```

```
Warning: Permanently added '192.168.33.10' (ECDSA) to the list of known hosts.
```

```
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-91-generic x86_64)
```

```
* Documentation: https://help.ubuntu.com
```

```
* Management: https://landscape.canonical.com
```

```
* Support: https://ubuntu.com/advantage
```

```
System information as of Sun Dec 19 23:18:55 UTC 2021
```

```
System load: 0.0          Processes:      114
Usage of /:  3.3% of 38.71GB Users logged in:    0
Memory usage: 20%         IPv4 address for enp0s3: 10.0.2.15
Swap usage:  0%          IPv4 address for enp0s8: 192.168.33.10
```

```
1 update can be applied immediately.
```

```
To see these additional updates run: apt list --upgradable
```

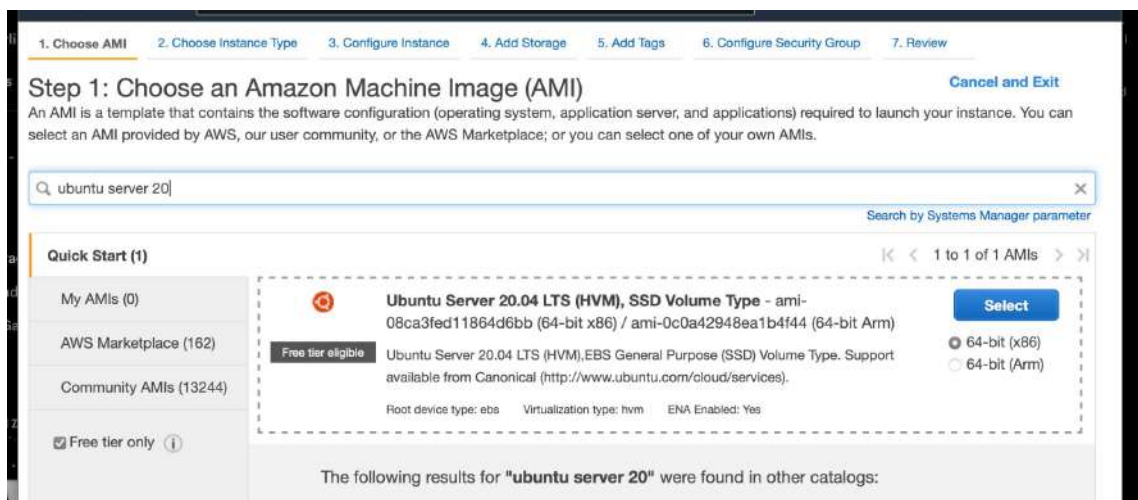
```
Last login: Sun Dec 19 23:17:38 2021 from 10.0.2.2
```

