Infraestructura I

Infraestructura I tiene por objetivo brindar al estudiante una visión holística del mundo moderno de Infraestructura, abordando conceptos como:

* cloud computing,
* containers (docker),
* configuration management,
* entre otros.

Sin dejar de lado los conocimientos fundamentales de infraestructura como virtualización, redes y sistemas operativos.

Objetivo:

Explicar que es una Máquina Virtual.

Explicar el uso de Certificados.

Entender y explicar el propósito de automatizar tareas en el mundo de infraestructura y que herramientas existen dentro del scripting.

Explicar exitosamente el concepto de contenedores y su ecosistema, y la diferencia entre una Máquina Virtual y un Contenedor.

Explicar el concepto de Cloud Computing, el modelo de responsabilidades.

Tareas:

Desarrollar scripts en Bash y PowerShell para resolver problemas sencillos.

Implementar un certificado en un servidor web.

Utilizar Ansible para desplegar configuraciones sencillas en un Sistema Operativo Linux.

Generar imágenes de contenedores e instanciarlos localmente.

Desplegar recursos de red, compute y almacenamiento en AWS.

**Inmersión**

Introducción a la Infraestructura tradicional. Se desarrollarán conocimientos básicos pero esenciales como los distintos sistemas operativos disponibles, los más populares, cuál es su rol en el ecosistema de infraestructura. Así como conceptos básicos sobre los procesos tradicionales que se utilizan para operar un ambiente de infraestructura.

Introducción a la materia & The Big Picture

Presentar brevemente al alumno el syllabus de la materia para fijar expectativas. Se hará una introducción de todo lo que se abordará en la materia  con el propósito de dar al alumno una visión integradora y general de todos los conceptos.

**Automatización**

El nuevo mundo de infraestructura demanda que los administradores y analistas de infraestructura adopten técnicas y tecnologías similares o inspiradas en el mundo del desarrollo. El principal objetivo es poder automatizar procesos y así enfocar los esfuerzos en actividades que agreguen valor y no actividades repetitivas. Así también como aumentar la predictibilidad de los resultados de los procesos del área minimizando el error.

### MODULO 1: Introduccion

Introduccion a la materia

La función de este profesional —conocido como administrador de infraestructura— es brindar respuestas a los problemas habituales de hardware y software en cualquier ámbito informatizado, planteándose desde una problemática y evaluando su mejor solución. Este servicio debe estar en manos de profesionales que orienten, configuren, prevengan y resuelvan eventualidades, manteniendo sus equipos en un estado óptimo. Además, el administrador de infraestructura debe ser capaz de prestar servicios de administración y soporte de sistemas de base y elementos de infraestructura para el procesamiento de aplicaciones informáticas, tales como servidores, dispositivos de almacenamiento masivo, otros dispositivos de hardware, sistemas operativos, máquinas virtuales y administradores de redes, servicios de comunicaciones a través de redes públicas y privadas, dispositivos de switching, firewalls, motores de bases de datos, entre otros.

Podrá además brindar servicios de administración de la infraestructura tecnológica en la cual opera el software de estas aplicaciones interviniendo en forma puntual para resolver los problemas que experimente esa infraestructura o su eficiencia operativa y realizar un diagnóstico de incidentes que se presenten en la operatoria habitual del sistema.

Arquitectura Cliente – Servidor

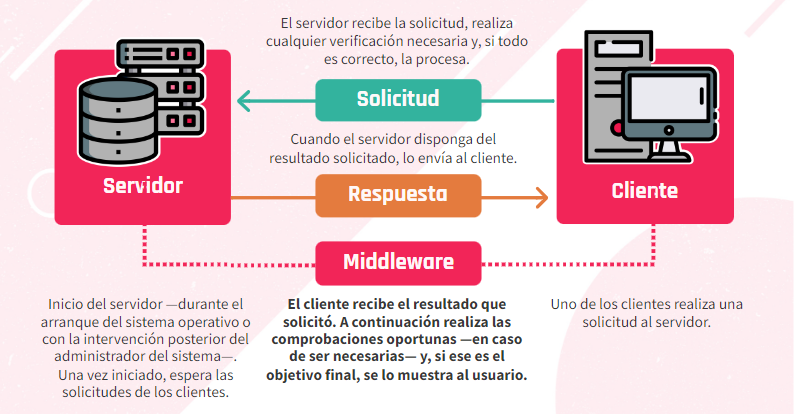
La arquitectura cliente-servidor persigue el objetivo de procesar la información de un modo distribuido. De esta forma, pueden estar dispersos en distintos lugares y acceder a recursos compartidos.

Además de la transparencia y la independencia del hardware y del software, una implementación cliente-servidor debe tener las siguientes características:

 Utilizar protocolos asimétricos, donde el servidor se limita a escuchar en espera de que un cliente inicie una solicitud.

 El acceso es transparente, multiplataforma y multiarquitectura.

 Se facilitará la escalabilidad, de manera que sea fácil añadir nuevos clientes a la infraestructura —escalabilidad horizontal— o aumentar la potencia del servidor o servidores, aumentando su número o su capacidad de cálculo —escalabilidad vertical—.