```
vagrant@ubuntu-focal:~$
```

Instancia EC2 de AWS

Si hemos seguido las lecciones en orden, ya conoceremos el servicio EC2 de AWS perfectamente. Si además hemos optado por crear una cuenta AWS, durante el primer año, tenemos la opción de lanzar máquinas en EC2 de **manera totalmente gratuita**, aunque de potencia reducida son más que suficientes para el propósito de pruebas.

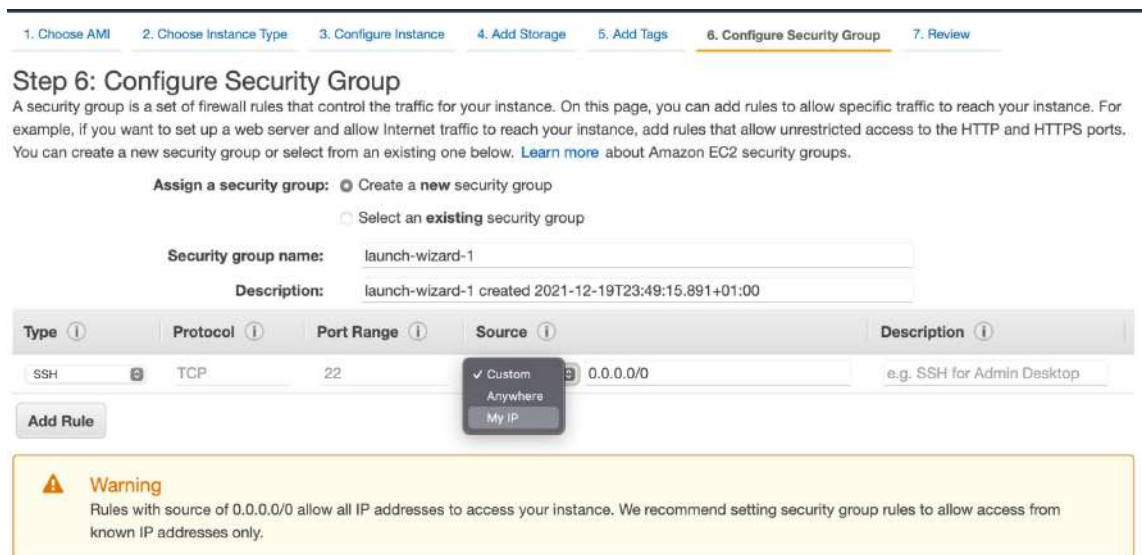
De lecciones pasadas disponemos de plantillas de CloudFormation o Terraform que se encargarán de lanzar una instancia por nosotros, que podremos después destruir a voluntad cuando no la necesitemos. Si preferimos crear una máquina usando la UI, y explorar las opciones disponibles. deberemos ir al [lanzador de instancias de EC2](#), marcar 'Free tier only' y buscar Ubuntu Server 20:



Se recomienda aceptar todas las opciones por defecto, en especial aquellas marcadas como ‘free tier’ pues nos garantizarán que mantenemos la instancia libre de coste. Recordemos que recursos que se creen al lanzar una instancia, como IPs Elásticas o volúmenes lógicos de almacenamiento, pueden repercutir en costes si no se eliminan a la vez que la instancia.

Como recomendaciones, para este caso particular, y como buenas prácticas, en general:

1. Usar un **Security Group** que permita acceso SSH (o de los servicios y protocolos que vayamos a necesitar) solo desde la IP de nuestra casa/oficina.



Se pueden dar de alta casi tantas entradas como IPs necesitemos y, en caso de quedarnos sin sitio, podemos crear más SGs y asociarlos.

2. Añadamos **al menos una tag** con un nombre significativo. Esto nos permite encontrar rápidamente nuestros recursos a medida que vayamos creando más y más elementos cloud y facilita la identificación de nuestra instancia mejor que el ID aleatorio que le asignará Amazon.

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 5: Add Tags

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver.
A copy of a tag can be applied to volumes, instances or both.
Tags will be applied to all instances and volumes. [Learn more](#) about tagging your Amazon EC2 resources.

Key (128 characters maximum)	Value (256 characters maximum)	Instances	Volumes	Network Interfaces
Name	instancia-pruebas-edix	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

[Add another tag](#) (Up to 50 tags maximum)

3. Si no tenemos creado un par de claves SSH, se nos pedirá que creamos una ahora. **No debemos reusar en exceso**, ya que si se filtraran unas credenciales usadas por todas las máquinas EC2 de una cuenta se comprometerían todas aquellas que estén levantadas en ese momento (una de las formas por las que hackers de todo el mundo rastrean repositorios públicos en busca de credenciales desprotegidas con las que poder entrar en AWS y minar bitcoins hasta que el usuario o AWS toman cartas en el asunto).

Administración de sistemas Linux

Existen muchas similitudes entre las muchas familias de distribuciones (UNIX, Debian, RedHat, Suse, Arch...) y de sus muchas variedades (Ubuntu, Manjaro, Centos...), incluso las diferentes versiones de MacOS X. Si bien es cierto que existen diferencias fundamentales (gestores de paquetes, comunidad de mantenedores, open source o no...), estudiaremos esas características comunes, aunque por simplicidad nos referiremos siempre a Ubuntu.

Importante:

En esta sección se introducirán multitud de comandos de Linux, en muchas ocasiones de forma superficial o cubriendo solo un pequeño subconjunto de funcionalidades. Para evitar quedarse con dudas de cómo funciona realmente, es imprescindible conocer y emplear el comando 'man', que muestra ayuda sobre los comandos del sistema. [Existe una versión online](#) de las páginas de ayuda, pero se recomienda acostumbrarse a hacerlo usando el propio comando.

Administración de usuarios y grupos

Linux es un sistema operativo multiusuario: Esto significa que permite el inicio de sesión de múltiples usuarios simultáneamente, ya sea por la línea de comandos o en una sesión de escritorio. También hay usuarios específicos para ciertos componentes del sistema operativo. Por ejemplo, el servidor web **nginx** suele ejecutarse bajo el usuario 'nginx' o el usuario 'www-data', según cómo se haya instalado el paquete.

Linux también tiene el concepto de **grupo**. Los usuarios pueden pertenecer a uno o más grupos y suelen agregarse a grupos para dar permisos de acceso a ciertos recursos. Por ejemplo, una carpeta compartida ofrecida en un servidor FTP puede dar acceso a un grupo concreto. Los miembros de ese grupo podrán leer los ficheros, facilitando la labor administrativa.

Por regla general, al crear un usuario se crea una carpeta homónima debajo de la ruta `/home`. Esta carpeta ofrece un lugar en el que los usuarios pueden guardar sus ficheros, además de ser la ubicación por defecto que muchas aplicaciones usan para guardar las configuraciones específicas de cada usuario. Además, existe un **atajo** por el que podemos referirnos a nuestra propia carpeta de usuario con el símbolo `~`.

Por ejemplo, las claves para las conexiones SSH se guardan en la carpeta `~/.ssh` y el historial de bash se guarda en `~/.bash_profile`.

sudo

El comando **sudo** permite la ejecución de comandos administrativos a usuarios distintos de root. Las ventajas de sudo son:

- Incrementa la seguridad.
- Permite más granularidad en el control de comandos administrativos.
- Incorpora auditoría de ejecución.
- Algunas distribuciones deshabilitan el usuario root, por lo que sudo es la única opción.

Los comandos de creación y modificación de usuarios son ejemplos de comandos administrativos que **solo el usuario root puede ejecutar**. Las distribuciones que deshabilitan el usuario root habilitan el comando sudo para todos los comandos para un usuario normal: Ubuntu lo habilita para el usuario que se crea durante la instalación y en Amazon, Linux es el usuario `ec2-user` el que tiene permisos.

Para modificar permisos de sudo hay que editar el fichero `/etc/sudoers` con el comando especial `visudo`. Este comando también necesita privilegios, así que habría que ejecutarlo con `sudo visudo`. Se abrirá el editor del sistema y se comprobará la sintaxis del fichero. Un fichero `/etc/sudoers` erróneo impedirá que se puedan ejecutar comandos administrativos y, por tanto, ni siquiera se podrá ejecutar `sudo visudo` para arreglar el fichero.

Según el contenido de `/etc/sudoers` de la imagen 1, hay dos formas de dar permisos de sudo a un usuario:

- Añadiendo el usuario individualmente con una línea nueva, por ejemplo:

`ubuntu ALL=(ALL:ALL) ALL.`
- Añadir el usuario como miembro de un grupo con permisos, como los grupos `admin` o `sudo`.

La sintaxis del fichero se puede consultar con `'man sudoers'`.

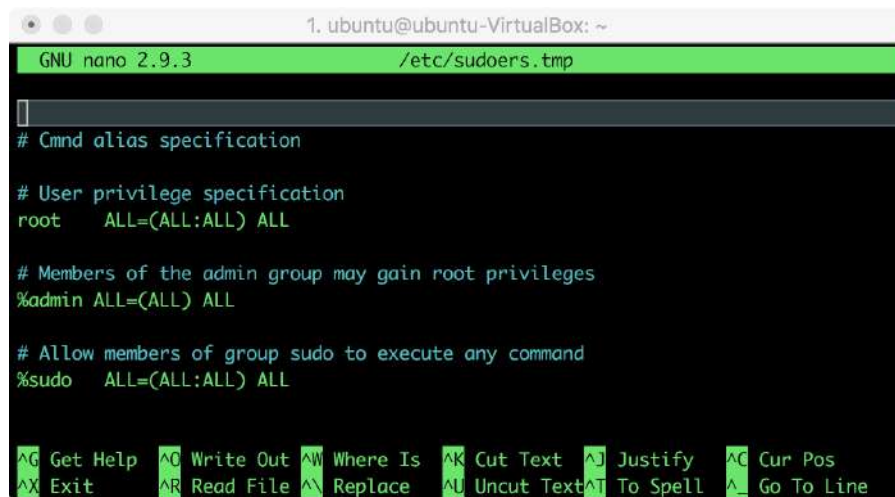


Imagen 1. Editor nano tras ejecutar sudo visudo.

Administración de usuarios

El comando 'useradd' permite crear usuarios nuevos. Durante la creación se especifican el nombre, la descripción (con el flag -c), la shell por defecto del usuario (con -s) y si se desea crear la home del usuario (con -m). El usuario estará desactivado y sin contraseña, que habrá que añadir con el comando 'passwd'. La imagen 2 muestra la creación de un usuario nuevo, la fijación de la contraseña y el inicio de sesión con este nuevo usuario.

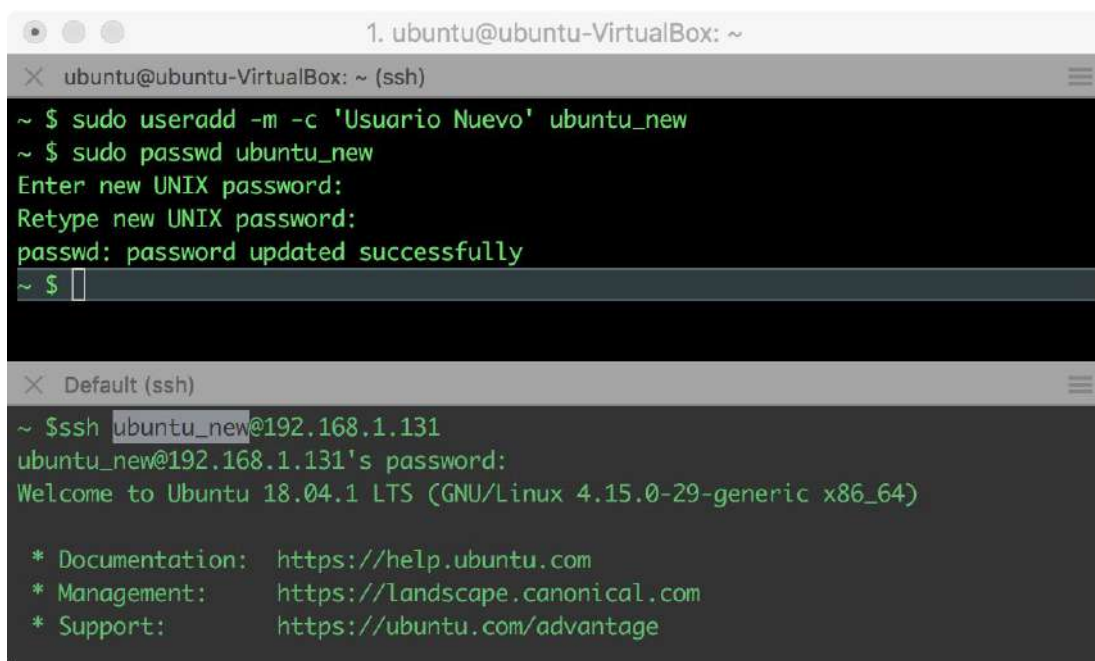
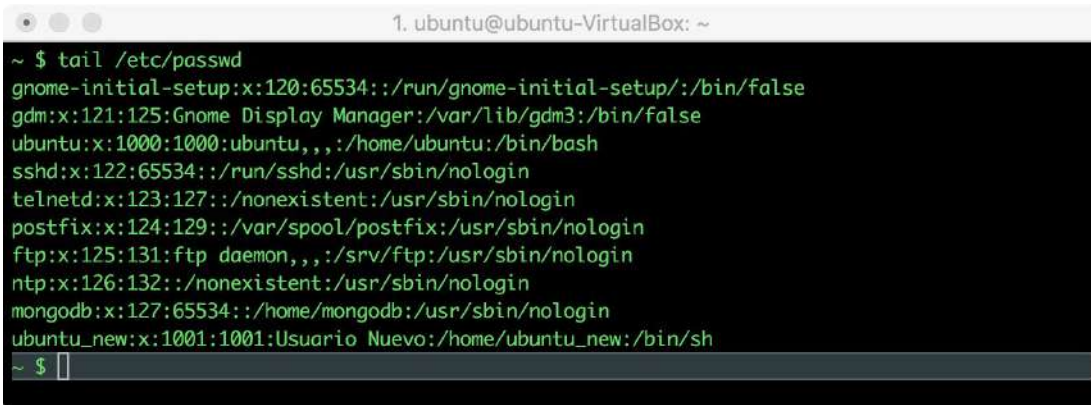


Imagen 2. Creación de usuario con 'useradd'.

La información del nuevo usuario se almacena en dos ficheros, '/etc/passwd' y '/etc/shadow'. El primero contiene detalles del usuario: nombre de usuario, id, id de grupo, ruta de la *home* y ruta de la shell por defecto, entre otros.

Los IDs son identificadores numéricos. Las utilidades de administración permiten que los usuarios trabajen con nombres de usuario y de grupo, aunque internamente el sistema trabaje con esos identificadores. La ruta de la *home* puede ser inválida (por ejemplo, /nonexistent) si el usuario no dispone de ella. De la misma manera, la ruta de la shell puede apuntar a /usr/sbin/nologin: esta es una shell especial que impide el inicio de sesión, pero registra el intento de inicio de sesión en los logs. Un usuario puede no recibir una shell en algunos casos. Un servidor FTP, por ejemplo, puede ofrecer acceso a sus usuarios exclusivamente por FTP, pero no por shell (se podría deshabilitar el acceso por SSH completamente, pero esto cierra la puerta al acceso de administradores).