Este esquema lo que hace es mostrar como funciona la comunicacion entre un cliente y un servidor. Algo que hacemos constantemente en el dia a dia. En primer lugar debemos saber que un cliente (que puede ser un navegador, o aplicacion), hace una request o peticion. En esa peticion el cliente solicita al servidor por ejemplo abrir un archivo o ir a una pagina web. Esa comunicacion no se hace de manera directa ya que éstos hablan diferentes lenguajes, y para esto existe el middleware, que es un software encargado de traducir la peticion en un idioma entendible por el servidor. Despues de recibida la request o peticion, lo que hace el servidor es procesar la peticion y preparar una response o respuesta que va a pasar por el mismo proceso que la request, donde va a ser traducida por el middleware para que el cliente la reciba, pueda interpretarla y asi mostrarla. ¿Pero, que tienen que ver los Sistemas Operativos con esta estructura cliente-servidor? Los sistemas operativos son parte fundamental en este funcionamiento ya que son quienes administran los recursos fisicos (hardware) del servidor y los protocolos de ejecusion del software.

Características fundamentales de un Sistema Operativo

**Soporte de Red:** Es indispensable que tengan un soporte completo para poder brindar conectividad.

**Amplia compatibilidad con el hardware:** Un punto fundamental para el aprovechamiento pleno de las características del servidor es que el S.O. sea capaz de exprimir al máximo las características técnicas del hardware en donde se ejecuta, es por ello que se debe priorizar el uso de S.O. actualizados y con un soporte importante de controladores. Por ejemplo, que nuestros controladores permitan acceder a características avanzadas de gestión de discos rígidos, con el propósito de realizar arreglos redundantes para tener mayor velocidad y tolerancia a fallos.

**Seguridad:** Teniendo en cuenta el rol que cumple el servidor, es de vital importancia que el S.O. instalado sea seguro; eso implica no solo que este actualizado con todos los parches/actualizaciones, sino que además debe tener aplicadas políticas estrictas de acceso para prevenir accesos no autorizados o ataques.

Adicionalmente esa seguridad debe reforzarse con la instalación de Firewalls (por software o hardware) y antivirus. En este ítem también debemos incluir el respaldo de la información, ya sea por medio de herramientas que el propio S.O. nos ofrezca o instalando software externo, con el propósito de tener la menor pérdida de datos posible en caso de fallos fatales.

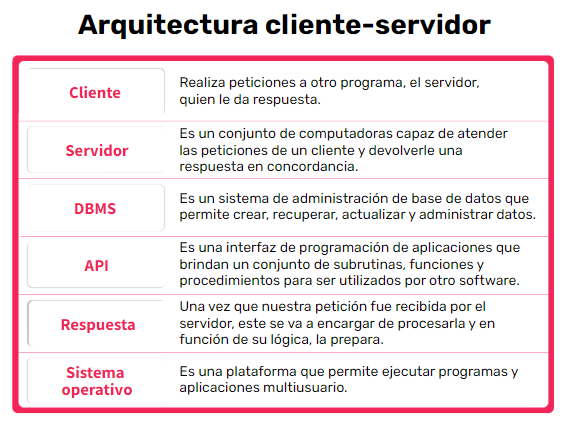
**Tolerancia a Fallos**: Debemos priorizar S.O. que en su arquitectura permita una tolerancia a fallos, ya sea mediante la generación de granja de servidores, que interconectados, operen como una gran unidad de proceso, dando la posibilidad que ante la caída de uno de los integrantes de la granja, otro puede tomar su rol y responsabilidad.

**¿Por que no usar los mismos sistemas operativos para un servidor que para un ordenador común?**

Manejo diferente del hardware: Debido al diferente propósito, los S.O. de estaciones de trabajo no pueden aprovechar todo el hardware disponible, como, por ejemplo, el manejo de memoria RAM —teniendo el caso de Windows 10 64-bit que puede manejar 6TB de RAM, mientras que Windows Server 2019 alcanza los 24TB—.

Características soportadas: Hay funcionalidades que nativamente un S.O. de estación de trabajo no es capaz de brindar, ya que en su versión de kernel están limitadas o deshabilitadas —casos tales como la virtualización en algunas versiones de Windows 10—.

Soporte: Algo muy importante a tener en cuenta es, cuando nuestro negocio o aplicación depende de un S.O., es el soporte por parte del fabricante/desarrollador. En el caso de los S.O. de estación de trabajo, el soporte/cobertura que tenemos es para un uso específico, si sobre esa base quisiéramos desplegar una arquitectura, por ejemplo, de servidor web, si bien es probable que nos funcione, vamos a carecer de soporte técnico ya que el fabricante nos indicará que para ese propósito esta la version “Server” del producto.



### MODULO 2: Automatización y Virtualización

AUTOMATIZACION

Automatización de la Infraestructura -> consiste en el uso de sistemas de software para crear instrucciones y procesos repetibles a fin de reemplazar o reducir la interacción humana.

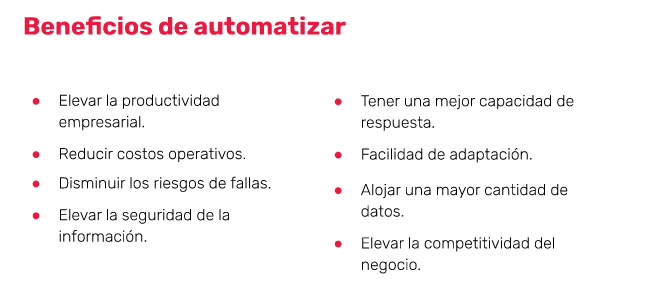
Infraestructura IT es un conjunto de aplicaciones y dispositivos de software necesarios para que cualquier empresa opere.

Se compone de elementos como software, hardware, redes, instalaciones, y todo lo que se requiera para desarrollar, controlar, monitorear, y dar soporte a los servicios que ofrece el departamento IT.

La automatización consiste en utilizar tecnología para realizar tareas casi sin necesidad de que intervengan las personas, y se puede implementar en cualquier sector que lleve a cabo tareas repetitivas.

**¿Por qué automatizar?**

Automatizar tareas permite ganar tiempo y maximizar la productividad de nuestra infraestructura IT.





Servidores legacy que son?

//Complemento: documento “Configuración y mantenimiento del sistema”

### VIRTUALIZACION

Introducción a la virtualización

La virtualización permite mejorar la agilidad, la flexibilidad y la escalabilidad de la infraestructura de IT, al mismo tiempo que proporciona un importante ahorro de costos.

Algunas ventajas de la virtualización son:

* Mayor movilidad de las cargas de trabajo.
* Aumento del rendimiento.
* Menos esfuerzo en los upgrades/updates del sistema.
* Mejor disponibilidad de los recursos o la automatización de las operaciones:
* Simplifican la gestión de la infraestructura de IT.
* Permiten reducir los costos de propiedad y operativos.



*NOTAS YOUTUBE*

*Maquina Virtual, que es?*

*Una plataforma de virtualizacion utiliza las caracteristicas necesarias de hardware y software para permitirnos ejecutar multiples maquinas virtuales en una misma computadora.*

*Una MV es un medio encapsulado. Cualquier cosa que se haga dentro de la maquina virtual no impactara en el equipo fisico, osea maquina anfitriona.*

*Puede funcionar con su hardware propio, direccion IP propia, puede tener su propia direccion MC, todo como si fuese una maquina comun y corriente.*

*Cuando se crea la maquina se establece la cantidad de memoria ram, cantidad de memoria del disco, etc, y se restan estos recursos del equipo anfitrion.*

*Cuando se virtualiza un S.O., nuestro S.O el que tenemos instalado en nuestra computadora, necesita unos recursos minimos para poder funcionar, por lo tanto con la maquina virtual no podemos usar todos los recursos porque de este modo no va a funcionar la virtualizacion.*

*Ventajas de la Virtualizacion*

*Permite probar multitud de S.O sin necesidad de tocar un equipo anfitrion*

*Proporcionan la misma experiencia que un equipo anfitrion normal. Se comportan igual que si fuesen un equipo anfitrion.*

*Se pueden hacer snapshots. Esto quiere decir "volver atras", por ejemplo si durante una instalacion ocurre algun problema, se puede acudir a instancias anteriores sin necesidad de formatear nuevamente.*

*El ahorro de dinero es otro punto importante. Suponiendo el caso de los servidores. Con los recursos de un mismo servidor se puede dar funcion a todos los sistemas operativos utilizando maquinas virtuales. Sin éstas se deberia tener un servidor para cada sistema operativo (windows, mc, debian, etc).*

*Desventajas de la Virtualizacion*

*Se requiere de un equipo bastante potente para llevar a cabo la virtualizacion de una manera fluida y correcta.*

*Nunca sera igual un sistema operativo instalado en una maquina virtual respecto a un equipo fisico desde el lado del hardware, ya que al instalar el S.O en una maquina virtual, el hardware disponible debera dividirse entre el equipo anfitrion y la maquina virtual.*

*Las incompatibilidades son otro punto, hay ciertos S.O que son dificiles de virtualizar porque requieren uso de comandos, y desbloqueando funciones ocultas. Incluso otros S.O que no permiten directamente la virtualizacion como CHRONOIS.*

*Programas para Virtualizar*

* *VirtualBox gratuito*
* *UMWare version paga*
* *Paralels exclusivo MC*
* *HiperUV exclusivo Microsoft*

### TERMINAL LINUX

**CONSOLA**: La interfaz de línea de comandos (o CLI (command line interface) es un método de comunicación entre usuario y maquina, que acepta instrucciones del usuario a través de líneas de texto (siguiendo reglas de sintaxis comprensibles por el S.O). La herramienta que posibilita esta función se demomina SHELL.

Tipos de SHELL:

* Bash (es el mas conocido al venir por defecto en varias distribuciones)
* Bourne Shell
* Korn Shell
* C Shell

Ejecucion de la consola:

La consola se inicia automáticamente en los niveles 1, 2, 3, y 4 en el S.O Linux

Iniciando en el nivel 5 puede variar el modo de ejecutar de acuerdo a la distribución instalada. En Ubuntu, podemos:

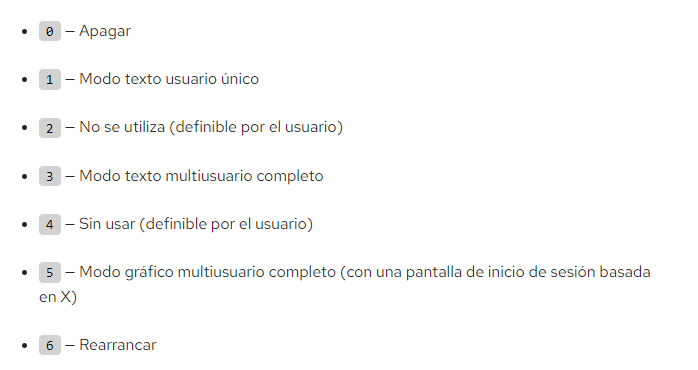
* Lanzando un TTY o espacio de trabajo sin entorno grafico

(falta completar)

### Niveles de ejecución

Las características de un nivel de ejecución dado determinan qué servicios init detenienen o inicia. Por ejemplo, el nivel de ejecución 1 (modo usuario único) detiene cualquier servicio de red, mientras que el nivel 3 arranca estos servicios. Asignando servicios específicos a ser detenidos o iniciados en un nivel dado, init puede fácilmente cambiar el modo de la máquina sin que el usuario tenga que arrancar o detener servicios manualmente.