```
1. ubuntu@ubuntu-VirtualBox: ~
~ $ tail /etc/passwd
gnome-initial-setup:x:120:65534:./run/gnome-initial-setup:./bin/false
gdm:x:121:125:Gnome Display Manager:/var/lib/gdm3:/bin/false
ubuntu:x:1000:1000:ubuntu,,,:/home/ubuntu:/bin/bash
sshd:x:122:65534:./run/sshd:/usr/sbin/nologin
telnetd:x:123:127:./nonexistent:/usr/sbin/nologin
postfix:x:124:129:./var/spool/postfix:/usr/sbin/nologin
ftp:x:125:131:ftp daemon,,,:/srv/ftp:/usr/sbin/nologin
ntp:x:126:132:./nonexistent:/usr/sbin/nologin
mongodb:x:127:65534:./home/mongodb:/usr/sbin/nologin
ubuntu_new:x:1001:1001:Usuario Nuevo:/home/ubuntu_new:/bin/sh
~ $
```

Imagen 3. Fichero '/etc/passwd'.

El fichero '/etc/passwd' también contiene la contraseña, aunque como se puede ver en el ejemplo de la imagen 3, todos los usuarios contienen una x en el campo reservado para la misma. Originalmente, las contraseñas se alojaban en ese campo como un hash unidireccional, pero este mecanismo era susceptible a ataques de fuerza bruta, especialmente si el fichero era robado, lo que no era imposible porque el fichero debía ser legible por todos los usuarios.

Para intentar reducir el riesgo, las contraseñas se movieron a un segundo fichero, '/etc/shadow', con un hash más complejo. Este fichero, además, solo es legible por el usuario root. La imagen 4 muestra el fichero '/etc/shadow' del mismo sistema que el fichero '/etc/passwd' de la imagen 3. Solo los usuarios ubuntu y ubuntu_new tienen contraseña.



```
1. ubuntu@ubuntu-VirtualBox: ~
~ $ sudo tail /etc/shadow
gnome-initial-setup:*:17737:0:99999:7:::
gdm:*:17737:0:99999:7:::
ubuntu:$6$2M.kYu2E$hbQHRUYT18nTeELt8yv16ULuIt6zsu0t7iTmaP1GdTeE8H9Q8j.hpeDVMdkv1uQee92oTNH0IphmIIItJ/gxK.:18374:0:99999:7:::
sshd:*:18374:0:99999:7:::
telnetd:*:18380:0:99999:7:::
postfix:*:18381:0:99999:7:::
ftp:*:18381:0:99999:7:::
ntp:*:18382:0:99999:7:::
mongodb:*:18382:0:99999:7:::
ubuntu_new:$6$2BcDaFqx$QvH8sQc3jRJdc/8op8k/aqxHahx1Jz0vH0MsSHYg42au/z81vV5y1058sNkc9iw8W7GMwSvjw7ADKj6mUp/:18384:0:99999:7:::
~ $
```

Imagen 4. Fichero '/etc/shadow'.

Administración de grupos

En Linux, cada usuario debe pertenecer al menos a un grupo. La mayoría de las distribuciones crean un nuevo grupo para cada nuevo usuario. Sólo este usuario pertenece a este grupo, que será el grupo primario del usuario. Los demás grupos de los que sea miembro serán grupos suplementarios. El comando `id` muestra tanto el ID del usuario como los grupos a los que pertenece. En el siguiente ejemplo, el usuario **ubuntu** pertenece a **ubuntu** como grupo principal

```
$ id ubuntu
uid=1000(ubuntu) gid=1000(ubuntu)
groups=1000(ubuntu),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),116(lpadmin),126(sambashare
)
```

Los grupos suplementarios pueden definirse durante la creación del usuario:

```
$ sudo useradd -m -G sudo,cdrom ubuntu_sudo
$ id ubuntu_sudo
uid=1002(ubuntu_sudo) gid=1002(ubuntu_sudo)
groups=1002(ubuntu_sudo),24(cdrom),27(sudo)
```

También es posible modificar la membresía de grupos una vez creado el usuario con `usermod`.