Los siguientes niveles de ejecución se encuentran definidos por defecto bajo Red Hat Enterprise Linux:



**SUPERUSUARIO ROOT**

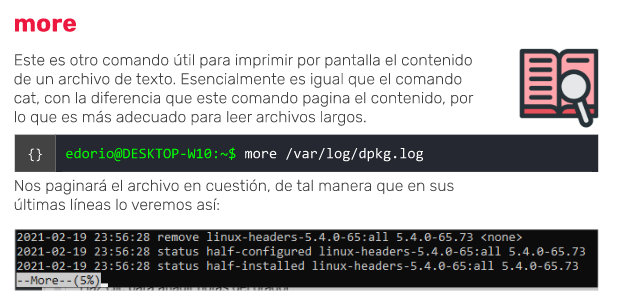
Linux separa la cuenta de un usuario común de la del super usuario a la que llamamos root, el cual tiene todos los privilegios y permisos para realizar acciones en el sistema.

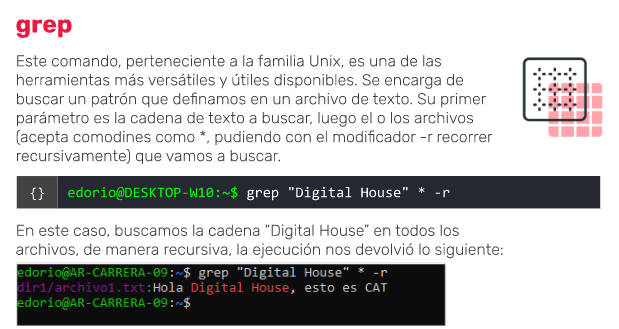
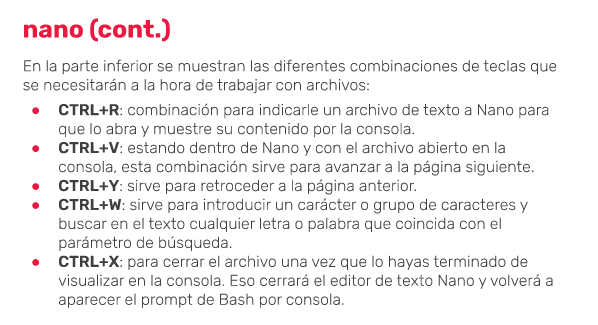
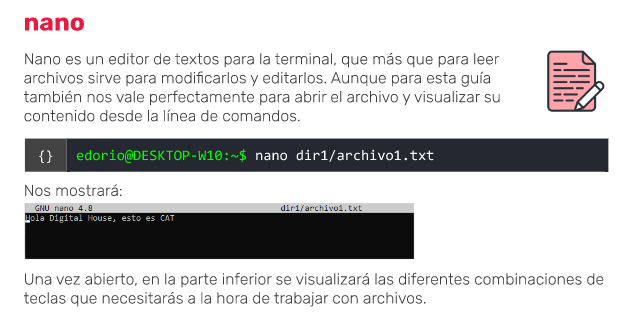
Hay ciertas acciones que el usuario creado por la persona no puede realizar, por ejemplo reiniciar un servicio. Eso devolvería que tenemos el permiso denegado. Para resolver esto es conveniente colocal el comando ´sudo´delante del comando que se quiere usar que necesita permiso del superusuario.

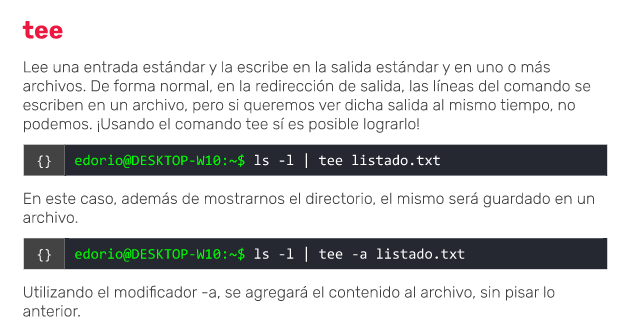
### Comandos útiles para la terminal Linux

|  |  |
| --- | --- |
| ls | Listar archivos y directorios de la carpeta en la que se esta situado. |
| ls -a | Muestra todo el contenido del directorio donde se encuentre en forma de lista. Incluye archivos y carpetas ocultos. |
| ls -l | Muestra el contenido del directorio donde se encuentra en forma de lista e incluye información referida al documento. |
| mkdir | Permite crear un directorio con el nombre y ruta especificada. |
| rmdir | Elimina el directorio que se le especifique. El directorio debe estar vacio. |
| rm | Elimina archivos sueltos y directorios que no estén vacios. |
| cp | Se usa para copiar archivos y directorios asi como ubicarlos en otras rutas definiendo primero el origen y luego destino. |
| mv | Permite mover archivos desde la consola. El comando es `mv ubicacion1/archivo.txt ubicacion2/archivo.txt. Tambien se puede usar para renombrar un archivo cuando las rutas son las mismas. |