```
$ sudo usermod -a -G plugdev ubuntu_sudo
$ id ubuntu_sudo
uid=1002(ubuntu_sudo) gid=1002(ubuntu_sudo)
groups=1002(ubuntu_sudo),24(cdrom),27(sudo),46(plugdev)
```

El comando `groupadd` permite crear grupos, mientras que `groupdel` los borra.

```
$ sudo groupadd new_group
$ sudo usermod -a -G new_group ubuntu_sudo
$ id ubuntu_sudo
uid=1002(ubuntu_sudo) gid=1002(ubuntu_sudo)
groups=1002(ubuntu_sudo),24(cdrom),27(sudo),46(plugdev),1003(new_group)
$ sudo groupdel new_group
```

El comando `groupadd` permite crear grupos, mientras que `groupdel` los borra.

```
$ sudo groupadd new_group
$ sudo usermod -a -G new_group ubuntu_sudo
$ id ubuntu_sudo
uid=1002(ubuntu_sudo) gid=1002(ubuntu_sudo)
groups=1002(ubuntu_sudo),24(cdrom),27(sudo),46(plugdev),1003(new_group)
$ sudo groupdel new_group
```

Sistema de archivos

Se suele decir que cualquier cosa en Linux es un archivo. Los archivos normales no son más que un tipo más de archivo y los directorios son archivos que contienen los nombres de otros archivos. El comando 'pwd' ofrece información sobre el directorio actual en la consola. El comando 'cd' permite cambiar de directorio.

```
$ pwd
/home/ubuntu
$ cd /var/log
/var/log
```

Árbol de directorios

El directorio raíz se identifica por la barra /. Este directorio es la base del árbol de directorios. Al contrario que Windows, todo el sistema operativo de Linux depende de un único árbol. No hay unidades de disco C: o D: como en Windows. En Linux, todas las particiones, unidades y almacenamiento se alojan en algún punto bajo la raíz /. El almacenamiento y las unidades externas se **montan** en un directorio. Este directorio aparece como una carpeta más de árbol. Por ejemplo, la ruta '/media' suele alojar las unidades de CD y las unidades de USB externas una vez montadas.

El directorio raíz es el inicio de cualquier ruta, o path, absoluta. Es posible cambiar de directorio usando rutas **absolutas** desde cualquier directorio. Por ejemplo, 'cd /var/log' funcionará aunque la consola se encuentre en '/home/ubuntu'.

Las rutas **relativas**, sin embargo, no parten de la raíz sino del directorio en el que se encuentra la consola en ese momento. Si el directorio actual es **/var**, tanto 'cd /var/log' como 'cd log' cambiarán al mismo directorio. El símbolo .. (dos puntos) denota el directorio padre al actual, mientras que . (un punto) denota al directorio actual. Ambos pueden usarse en rutas relativas.

```
$ cd /var/log
/var/log

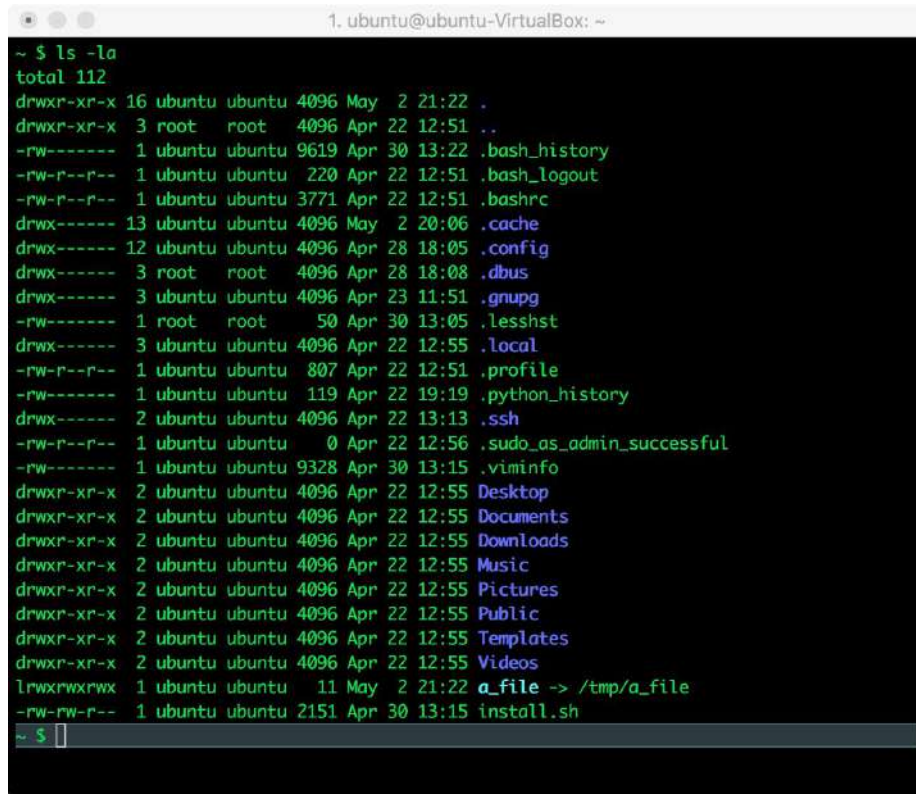
$ cd log
-bash: cd: log: No such file or directory
/var/log $ cd ..
/var $ cd log
/var/log $ cd ./cups
/var/log/cups $
```

La mayoría de las distribuciones de Linux siguen un mismo esquema de árbol de directorios. Aunque hay diferencias, rutas como /bin, /boot, /etc, /home o /tmp son prácticamente una constante en cualquier Linux. La tabla 1 lista los directorios habituales de la raíz y su contenido.

Directorios habituales de Linux	
/bin	Comandos y binarios de usuario.
/boot	Ficheros del gestor de arranque.
/dev	Archivos de dispositivos.
/etc	Ficheros de configuración.
/home	Carpetas home de usuarios.
/lib	Librerías compartidas y módulos del kernel.
/media	Punto de montaje habitual de medios externos.
/mnt	Otro punto de montaje habitual de medios externos.
/opt	Software adicional instalado en el sistema.
/proc	Detalles del núcleo y de los procesos, almacenado en modo texto.
/root	Carpeta home del usuario root.
/run	Datos de aplicaciones en ejecución.
/sbin	Binarios de sistema.
/srv	Datos de servicios provistos por el equipo.
/sys	Sistema de ficheros virtual con información del kernel.
/tmp	Ficheros temporales.
/usr	Utilidades de usuario.
/var	Datos transitorios o variables: logs, colas de correo, trabajos de impresión, etc.

Tabla 1. Directorios habituales de Linux.

Para examinar el contenido de los directorios se puede usar el comando 'ls'. Ejecutando 'ls -la' en la *home* de un usuario se obtiene la siguiente información.



```
1. ubuntu@ubuntu-VirtualBox: ~
~ $ ls -la
total 112
drwxr-xr-x 16 ubuntu ubuntu 4096 May  2 21:22 .
drwxr-xr-x  3 root   root   4096 Apr 22 12:51 ..
-rw-r--r--  1 ubuntu ubuntu 9619 Apr 30 13:22 .bash_history
-rw-r--r--  1 ubuntu ubuntu  220 Apr 22 12:51 .bash_logout
-rw-r--r--  1 ubuntu ubuntu 3771 Apr 22 12:51 .bashrc
drwxr-xr-x 13 ubuntu ubuntu 4096 May  2 20:06 .cache
drwxr-xr-x 12 ubuntu ubuntu 4096 Apr 28 18:05 .config
drwxr-xr-x  3 root   root   4096 Apr 28 18:08 .dbus
drwxr-xr-x  3 ubuntu ubuntu 4096 Apr 23 11:51 .gnupg
-rw-r--r--  1 root   root    50 Apr 30 13:05 .lessht
drwxr-xr-x  3 ubuntu ubuntu 4096 Apr 22 12:55 .local
-rw-r--r--  1 ubuntu ubuntu  807 Apr 22 12:51 .profile
-rw-r--r--  1 ubuntu ubuntu  119 Apr 22 19:19 .python_history
drwxr-xr-x  2 ubuntu ubuntu 4096 Apr 22 13:13 .ssh
-rw-r--r--  1 ubuntu ubuntu   0 Apr 22 12:56 .sudo_as_admin_successful
-rw-r--r--  1 ubuntu ubuntu 9328 Apr 30 13:15 .viminfo
drwxr-xr-x  2 ubuntu ubuntu 4096 Apr 22 12:55 Desktop
drwxr-xr-x  2 ubuntu ubuntu 4096 Apr 22 12:55 Documents
drwxr-xr-x  2 ubuntu ubuntu 4096 Apr 22 12:55 Downloads
drwxr-xr-x  2 ubuntu ubuntu 4096 Apr 22 12:55 Music
drwxr-xr-x  2 ubuntu ubuntu 4096 Apr 22 12:55 Pictures
drwxr-xr-x  2 ubuntu ubuntu 4096 Apr 22 12:55 Public
drwxr-xr-x  2 ubuntu ubuntu 4096 Apr 22 12:55 Templates
drwxr-xr-x  2 ubuntu ubuntu 4096 Apr 22 12:55 Videos
lrwxrwxrwx  1 ubuntu ubuntu  11 May  2 21:22 a_file -> /tmp/a_file
-rw-rw-r--  1 ubuntu ubuntu 2151 Apr 30 13:15 install.sh
~ $
```

Imagen 5. Comando 'ls -la' en la home del usuario.

Tipos de archivos

Los ficheros que comienzan con **un punto** son archivos ocultos. El comando 'ls' no lo muestra por defecto a menos que se invoque con el modificador *-a*. Son archivos normales a todos los efectos. Es habitual que las configuraciones específicas de ciertas aplicaciones se guarden como archivos ocultos en la *home*. Por ejemplo, '.viminfo' son detalles de vim, mientras que '.python_history' guarda el histórico de la shell de Python.

La primera columna de detalles muestra información sobre el tipo de archivo y los permisos. El primer carácter de la columna indica el tipo de archivo:

- Archivos 'normales' - sin flag.
- Directorios - **d**.
- Enlaces - **l**.
- Dispositivos de bloque - **b**.
- Dispositivos de carácter - **c**.
- Sockets - **s**.
- Pipes - **p**.

Sistema de permisos

Los permisos del archivo se indican con los últimos 9 caracteres de la primera columna. En Linux, los permisos se usan para determinar de qué acceso disponen los usuarios y grupos sobre un archivo.

El control de permisos sobre archivos y aplicaciones es **crítico** para la seguridad y el correcto funcionamiento de un host.

Un permiso erróneo puede suponer un agujero de seguridad (por ejemplo, si todas las subcarpetas de /home permiten lectura al resto de usuarios en un servidor de FTP) o impedir que un servicio funcione correctamente (por ejemplo, si un servidor nginx se ejecuta con el usuario www-data, pero la carpeta de archivos estáticos solo permite acceso al usuario root). Los tres permisos principales son:

- **[r]** Lectura - Permite leer o ver un archivo normal, pero no los detalles del contenido de un directorio.
- **[w]** Escritura/edición - Permite hacer cambios sobre un archivo o utilizar un directorio (crear, borrar o renombrar sus ficheros).
- **[x]** Ejecución - en el caso de scripts o ficheros con un *shebang* ejecutable, permite su ejecución directa por el sistema.

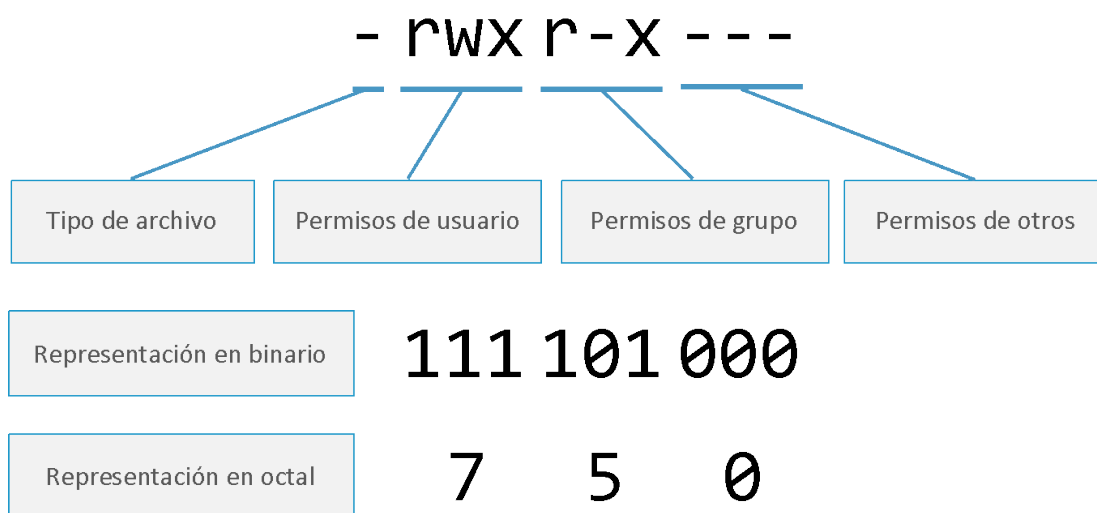


Imagen 6. Columna de tipo de archivo y permisos y su representación binaria y octal.

Los flags están a su vez agrupados en 3 clases:

- Usuario - Los permisos aplican al usuario dueño del recurso.
- Grupo - Los permisos aplican al grupo al que pertenece el recurso y, por extensión, a todos los usuarios que pertenezcan a ese grupo.
- Otros - Los permisos aplican a todos los demás usuarios que no estén incluidos en lo anterior.

El formato de los flags no es arbitrario, ya que se corresponde con la representación **binaria**, de forma que cada flag es un bit. Si el permiso está activo, el bit estará a uno. Una presentación muy habitual de los permisos es agrupar los 3 bits de cada clase y mostrar la representación octal de esos bits. Así, según muestra la imagen 6, si los tres flags están activos, la representación octal de 111 es 7, mientras que si solo los flags de lectura y ejecución están activos, la representación octal de 101 es 5.

Este formato es muy útil para modificar la configuración de permisos de un archivo. El comando que permite cambiar los permisos es 'chmod' y acepta dos sintaxis:

- Clase|activar/desactivar|permiso.
- Representación binaria.

Por ejemplo, dado un archivo con permisos rw-rwx---, es posible activar la ejecución para el usuario propietario, desactivar la ejecución para el grupo y activar la lectura para el resto con los dos comandos siguientes:

```
$ ls -la
-rw-r--r-- 1 ubuntu wheel 53248 Nov 29 10:15 .coverage
$ chmod u+x,g-x,o+r .coverage