### Comandos para la lectura de archivos de texto



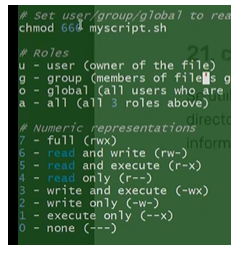




**Mas comandos útiles…** [Apuntes\Comandos básicos de Linux.docx](Apuntes/Comandos%20básicos%20de%20Linux.docx)

AUTOMATIZACION

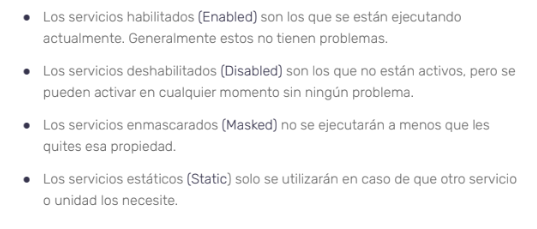
### SHELL SCRIPTING



SERVICIOS LINUX

### Para listar servicios Linux







### ADMINISTRAR SERVICIOS LINUX

### Tené en cuenta que cada servicio representa un software y cada uno de los softwares funciona de manera diferente. En este tutorial solo mostraremos cómo iniciar, verificar el estado y detener los servicios, es decir, los controles básicos.

### 

### 

### MAS COMANDOS:

### Contexto: OBTENER DATOS DESDE UN WEB SERVICE

### La terminal de Linux nos permite vincularla con un servidor web, obtener datos de ahí, y procesarlos para agregarlo a archivos en nuestro servidor, modificarlos y republicarlos. Por ejemplo obtener un JSON desde una URL externa, procesar su contenido, obtener el/los atributos que nos interesen y crear nuevos archivos, insertarlos en una base de datos.

### 

**El comando cURL**

### abreviatura de Client URL. Disponible en varios de los sistemas basados en UNIX. Su función es verificar la conectividad a las URL y se utiliza también para transferir datos.

Comando **JQ**

### Es un procesador JSON para la consola. El comando JQ se utiliza para transformar datos JSON en un formato más legible e imprimirlos en una salida estándar en Linux. El comando JQ se basa en filtros que se utilizan para buscar e imprimir solo los datos requeridos por un archivo JSON.

### 

### 

### COMBINAR EL USO DE COMANDOS CURL Y JQ

### PIPELINE: Funcion que permite utilizar la salida de un programa como entrada de otro.

### 

### 

### 

### AUTOMATIZACION

### SHELL DE UNIX

### BASH es una consola que interpreta las ordenes que un usuario hace al sistema ejecutado en una consola de unix. También puede leer y ejecutar ordenes desde un archivo llamado Script. Este Script permite hacer de manera persistente un listado de tareas a realizar por el S.O.

### BASH es una herramienta Open Source y pertenece al proyecto GNU y es un lenguaje de Scripting, lo que la hace muy potente para automatización de tareas y administración de sistemas

### SHEBANG --> sirve para indicarle al S:O que interprete debe usar el script.

### 

### 

### 

### Esta forma de generar ficheros con ordines la utilizaremos para que el sistema desarrolle tareas, ejecute procesos y:

### Realizar rutinas de mantenimiento de servidores

### Backups de datos

### Otorgamiento de permisos

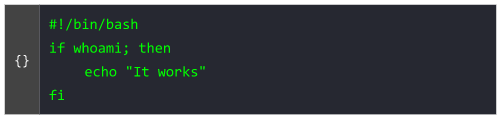
### Búsquedas

### Etc

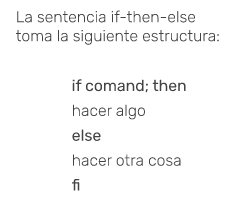
### 

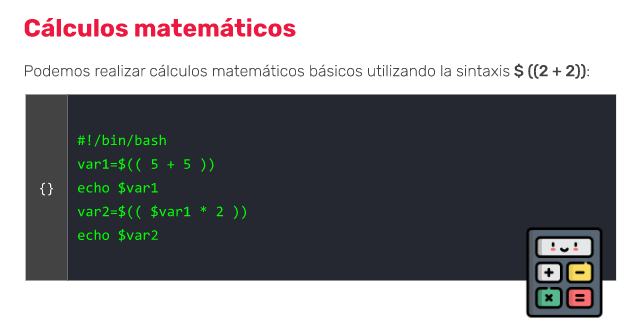
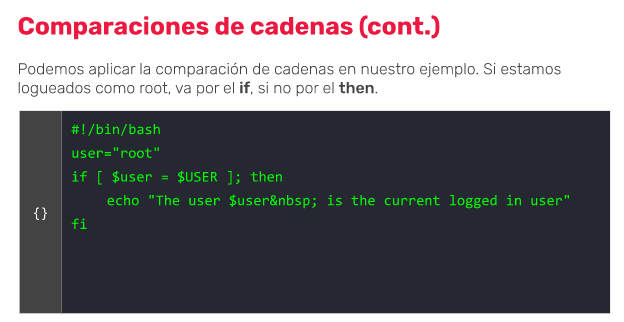
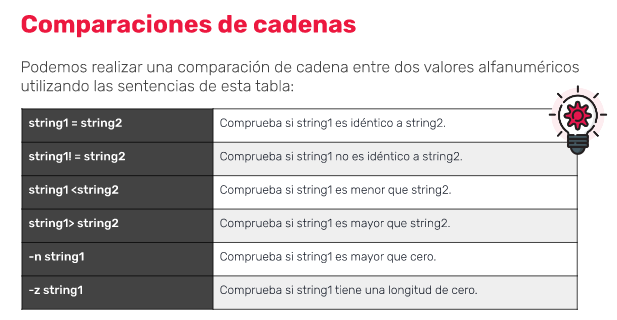
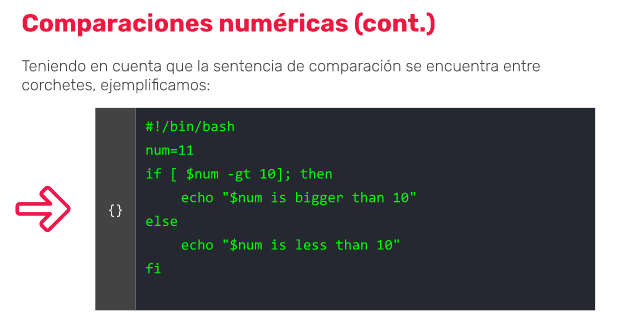
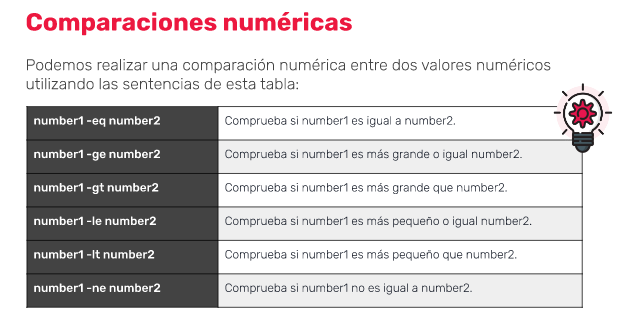
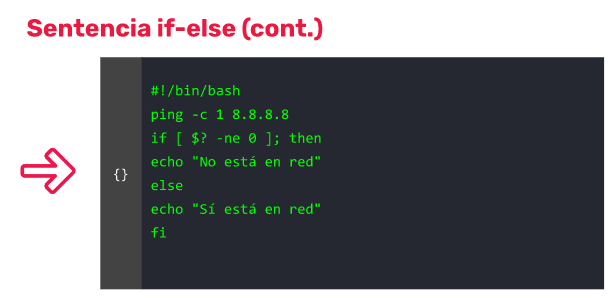
Estructuras de Control

* Sentencia if - then



* Sentencia if – else





POWERSHELL (Windows powersell)

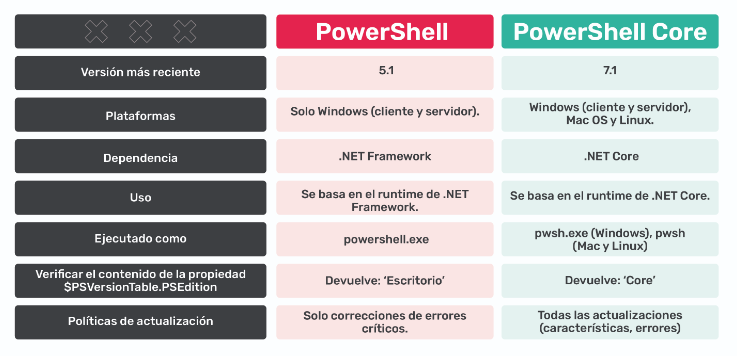
Interfas de línea de comandos (CLI) puede ejecutar scripts y facilita la configuración, administración, y automatización de tareas multiplataforma.

Funciona en Windows 10, Linux y MAC OS

Tiene una salida basada en objetos, por lo que acepta y devuelve objetos de .net fundamentado en .net clr. Esto es ideal para la automatización.

Su capacidad de trabajar con objetos es la diferencia principal con el resto de los otros CLI que solo devuelven textos.

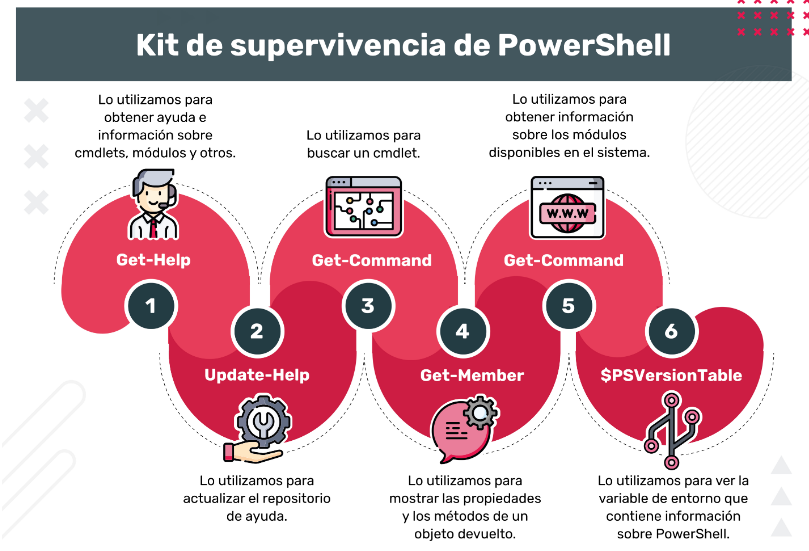
## **¿Qué es PowerShell Core? ¿Y cuál es la diferencia con Windows PowerShell?**