$ ls -la .coverage
-rwxr--r-- 1 ubuntu wheel 53248 Nov 29 10:15 .coverage

$ chmod 764 file
$ ls -la .coverage
-rwxrw-r-- 1 ubuntu wheel 53248 Nov 29 10:15 .coverage
```

La tabla 2 muestra algunos ejemplos de permisos junto a su representación en octal.

Permisos habituales	
600	rw-----
644	rw-r--r--
664	rw-rw-r--
666	rw-rw-rw-
755	rxwxr-xr-x
777	rxwxrwxrwx

Tabla 2. Ejemplos de permisos habituales en formato octal.

Enlaces

Hay dos tipos de enlaces: enlaces fuertes y enlaces débiles o simbólicos. Los enlaces fuertes, o **hard links**, son referencias reales al archivo. Si se borran todos los enlaces fuertes a un archivo, el archivo se borra también. Un enlace fuerte solo se puede crear en la misma partición que el archivo al que hace referencia.

Los enlaces simbólicos (**soft links/symbolic links**) se pueden borrar sin afectar al fichero al que hacen referencia. Es habitual que los archivos binarios tengan enlaces simbólicos para poder acceder a ellos con varios nombres o para poder cambiar la versión de un binario sin cambiar el nombre con el que se accede.

Por ejemplo, la imagen 7 muestra los ejecutables de Python en la carpeta '/usr/bin'. La versión instalada de Python es la 3.6, tal como se indica en el binario python3.6. Sin embargo, es posible ejecutarlo tanto con python como con python3. Una actualización a Python 3.8 cambiaría estos enlaces simbólicos para que apunten al nuevo binario, pero manteniendo los nombres python y python3.

```
1. ubuntu@ubuntu-VirtualBox: ~
~ $ ll /usr/bin/python*
lrwxrwxrwx 1 root root      9 May  2 22:43 /usr/bin/python -> python3.6*
lrwxrwxrwx 1 root root     10 Apr 22 12:49 /usr/bin/python3 -> python3.6*
-rwxr-xr-x 1 root root 4576440 Apr  1 2018 /usr/bin/python3.6*
-rwxr-xr-x 1 root root 4576440 Apr  1 2018 /usr/bin/python3.6m*
lrwxrwxrwx 1 root root     10 Apr 22 12:49 /usr/bin/python3m -> python3.6m*
~ $
```

Imagen 7. Enlaces simbólicos.

Los enlaces se crean con el comando 'ln'. Los enlaces serán fuertes por defecto y simbólicos con el flag '-s'. La manera más sencilla de crear un enlace simbólico en la ruta actual es simplemente indicando la ruta del archivo enlazado (ver imagen 8).

```
1. ubuntu@ubuntu-VirtualBox: ~
~ $ ln -s /usr/bin/python
~ $ ll python*
lrwxrwxrwx 1 ubuntu ubuntu 15 May  2 23:09 python -> /usr/bin/python*
~ $ ./python --version
Python 3.6.5
~ $
```

Imagen 8. Creación de enlace simbólico.

Procesos y servicios

Al igual que otros sistemas operativos, en Linux existe el concepto de servicio, también llamado daemon. Los servicios son procesos que no terminan inmediatamente tras la ejecución, ofreciendo ciertas funcionalidades al sistema operativo o a los usuarios. Algunos ejemplos de servicios son sshd (el daemon de SSH) o http del servidor web Apache.

El comando 'ps aux' muestra todos los procesos en ejecución, tanto servicios como otros procesos, ordenados por PID (*process ID*).

```
1. ubuntu@ubuntu-VirtualBox: ~
~ $ ps aux | grep -v '\.*/\.'
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.4 160084  9212 ?        Ss   May02   0:02 /sbin/init splash
root       228  0.0  1.5 127908 31056 ?        Ss   May02   0:00 /lib/systemd/systemd-journald
root       238  0.0  0.2  46744  4744 ?        Ss   May02   0:00 /lib/systemd/systemd-udev
systemd+  313  0.0  0.3  70744  6344 ?        Ss   May02   0:00 /lib/systemd/systemd-resolved
root       488  0.0  0.4  427256  8848 ?        Ssl  May02   0:00 /usr/sbin/ModemManager
syslog    492  0.0  0.2  263036  4208 ?        Ssl  May02   0:00 /usr/sbin/rsyslogd -n
root      502  0.0  0.1  38428  3340 ?        Ss   May02   0:00 /usr/sbin/cron -f
message+  505  0.0  0.2  51492  5632 ?        Ss   May02   0:00 /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation --syslog-only
root       526  0.0  0.9 580244 18872 ?        Ssl  May02   0:00 /usr/sbin/NetworkManager --no-daemon
root       527  0.0  0.2  44752  5116 ?        Ss   May02   0:00 /sbin/wpa_supplicant -u -s -O /run/wpa_supplicant
root       530  0.0  0.6 518004 13216 ?        Ssl  May02   0:00 /usr/lib/udisks2/udisksd
root       531  0.0  0.0   4552   752 ?        Ss   May02   0:00 /usr/sbin/acpid
root       535  0.0  0.8 177636 17308 ?        Ssl  May02   0:00 /usr/bin/python3 /usr/bin/networkd-dispatcher --run-startup-triggers
root       541  0.0  0.2  70688  6036 ?        Ss   May02   0:00 /lib/systemd/systemd-logind
root       543  0.0  0.4 311004  9092 ?        Ssl  May02   0:00 /usr/lib/accounts-service/accounts-daemon
root       561  0.0  1.2 658696 25736 ?        Ssl  May02   0:03 /usr/lib/snapd/snapd
avahi     564  0.0  0.0  47076   336 ?        S   May02   0:00 avahi-daemon: chroot helper
root      596  0.0  0.5 311308 11304 ?        Ssl  May02   0:00 /usr/lib/policykit-1/polkitd --no-debug
root      650  0.0  0.1  28676  2812 ?        Ss   May02   0:00 /usr/sbin/vsftpd /etc/vsftpd.conf
root      653  0.0  0.2  72296  5680 ?        Ss   May02   0:00 /usr/sbin/sshd -D
root      667  0.0  0.4 308180  8664 ?        Ssl  May02   0:00 /usr/sbin/gdm3
whoopsie  729  0.0  0.6 466612 12808 ?        Ssl  May02   0:00 /usr/bin/whoopsie -f
root      730  0.0  0.1  33992  3232 ?        Ss   May02   0:00 /usr/sbin/inetd
kernoops  751  0.0  0.0  56932   420 ?        Ss   May02   0:00 /usr/sbin/kerneloops --test
kernoops  755  0.0  0.0  56932   416 ?        Ss   May02   0:00 /usr/sbin/kerneloops
ubuntu    786  0.0  0.4  77024  8188 ?        Ss   May02   0:00 /lib/systemd/systemd --user
ubuntu    795  0.0  0.1 114132  2704 ?        S   May02   0:00 (sd-pam)
ubuntu    959  0.0  0.3 288380  6688 ?        Ssl  May02   0:00 /usr/bin/gnome-keyring-daemon --daemonize --login
ubuntu    965  0.0  0.3 212124  6144 tty1    Ssl+ May02   0:00 /usr/lib/gdm3/gdm-x-session --run-script env GNOME_SHELL_SESSION_MODE=ubuntu gnome-session --session=ubuntu
ubuntu   974  0.0  3.4 409948 70408 tty1    Sl+  May02   0:02 /usr/lib/xorg/Xorg vt1 -displayfd 3 -auth /run/user/1000/gdm/Xauthority -b
```

Imagen 9. Comando 'ps aux'.

El comando 'top' es una herramienta de monitorización interactiva que muestra los procesos en funcionamiento en el equipo. Al contrario que 'ps', 'top' se actualiza cada pocos segundos. Por defecto, 'top' ordena los procesos por el uso instantáneo de CPU, por lo que los primeros procesos de la lista son los que más cargan el equipo.

Imagen 10. Comando 'top' durante una ejecución de apt-get update.

Los procesos pueden terminar por sí mismos si terminan su cometido o si sufren un error que provoca que se cierren con un código de salida diferente de 0. En algunos casos, no obstante, es necesario detener los procesos antes de tiempo. Para eso, podemos usar la familia de comandos 'kill.' que se basan en códigos de señales, identificadas por un código numérico. Algunos de los más comunes son:

- SIGHUP (1) - Fuerza a un proceso a releer su configuración, sin parar.
- SIGKILL (9) - Termina un proceso de forma abrupta.
- SIGTERM (15) - Solicita a un proceso que inicie la terminación, tomando las medidas que necesite.
- SIGUSR1 / SIGUSR2 (10 y 12) - Señales específicas que el proceso puede capturar e interpretar como necesite.

Para enviar esas señales se emplea el comando 'kill', que necesita también el PID del proceso obtenido de 'ps', 'top' o al vuelo usando esta sencilla sustitución en Bash:

```
$ sudo kill -9 $(pidof https)
$ sudo kill -9 `pidof https` # sintaxis antigua, a deprecarse
```

El comando 'killall' es un poco más cómodo de usar para un usuario, ya que permite especificar el nombre del proceso. Como su nombre indica, 'killall' terminará todos los procesos con ese nombre. También permite especificar un usuario, por lo que terminará los procesos de ese usuario. Si el usuario ha iniciado una sesión interactiva, 'killall' terminará también el proceso de consola y, por tanto, terminará la sesión del usuario.

```
$ killall -u username
$ killall -r httpd
```

Finalmente, **'pkill'** permite terminar procesos también con el nombre del proceso. Es también capaz de terminar los procesos hijos de un proceso padre con el flag **'-P'**.

```
$ pkill -P pid
```

Administración de servicios

Los servicios se pueden administrar como procesos con el comando **'kill'**, pero es posible usar la funcionalidad de **Systemd** para obtener más información. La utilidad **'systemctl'** permite arrancar, parar, recargar y obtener información de los servicios arrancados con Systemd. Por ejemplo, para el servicio de SSH, **'systemctl'** muestra la información mostrada en la imagen 11.

```

2. ubuntu@ubuntu-VirtualBox: ~
ubuntu@ubuntu-VirtualBox:~$ systemctl status sshd
• ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2020-05-02 20:05:40 CEST; 6h ago
   Main PID: 653 (sshd)
   Tasks: 1 (limit: 2322)
   CGroup: /system.slice/ssh.service
           └─653 /usr/sbin/sshd -D

May 02 23:37:52 ubuntu-VirtualBox sshd[3993]: Accepted password for ubuntu_new from 192.168.1.132 port 53683 ssh2
May 02 23:37:52 ubuntu-VirtualBox sshd[3993]: pam_unix(sshd:session): session opened for user ubuntu_new by (uid=0)
May 02 23:39:06 ubuntu-VirtualBox sshd[4138]: Accepted password for ubuntu_new from 192.168.1.132 port 53691 ssh2
May 02 23:39:06 ubuntu-VirtualBox sshd[4138]: pam_unix(sshd:session): session opened for user ubuntu_new by (uid=0)
May 03 02:09:51 ubuntu-VirtualBox sshd[4606]: Accepted password for ubuntu from 192.168.1.132 port 55535 ssh2
May 03 02:09:51 ubuntu-VirtualBox sshd[4606]: pam_unix(sshd:session): session opened for user ubuntu by (uid=0)
May 03 02:58:10 ubuntu-VirtualBox sshd[5008]: Accepted publickey for ubuntu from 192.168.1.132 port 56402 ssh2: RSA SHA256:chuAsD
May 03 02:58:10 ubuntu-VirtualBox sshd[5008]: pam_unix(sshd:session): session opened for user ubuntu by (uid=0)
May 03 02:58:49 ubuntu-VirtualBox sshd[5098]: Accepted password for ubuntu_new from 192.168.1.132 port 56405 ssh2
May 03 02:58:49 ubuntu-VirtualBox sshd[5098]: pam_unix(sshd:session): session opened for user ubuntu_new by (uid=0)
ubuntu@ubuntu-VirtualBox:~$

```

Imagen 11. Información del servicio sshd con **'systemctl'**.

El campo **'Loaded'**, en la segunda línea, indica el fichero utilizado para arrancar el servicio. Este *unit file* no es el binario como tal, sino un fichero de configuración de Systemd que indica, entre otras cosas, condiciones de ejecución, la ruta al binario con los flags necesarios para el arranque y ficheros con variables de entorno. El fichero de Systemd de sshd se muestra a continuación.

```
$ cat /lib/systemd/system/ssh.service
[Unit]
Description=OpenBSD Secure Shell server
After=network.target auditd.service
ConditionPathExists=!/etc/ssh/sshd_not_to_be_run

[Service]
EnvironmentFile=/etc/default/ssh
ExecStartPre=/usr/sbin/sshd -t
ExecStart=/usr/sbin/sshd -D $SSH_OPTS
ExecReload=/usr/sbin/sshd -t
```



```
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartPreventExitStatus=255
Type=notify
RuntimeDirectory=sshd
RuntimeDirectoryMode=0755
```

```
[Install]
WantedBy=multi-user.target
Alias=sshd.service
```

Estos ficheros son editables por un administrador y es posible definir ficheros propios para configurar aplicaciones propias como servicios.

La información provista por ‘systemctl’ va más allá. Indica también el estado del servicio, *active (running)* en este caso. El estado puede ser *inactive (dead)* si se ha parado el servicio o *failed*, con el código de salida, si el proceso terminó de manera abrupta. Además, las últimas líneas del fichero de log se incluyen en la salida de ‘systemctl status’, lo que puede ser útil si el proceso ha detectado un error antes de detenerse.

Además de ofrecer información, ‘systemctl’ permite arrancar y parar los servicios con ‘**systemctl start**’ y ‘**systemctl stop**’. La imagen 12 muestra ejemplos de ambos comandos.

```
2. ubuntu@ubuntu-VirtualBox: ~
ubuntu@ubuntu-VirtualBox:~$ sudo systemctl status vsftpd
• vsftpd.service - vsftpd FTP server
   Loaded: loaded (/lib/systemd/system/vsftpd.service; enabled; vendor preset: enabled)
   Active: failed (Result: exit-code) since Sun 2020-05-03 03:14:41 CEST; 3min 2s ago
   Process: 5998 ExecStart=/usr/sbin/vsftpd /etc/vsftpd.conf (code=exited, status=2)
   Process: 5997 ExecStartPre=/bin/mkdir -p /var/run/vsftpd/empty (code=exited, status=0/SUCCESS)
   Main PID: 5998 (code=exited, status=2)

May 03 03:14:41 ubuntu-VirtualBox systemd[1]: Starting vsftpd FTP server...
May 03 03:14:41 ubuntu-VirtualBox systemd[1]: Started vsftpd FTP server.
May 03 03:14:41 ubuntu-VirtualBox systemd[1]: vsftpd.service: Main process exited, code=exited, status=2/INVALIDARGUMENT
May 03 03:14:41 ubuntu-VirtualBox systemd[1]: vsftpd.service: Failed with result 'exit-code'.
ubuntu@ubuntu-VirtualBox:~$ sudo systemctl start vsftpd
ubuntu@ubuntu-VirtualBox:~$ sudo systemctl status vsftpd
• vsftpd.service - vsftpd FTP server
   Loaded: loaded (/lib/systemd/system/vsftpd.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2020-05-03 03:17:47 CEST; 2s ago
   Process: 6013 ExecStartPre=/bin/mkdir -p /var/run/vsftpd/empty (code=exited, status=0/SUCCESS)
   Main PID: 6014 (vsftpd)
   Tasks: 1 (limit: 2322)
   CGroup: /system.slice/vsftpd.service
           └─6014 /usr/sbin/vsftpd /etc/vsftpd.conf

May 03 03:17:47 ubuntu-VirtualBox systemd[1]: Starting vsftpd FTP server...
May 03 03:17:47 ubuntu-VirtualBox systemd[1]: Started vsftpd FTP server.
ubuntu@ubuntu-VirtualBox:~$ sudo systemctl stop vsftpd
ubuntu@ubuntu-VirtualBox:~$
```

Imagen 12. Arranque y parada de servicios con ‘systemctl’.

Es necesario identificar el servicio con el nombre, y ese servicio no siempre tiene el mismo nombre que el proceso que ejecuta. Es posible listar todos los servicios en **ejecución** filtrando la salida de ‘systemctl’ usando *grep running*, o todos los servicios **habilitados** (que pueden no estar en ejecución si se han parado manualmente o si han fallado) con *grep enabled*.

Las primeras líneas muestran un resumen de información del sistema, en este orden:

- Hora actual, *uptime* del sistema, número de usuarios que han iniciado sesión y la carga media de CPU durante el último minuto, últimos 5 minutos y últimos 15 minutos.
- Número total de procesos y el número en cada estado.
- Uso de CPU y porcentaje de CPU dedicado a procesos de usuario, a procesos del núcleo del sistema y sin uso.
- Uso de memoria física: total, libre y en uso, en bytes.
- Uso de memoria virtual: espacio total de swap, libre y en uso.

uptime

El comando 'uptime' muestra por consola un resumen del estado del sistema: la hora actual, el tiempo que el sistema ha estado arrancado, el número de usuarios conectados y el uso medio de CPU en el último minuto, últimos 5 minutos y últimos 15 minutos.

\$ uptime

23:32:56 up 37 min, 2 users, load average: 0.00, 0.00, 0.05

vmstat

El comando 'vmstat' ofrece un resumen de métricas de rendimiento en intervalos definidos por el usuario. En vez de actualizar los valores en la consola, como 'top', imprime los valores de las métricas en una línea periódicamente.

Cada 3 segundos, hasta un máximo de 8 entradas

\$ vmstat 3 8

```
procs -----memory----- ---swap-- -----io---- -system-- -----cpu-----
r b swpd free buff cache si so bi bo in cs us sy id wa st
1 0 0 532320 33792 297184 0 0 270 21 64 101 1 2 96 1 0
0 0 0 532320 33800 297184 0 0 0 7 54 66 0 0 100 0 0
0 0 0 532320 33800 297184 0 0 0 0 33 29 0 0 100 0 0
0 0 0 532320 33808 297184 0 0 0 7 61 73 0 0 100 0 0
1 0 0 532320 33816 297176 0 0 0 80 47 52 0 0 99 0 0
0 0 0 532320 33816 297188 0 0 0 0 44 46 0 0 100 0 0
1 0 0 532320 33824 297188 0 0 0 12 49 50 0 0 100 0 0
0 0 0 532320 33824 297188 0 0 0 0 25 21 0 0 100 0 0
```

df

El comando df muestra el espacio disponible en los sistemas de ficheros montados... El modificador '-h' imprime los valores en unidades más legibles para un usuario, como KB, MB, etc.

\$ df -h

Filesystem	Size	Used	Avail	Use%	Mounted on
udev	474M	0	474M	0%	/dev
tmpfs	99M	948K	98M	1%	/run
/dev/sda1	39G	1.4G	38G	4%	/
tmpfs	491M	0	491M	0%	/dev/shm
tmpfs	5.0M	0	5.0M	0%	/run/lock
tmpfs	491M	0	491M	0%	/sys/fs/cgroup
/dev/loop0	62M	62M	0	100%	/snap/core20/1270
/dev/loop1	44M	44M	0	100%	/snap/snapd/14295
/dev/loop2	68M	68M	0	100%	/snap/lxd/21835
tmpfs	99M	0	99M	0%	/run/user/1000
vagrant	234G	180G	55G	77%	/vagrant

Programación de tareas

Ciertos programas de Linux no están pensados para ejecutarse continuamente como un servicio, pero es necesario ejecutarlos en un instante futuro dado o periódicamente. Algunas de estas tareas pueden ser copia de seguridad, análisis de logs o descarga de grandes ficheros (por ejemplo, ficheros de actualización o sincronización de mirrors) en horario nocturno.

Para ello, se puede usar el servicio **'cron'** y el comando **'crontab'**. **'Cron'** no permite trabajos con más de un comando, pero siempre se puede escribir un script con los comandos necesarios. El servicio comprueba los trabajos disponibles cada minuto y ejecuta los que deban ejecutarse en ese momento.

Los pasos para programar un trabajo de **'cron'** son:

- Preparar el script o el comando que se ejecutará.
- Preparar un archivo de texto con la programación y el comando. El archivo puede contener más de una tarea, cada una con su programación.
- Ejecutar **'crontab <archivo>'** para confirmar la definición.
- Confirmar que el trabajo está definido con **'crontab -l'**.

El formato de definición de tareas es el siguiente:

```
5 0 * * * ps aux > /tmp/$(date +%Y%M%d$h%m)-ps.log
```

Se recomienda utilizar herramientas online que nos ayuden a verificar el formato de la expresión hasta que tengamos cierta soltura como, por ejemplo, [Crontab.guru](#) o [Crontab Format](#). Unos ejemplos típicos son:

- **5 0 * * *** → Todos los días a las 00:05.
- **0 3 1 * *** → El día 1 de cada mes a las 03:00.
- *** * * * 2** → Cada minuto los martes.

Cuando un usuario ejecuta 'crontab -l', el comando solo muestra los trabajos programados por él mismo. Por ejemplo, si el usuario root define el trabajo del ejemplo, su listado de trabajo sería el siguiente:

```
$ crontab -l
5 0 * * * ps aux > /tmp/$(date +%Y%M%d%H%m)-ps.log
```

Es decir, exactamente el contenido del fichero con el que se ha definido el trabajo. Sin embargo, el fichero '/etc/crontab' contiene la programación de trabajos de sistema. Por ejemplo, en un equipo Ubuntu, el fichero '/etc/crontab' por defecto contiene lo siguiente:

```
$ cat /etc/crontab
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
```

Además de unas variables de entorno que se aplicarán a cada uno de los trabajos, el fichero contiene 4 trabajos que corresponden con tareas horarias, diarias, semanales y mensuales. El comando *run-parts* ejecutará los scripts contenidos en cada una de las carpetas '/etc/cron.*' siguiendo la programación del fichero 'crontab'.

Los usuarios y los servicios de sistema pueden **aprovechar esta funcionalidad** para añadir scripts recurrentes en estas carpetas y así delegar la ejecución periódica a cron.

unir LA UNIVERSIDAD
EN INTERNET | FORMACIÓN
PROFESIONAL

PROEDUCA