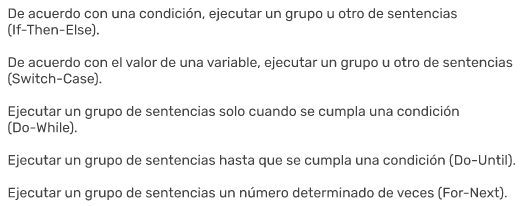
USOS COMUNES DE POWERSHELL

* Automatizacion de tareas: PowerShell sirve para facilitar a los administradores de sistemas las tareas de automatización, administración y configuración de sistemas.
* Configuration Management: Al utilizar PowerShell DSC.
* Monitoreo: Para escribir scripts que pueden ser utilizados por un software de monitoreo.
* Testear Infraestructura: Las pruebas de infraestructura en Pester son código de PowerShell que ejecuta el módulo Pester PowerShell y se crea de una manera específica, conocida como lenguaje específico de dominio (DSL). Este DSL describe el estado deseado y tiene el código necesario para verificar ese estado y comparar el resultado.
* Automatizacion de procesos: Automatizar tareas en un proceso de liberación de software (CI/CD).

Comandos CMDLETS 🡪 [Apuntes\glosariocmdlets V2.pdf](Apuntes/glosariocmdlets%20V2.pdf)



PIPELINE – VARIABLES – ESTRUCTURAS DE CONTROL – SCRIPTS Y FUNCIONES

1. PIPELINE: Son una serie de comandos conectados mediante operadores de canalización. 
2. VARIABLE: Unidad de mediden que se almacenan los valores. En powershell se representan mediante $. Ej.: $my\_var, $process
3. ESTRUCTURAS DE CONTROL: Permiten modificar el flujo de ejecusion de las instrucciones de un programa
4. SCRIPT DE POWERSHELL: Es un archivo de texto sin formato que tiene uno o mas comandos de powershell. Estos archivos tienen una extensión de tipo .ps1
5. FUNCION EN POWERSHELL: Es un conjunto de instrucciones a las que le damos un nombre. Lo importante de las funciones es que podemos llamarlas cuantas veces sea necesario sin tener que escribir las instrucciones nuevamente.

MODULOS EN POWERSHELL 🡪 [Apuntes\Definición Módulo de Powershell V2 - PDF.pdf](Apuntes/Definición%20Módulo%20de%20Powershell%20V2%20-%20PDF.pdf)

*Un modulo funciona en forma similar al gestor de paquetes APT de Linux*

*Se pueden descargar nuevos modulos de powershell con get-module*

PYTHON

**Python** es un **lenguaje de escritura rápida, escalable, robusto y de código abierto**. Esto hace que Python sea un aliado perfecto para la inteligencia artificial, ya que permite plasmar ideas complejas con unas pocas líneas de código, lo que no es posible con otros lenguajes.

PYTHON

Es legible por cualquier persona con conocimientos básicos de programación

Es util para desarrollo web, big data, inteligencia artificial, programación de videojuegos, finanzas y blockchange.

Es gratuito que hace que continuamente se estén desarrollando nuevas librerias, open source (de código abierto)

Es un lenguaje multiparadigma, combina propiedades de diferentes paradigmas de programación haciéndolo mas flexible y mas fácil de aprender

Es apto para todas las plataformas, se puede ejecutar en varios S.O.

Python necesita un interprete que permita ejecutar un programa o script escrito en Python sin necesidad de compilarlo.

* **Python** se define como un **“lenguaje de programación versátil, multiplataforma y multiparadigma que se destaca por su código legible y limpio”**.
* A su vez, se emplea en plataformas de alto tráfico como Google, YouTube o Facebook.
* Su principal objetivo es la automatización de procesos, lo que hará de las tareas algo mucho más simple.
* Uno de sus puntos fuertes es que **“comprueba los errores sobre la marcha”**. En este sentido, Python crea un código con gran legibilidad, que ahorra tiempo y recursos.

Python también se utiliza para:

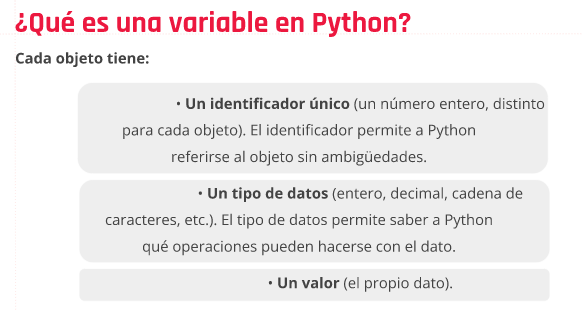
* obtener información de otros sitios web, lo que comúnmente se denomina “scraping”
* tiene aplicaciones prometedoras en el campo médico que mejoran la capacidad de brindar diagnósticos y tratamientos precisos a los pacientes
* Otras áreas como astronomía, robótica, vehículos autónomos, negocios, meteorología y desarrollo de interfaces gráficas de usuario también se benefician con el uso de Python.



Variables en PYTHON

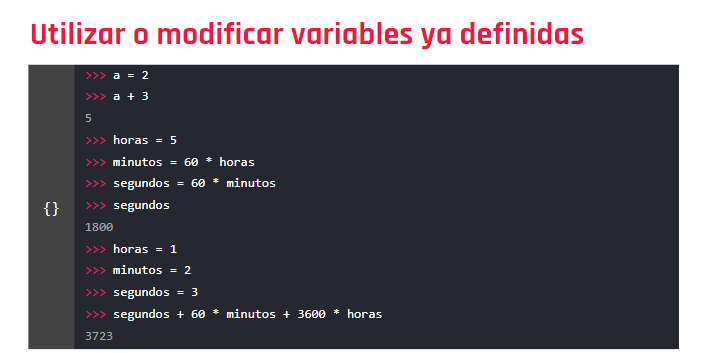
En algunos lenguajes de programación, las variables se pueden entender como "cajas" en las que se guardan los datos, pero en Python las variables son "etiquetas" que permiten hacer referencia a los datos (que se guardan en unas "cajas" llamadas objetos).

Python es un lenguaje de programación orientado a objetos y su modelo de datos también está basado en objetos. Para cada dato que aparece en un programa, Python crea un objeto que lo contiene.



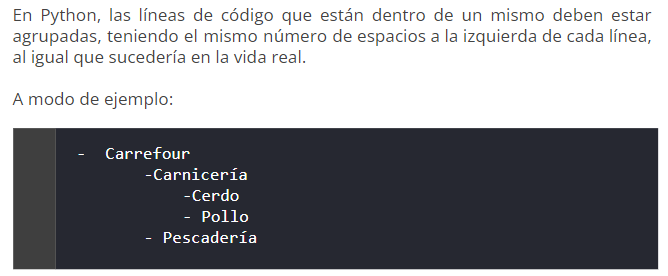
Las variables en Python se crean cuando se definen por primera vez, es decir, cuando se les asigna un valor por primera vez. Para asignar un valor a una variable se utiliza el operador de igualdad (=). A la izquierda de la igualdad se escribe el nombre de la variable y a la derecha el valor que se quiere dar a la variable.





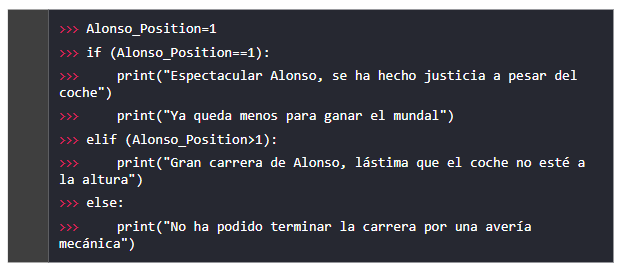
ESTRUCTURAS DE CONTROL EN PYTHON

1. Indentacion

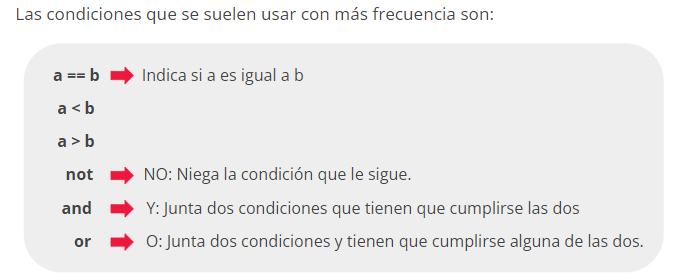


1. If

Esto se hace con el comando if (condición principal), con los opcionales elif (condiciones adicionales, se pueden poner tantas como se quiera) y else (si no se ha cumplido ninguna de las anteriores, solo se puede poner una vez y al final).



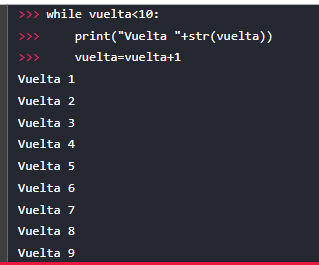
1. Condicionales



1. Bucles While y For

While: En ocasiones, tenemos que repetir varias veces una determinada tarea hasta conseguir nuestro objetivo.

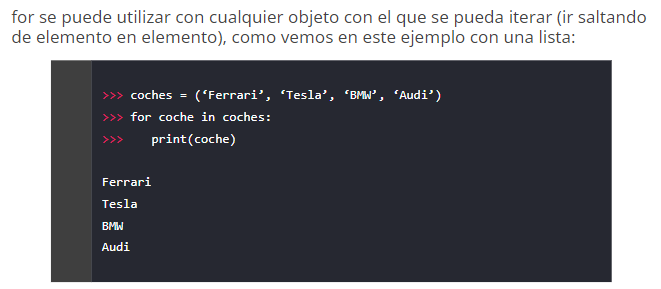
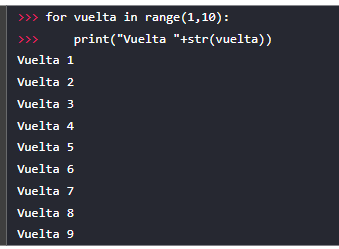
En Python esto se realiza con el comando while. Con los while, hay que tener la precaución de no realizar un «bucle infinito», que consiste en un bucle que nunca termina por un error en la programación. En el caso siguiente, esto ocurriría si no hubiéramos puesto la línea vuelta=vuelta+1.



For: En ocasiones, tenemos que repetir varias veces una determinada tarea hasta conseguir nuestro objetivo.

En Python esto se realiza con el comando for. En el caso del for, no es posible realizar un bucle infinito.

Como se puede ver en el siguiente ejemplo, range genera una secuencia de números desde 1 hasta 10. for se puede utilizar con cualquier objeto con el que se pueda iterar (ir saltando de elemento en elemento).



Si lo usamos con la función enumerate además enumerarà cada opción:

