

SERVICIOS EN RED



Servicios de Red



es un componente fundamental de la infraestructura de red que ofrece **funcionalidades específicas** para permitir la **comunicación, la colaboración y el acceso a recursos** en una red de computadoras

Algunas características de los servicios en red incluyen:

- 1. Acceso Remoto:** Permiten a los usuarios acceder a recursos o realizar tareas desde ubicaciones remotas a través de la red.
- 2. Compartición de Recursos:** Facilitan el uso compartido de dispositivos como impresoras, unidades de almacenamiento, etc., entre múltiples usuarios de la red.
- 3. Comunicación:** Proporcionan medios para la comunicación entre dispositivos y usuarios, como correo electrónico, mensajería instantánea, etc.
- 4. Gestión de Red:** Ayudan en la administración y supervisión de dispositivos de red, permitiendo la configuración, monitorización y mantenimiento de la red.
- 5. Seguridad:** Algunos servicios en red están dedicados a proporcionar medidas de seguridad, como cortafuegos, sistemas de detección de intrusiones, entre otros.

1. SSH (Secure Shell)

- Propósito: Acceso remoto seguro a sistemas.
- Funcionamiento: Permite la conexión segura a través de una terminal remota.
- Protocolo: SSH (Puerto 22 por defecto).

2. HTTP (Hypertext Transfer Protocol)

- Propósito: Servicio web para distribución de contenido.
- Funcionamiento: Permite la transferencia de datos en formato HTML.
- Protocolo: HTTP (Puerto 80 por defecto).

3. HTTPS (Hypertext Transfer Protocol Secure)

- Propósito: Versión segura del HTTP.
- Funcionamiento: Cifra la comunicación entre el cliente y el servidor.
- Protocolo: HTTPS (Puerto 443 por defecto).

4. DNS (Domain Name System)

- Propósito: Traduce nombres de dominio a direcciones IP.
- Funcionamiento: Resuelve consultas de nombres de dominio.
- Protocolo: DNS (Puertos 53 UDP/TCP).

5. DHCP (Dynamic Host Configuration Protocol)

- Propósito: Asignación automática de direcciones IP a dispositivos en una red.
- Funcionamiento: Distribuye direcciones IP, máscaras de red, puertas de enlace, etc.
- Protocolo: DHCP (Puerto 67/68 UDP).

6. FTP (File Transfer Protocol)

- Propósito: Transferencia de archivos entre sistemas.
- Funcionamiento: Permite la transferencia de archivos entre un cliente y un servidor.
- Protocolo: FTP (Puertos 20/21 por defecto).

7. SMTP (Simple Mail Transfer Protocol)

- Propósito: Envío de correos electrónicos.
- Funcionamiento: Transfiere correos electrónicos entre servidores de correo.
- Protocolo: SMTP (Puerto 25 por defecto).

8. POP3/IMAP (Post Office Protocol 3/Internet Message Access Protocol)

- Propósito: Recuperación de correos electrónicos.
- Funcionamiento: Permite a los clientes de correo electrónico recuperar mensajes de un servidor.
- Protocolo: POP3 (Puerto 110 por defecto), IMAP (Puerto 143 por defecto).

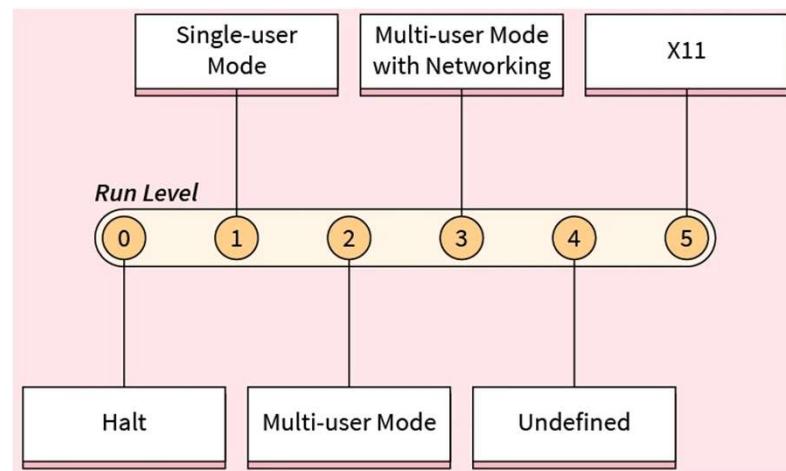
9. SNMP (Simple Network Management Protocol)

- Propósito: Monitoreo y gestión de dispositivos de red.
- Funcionamiento: Recopila información y permite la configuración remota de dispositivos de red.
- Protocolo: SNMP (Puerto 161/162 UDP).

10. NTP (Network Time Protocol)

- Propósito: Sincronización de relojes en sistemas de red.
- Funcionamiento: Ajusta la hora y la fecha de manera precisa en los sistemas.
- Protocolo: NTP (Puerto 123 UDP).

Run Levels



Los **run levels** (niveles de ejecución) en Linux son un sistema de categorización utilizado para definir los **estados de operación** del sistema.

Tradicionalmente, Linux ha utilizado **siete** run levels numerados del 0 al 6, aunque el número exacto y su función pueden variar ligeramente entre diferentes distribuciones.

Run levels más comunes:

Run Level 0: Este es el nivel de apagado. El sistema está detenido y listo para ser apagado de manera segura.

Run Level 1: También conocido como "modo de usuario único" o "modo de mantenimiento", este nivel es utilizado para tareas de mantenimiento y recuperación. Solo se inicia un número mínimo de servicios.

Run Level 2: Similar al nivel de ejecución 3, pero sin soporte para la red.

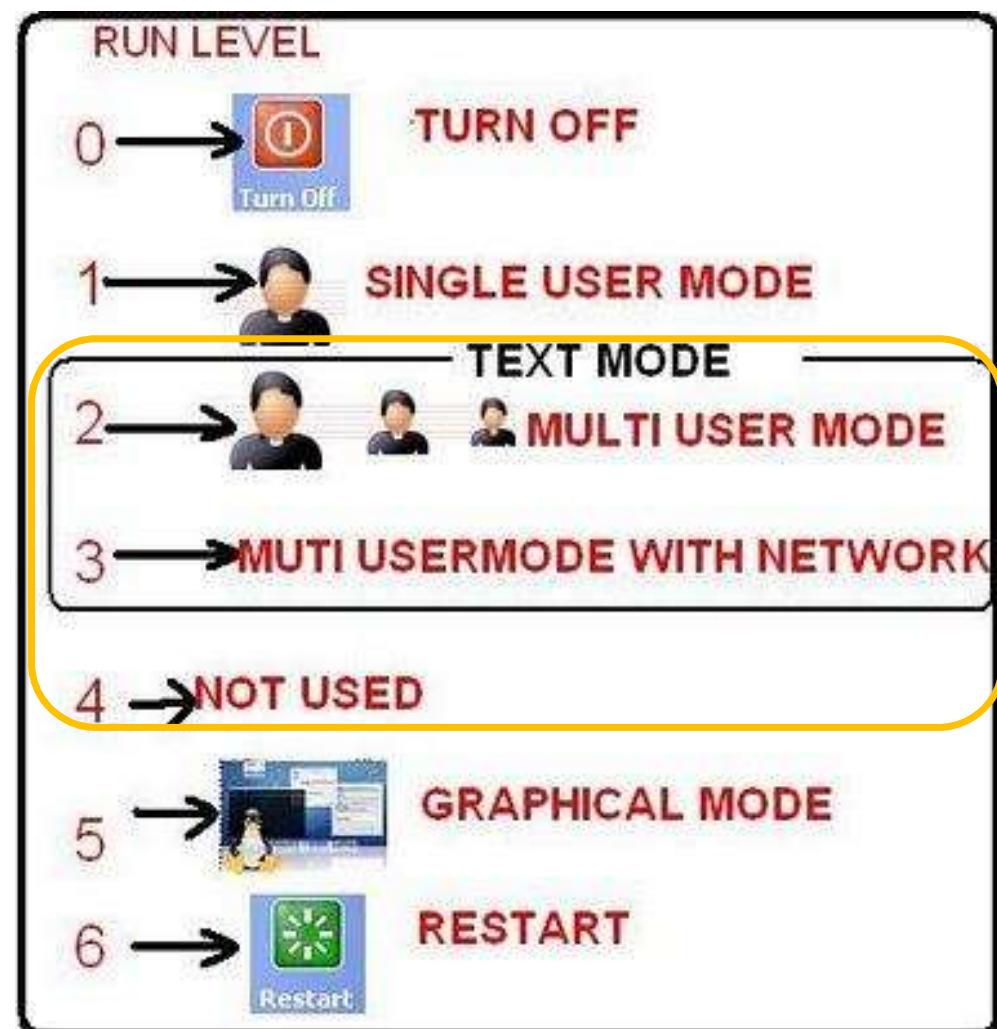
Run Level 3: Modo de usuario múltiple (sin interfaz gráfica). En este nivel, el sistema está completamente funcional con soporte de red y normalmente se utiliza en servidores.

Run Level 4: Tradicionalmente no está definido por defecto en la mayoría de las distribuciones de Linux y se deja para personalización del usuario. A menudo se usa para implementar un nivel de ejecución personalizado.

Run Level 5: Similar al nivel de ejecución 3, pero con una interfaz gráfica. Este nivel se utiliza comúnmente en sistemas de escritorio.

Run Level 6: Este es el nivel de reinicio. El sistema se reinicia cuando se cambia a este nivel. ¹⁰⁷

Run Level	Name	Description
0	<i>Halt</i>	Shuts down all services when the system will not be rebooted.
1	<i>Single User</i>	Used for system maintenance. No Networking capabilities.
2	<i>MultiUser No Network Support</i>	Used for maintenance and system testing.
3	<i>MultiUser Network Support</i>	Non-Graphical Text Mode operations for server systems.
4	-	Custom Mode, used by SysAdmin
5	<i>Graphical X11</i>	Graphical login with same usability of Run Level 3.
6	<i>Reboot</i>	Shuts down all services when the system is being rebooted.



Sistema de gestión de servicios **systemd**

Actualmente se ha adoptado el **sistema de gestión de servicios systemd**, que proporciona un enfoque diferente para el manejo de los estados del sistema, aunque aún se puede utilizar el concepto de run levels de forma simbólica.

“Sistema de **inicialización** y **gestión** de **servicios** que se ha convertido en el **estándar** en muchas distribuciones de Linux modernas”

En general, **systemd** ofrece varias mejoras sobre el sistema de inicialización tradicional init:

- arranque más rápido
- control más preciso sobre los servicios
- gestión más eficiente de los recursos del sistema

Conceptos clave de **systemd**:

1. Unidad (Unit): En systemd, todo es una unidad

- unidades de servicios (service units) *.service*
- unidades de sockets (socket units) *.socket*
- unidades de dispositivos (device units) *.device*
- unidades de montaje (mount units)... *.mount*

- ❖ Cada unidad es un archivo de configuración que describe un componente del sistema y cómo debe ser gestionado.

2. Servicio (Service):

- Las unidades de servicio son las más comunes.
- Representan un **proceso o aplicación** que systemd puede iniciar, detener y gestionar
- Cada servicio tiene un **archivo de configuración** con la extensión `.**service**`, que define cómo se debe manejar el servicio, incluyendo el comando para iniciar el servicio, las dependencias, el usuario bajo el cual se debe ejecutar, entre otros.

3. **Targets**: Los targets son grupos de unidades que representan **estados** específicos del sistema, similar a los run levels en el antiguo sistema init.

Por ejemplo

- `multi-user.target` es similar al run level 3 en sistemas basados en SysV init
- `graphical.target` es similar al run level 5.

runlevel(8) — Linux manual page

[NAME](#) | [SYNOPSIS](#) | [OVERVIEW](#) | [DESCRIPTION](#) | [OPTIONS](#) | [EXIT STATUS](#) | [ENVIRONMENT](#) |
[FILES](#) | [SEE ALSO](#) | [COLOPHON](#)

Search online pages

RUNLEVEL(8)

runlevel

RUNLEVEL(8)

NAME [top](#)

runlevel - Print previous and current SysV runlevel

SYNOPSIS [top](#)

runlevel [options...]

OVERVIEW [top](#)

"Runlevels" are an [obsolete way to start and stop groups of services used in SysV init.](#) [systemd](#) provides a compatibility layer that maps runlevels to targets, and associated binary like **runlevel**. Nevertheless, only one runlevel can be "active" at a given time, while [systemd](#) can activate multiple targets concurrently, so the mapping to runlevels is confusing and approximate. Runlevels should not be used in new code, and mostly useful as a shorthand way to refer the matching system targets in kernel boot parameters.

Table 1. Mapping between runlevels and systemd targets

Runlevel	Target
0	poweroff.target
1	rescue.target
2, 3, 4	multi-user.target
5	graphical.target
6	reboot.target

Table 1. Mapping between runlevels and systemd targets

Runlevel	Target
0	poweroff.target
1	rescue.target
2, 3, 4	multi-user.target
5	graphical.target
6	reboot.target

4. **Systemctl**: Es la herramienta de línea de comandos principal para interactuar con systemd.

Se utiliza para controlar los servicios, ver su estado, iniciar y detener unidades, habilitar o deshabilitar unidades para que se inicien automáticamente al arrancar el sistema, entre otras tareas.

SERVICIOS EN RED



GESTIÓN DE LOS SERVICIOS

Para **gestionar los servicios**, en Ubuntu se utiliza el comando

Systemctl

Inicio y Parada de Servicios:

- **systemctl start nombre_servicio**
- **systemctl stop nombre_servicio**
- **systemctl restart nombre_servicio**
- **systemctl reload nombre_servicio**

(Recarga la configuración de un servicio sin reiniciarlo)

Activación y Desactivación de Servicios en el arranque del sistema:

- **systemctl enable nombre_servicio**
- **systemctl disable nombre_servicio**

Consultar el Estado de los Servicios:

- **systemctl status nombre_servicio**

Listar Todos los Servicios:

- **systemctl list-units --type=service**

Fijar acceso a un “runlevel”:

```
systemctl set-default graphical.target
```

Gestionar la energía – Apagado y reinicio del sistema:

\$ systemctl reboot

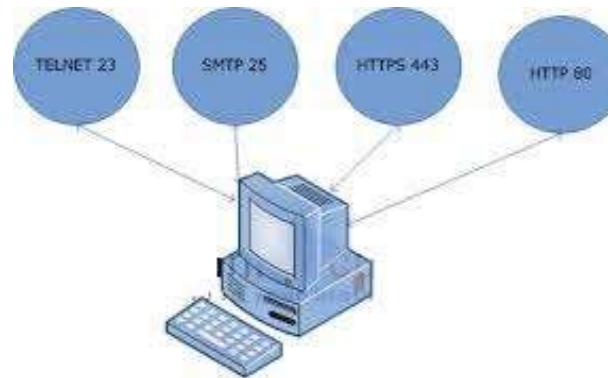
\$ systemctl poweroff

\$ systemctl hibernate

\$ systemctl suspend

Para **abrir y cerrar PUERTOS**, en Ubuntu se utiliza el comando:

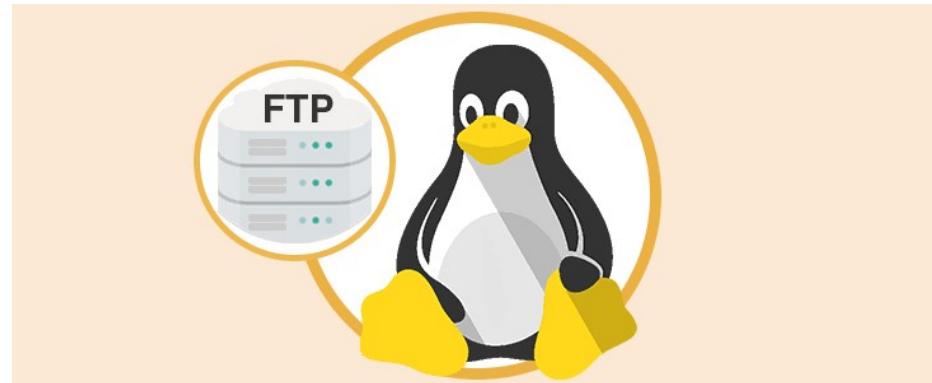
ufw



sudo ufw allow número_puerto

sudo ufw deny número_puerto

TRANSFERENCIA DE ARCHIVOS



FTP – File Transfer Protocol

ACCESO REMOTO



FTP - “File Transfer Protocol”

Funcionamiento: transferir archivos entre computadoras a través de una red.

FTP utiliza dos canales de comunicación:

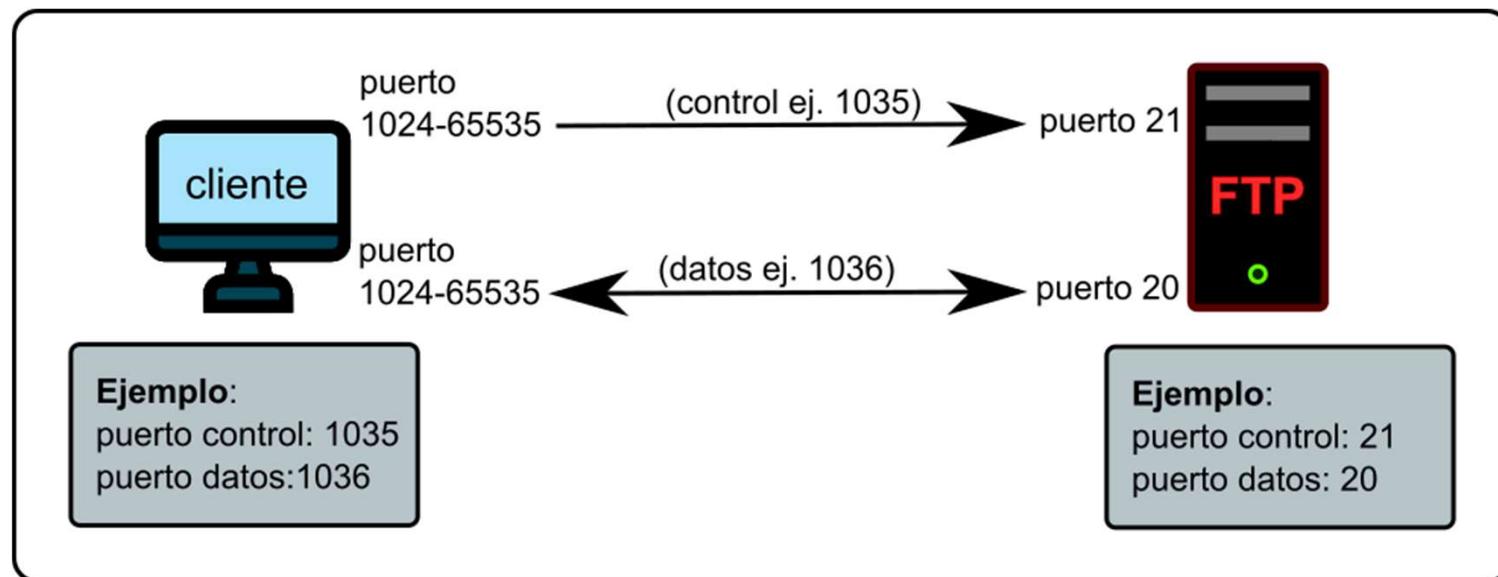
- *El canal de control maneja la comunicación entre el cliente y el servidor para comandos de control (iniciar sesión, cambiar directorios, solicitar archivos...)*
- *El canal de datos, para transferir los propios archivos.*

FTP – PUERTOS:



- **Puerto 21:** para conexiones estándar (canal de control)
- **Puerto 20:** para conexiones de datos

El cliente se conecta al servidor utilizando el **puerto 21** para el **canal de control**, por donde el cliente puede enviar **comandos** al servidor para realizar acciones como *listar archivos, subir o bajar archivos, y crear o eliminar directorios*



Cuando se *inicia una transferencia de archivos*, se *abre un segundo canal de datos*, por defecto en el **puerto 20**



Guía básica para comenzar a usar FTP (File Transfer Protocol)

1. Instalación del servidor FTP

2. Configuración del servidor FTP

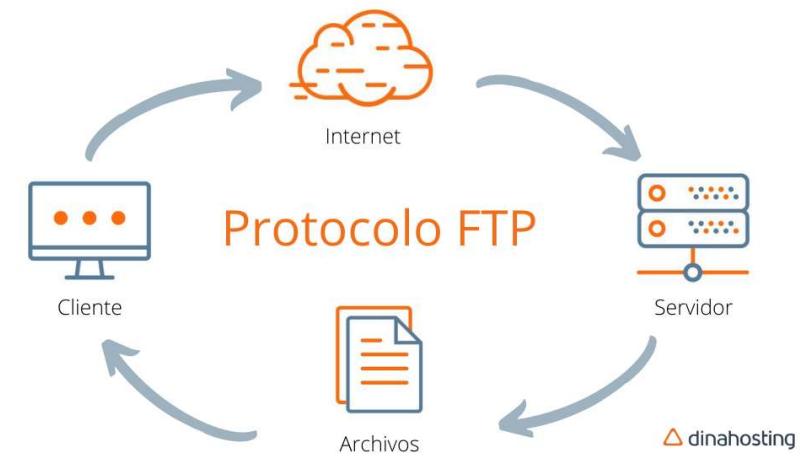
3. Configuración de usuarios

4. Reinicio del servidor FTP

5. Configuración del firewall

6. Acceso al servidor FTP desde un cliente

7. Transferencia de archivos





1. Instalación del servidor FTP

uno de los más comunes es vsftpd (Very Secure FTP Daemon).

```
sudo apt-get install vsftpd
```

2. Configuración del servidor FTP

Configurar los ajustes en el archivo de configuración principal **'/etc/vsftpd.conf'**

```
sudo nano /etc/vsftpd.conf
```

- habilitar/deshabilitar opciones como el acceso anónimo
- uso de conexiones cifradas (SSL/TLS)
- puertos que escucha el servidor
- establecer directorios raíz para los usuarios, etc



3. Configuración de usuarios

- Puedes crear usuarios específicos para FTP y configurar sus permisos de acceso.
- Los usuarios de FTP pueden ser usuarios del sistema o usuarios virtuales específicos para FTP.
- Descomenta o agrega las siguientes líneas, para que cada usuario acceda a su directorio de inicio cuando se conecte al servidor FTP :

```
user_sub_token=$USER  
local_root=/home/$USER
```

4. Reinicio del servidor FTP



```
sudo systemctl restart vsftpd
```

5. Configuración del firewall

El puerto FTP (por defecto, el puerto 21 para conexiones FTP estándar y el puerto 20 para conexiones de datos) tiene que estar abierto para permitir el tráfico de FTP

```
sudo ufw allow 21/tcp
```

6. Acceso al servidor FTP desde un cliente

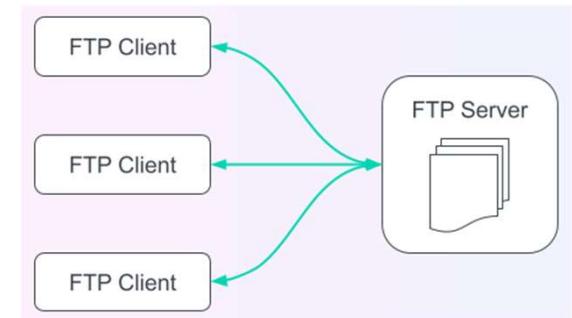


En sistemas Linux, puedes usar el comando `ftp` en la línea de comandos o un cliente FTP gráfico como FileZilla.

ftp dirección_ip_o_nombre_de_dominio

7. Transferencia de archivos

- `put` para cargar archivos en el servidor
- `get` para descargar archivos del servidor



Recuerda revisar la documentación oficial de vsftpd para obtener más información sobre configuraciones avanzadas y medidas de seguridad adicionales.



FTP carece de cifrado por defecto, lo que significa que los datos, incluyendo las credenciales de inicio de sesión, se envían en texto plano.

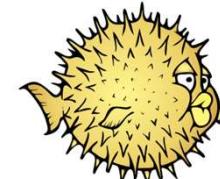
Por esta razón, se recomienda utilizar variantes seguras del protocolo FTP, como **FTPS (FTP Seguro)** o **SFTP (Protocolo de Transferencia de Archivos SSH)**, que cifran la comunicación para mayor seguridad.

ACCESO REMOTO



SSH – Secure Shell

ACCESO REMOTO

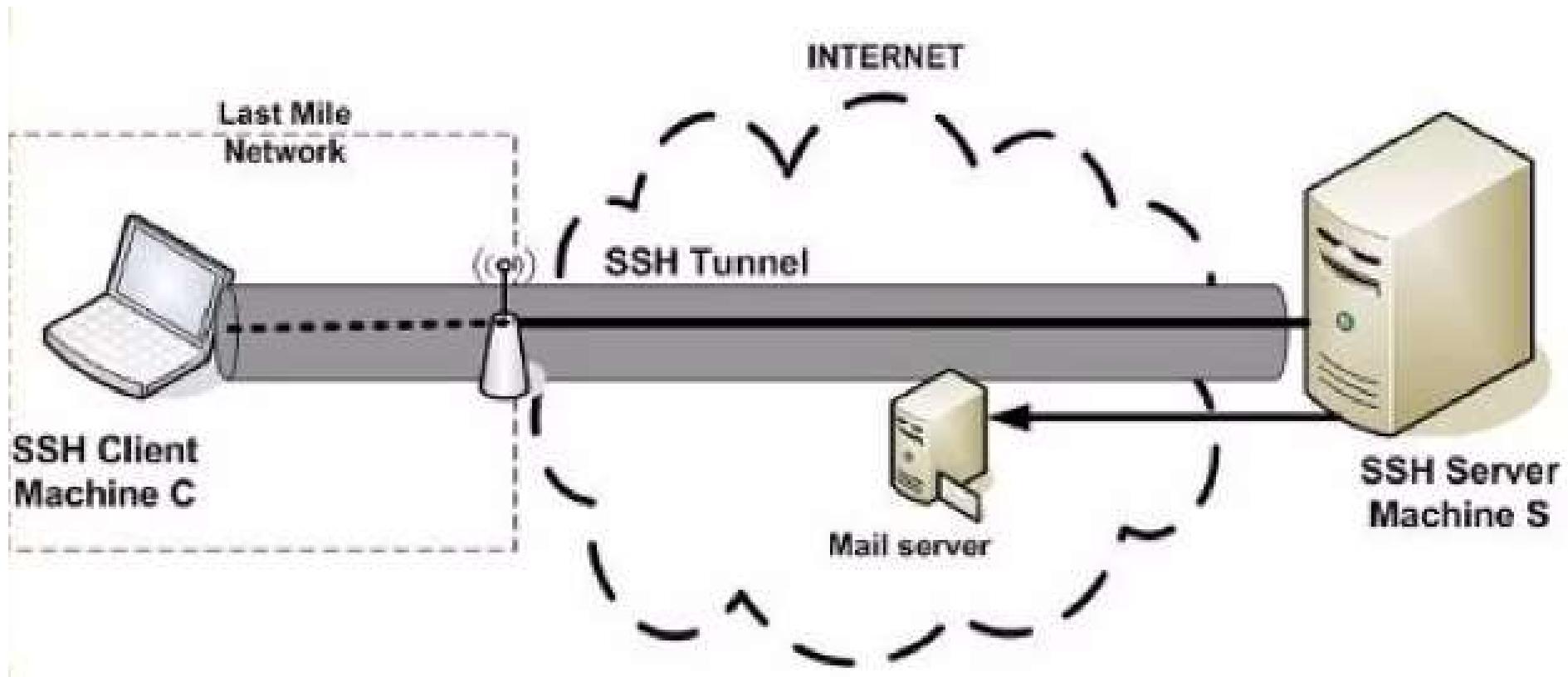
SSH**OpenBSD**

“Secure Shell” – Línea de comandos segura

Funcionamiento: para acceder y administrar de forma segura sistemas remotos a través de una conexión encriptada.

Ejemplos
de uso

- Administración de servidores de forma remota
- Transferir archivos de manera segura
- Ejecutar comandos en una máquina remota





SSH – PUERTOS:

- **Puerto 22:** puerto predeterminado* para establecer la conexión segura entre computadoras

*Si es necesario este puerto puede cambiarse en la configuración del servidor SSH.



Guía básica para comenzar a usar SSH (Secure Shell) :

1. Instalación del servidor SSH
2. Configuración del servidor SSH
3. Reinicio del servidor SSH
4. Acceso al servidor SSH desde otro dispositivo (cliente ssh)
5. Autenticación y seguridad



1. Instalación del servidor SSH:

```
sudo apt-get install openssh-server
```

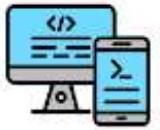
Una vez instalado, el servidor SSH comenzará a ejecutarse automáticamente

Comprobación

```
$ systemctl status sshd
```

```
$ ps aux | grep sshd
```

```
$ netsat -atun | grep 22
```



2. Configuración del servidor SSH:

- Puedes configurar opciones adicionales editando el archivo de configuración principal, que generalmente se encuentra en **'/etc/ssh/sshd_config'**.
 - ❑ el **puerto** predeterminado (22) en el que escucha el servidor
 - ❑ el **acceso** de usuarios, restringiendo el acceso de usuarios o grupos
 - ❑ la **autenticación** (contraseña, clave pública, certificados,...)
 - ❑ configuración de **parámetros** relacionados con el intercambio de claves y cifrado

Ejemplo básico de configuración del archivos `sshd_config`

```
# Puerto en el que escucha el servidor SSH  
Port 22  
# Archivo de clave pública para la autenticación  
# de clave pública  
PubkeyAuthentication yes  
# Permitir autenticación con contraseña  
PasswordAuthentication no  
# Permitir acceso solo a ciertos usuarios  
AllowUsers usuario1 usuario2  
# Rutas permitidas para la autenticación con  
# clave pública  
AuthorizedKeysFile .ssh/authorized_keys
```





3. Reinicio del servidor SSH:

```
sudo systemctl restart ssh
```

4. Acceso al servidor SSH desde otro dispositivo (cliente ssh):

- En sistemas **Linux y macOS**, puedes usar el terminal y el comando `ssh` seguido de la dirección IP o nombre de dominio del servidor y el nombre de usuario:

```
ssh usuario@dir_ip_o_nombre_dominio
```

- En sistemas **Windows**, puedes usar programas como **PuTTY** o **PowerShell** para conectarte al servidor SSH.
- *Puedes ejecutar comandos shell de la misma manera que lo harías si estuvieras operando físicamente el equipo remoto*

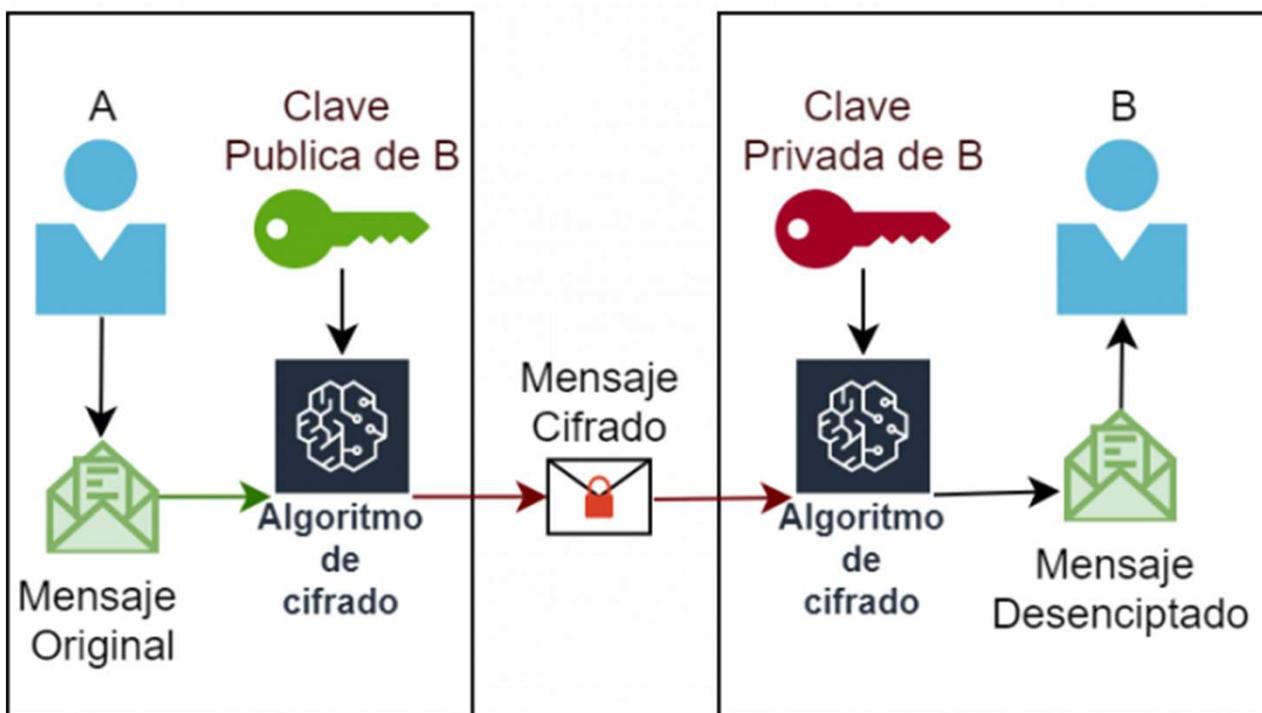


5. Autenticación y seguridad:

SSH utiliza autenticación basada en **clave pública/privada** de forma predeterminada, lo que proporciona una mayor seguridad en comparación con la autenticación basada en contraseñas.

Puedes generar un par de claves SSH en tu cliente y copiar la clave pública al servidor para habilitar la autenticación sin contraseña.

Recuerda revisar la documentación oficial de SSH para obtener más detalles sobre configuraciones avanzadas y medidas de seguridad adicionales.



Clave
Publica de B



Cualquiera que lo desee
pude obtenerla, es capaz de cifrar
contenido para desencriptar
con su clave privada asociada

Clave
Privada de B



Es capaz de desencriptar mensajes
que han sido cifrados con su clave
publica asociada, solo B la conoce

<https://iberasync.es/clave-simetrica-y-asimetrica/>

https://es.wikipedia.org/wiki/Criptograf%C3%A1_a_ssim%C3%A9trica



Generar un par de claves pública/privada para usar con SSH

1. ssh-keygen -t rsa -b 2048

- generará un par de claves RSA de 2048 bits
- otros algoritmos: -t ed25519 o -t ecdsa

2. Especificar la ubicación donde guardar la clave (o aceptar por defecto)

3. Ingresar una frase de paso (passphrase) para proteger tu clave privada (o dejarlo en blanco)



Generar un par de claves pública/privada para usar con SSH

❖ Se generarán **dos archivos**:

- uno para la clave privada (generalmente llamado **id_rsa**)
 - otro para la clave pública (generalmente llamado **id_rsa.pub**)
-
- ❖ La clave privada debe mantenerse segura y nunca debe ser compartida, mientras que la clave pública puede ser compartida con los servidores a los que deseas acceder.



Generar un par de claves pública/privada para usar con SSH

4. Copiar la clave pública en el servidor

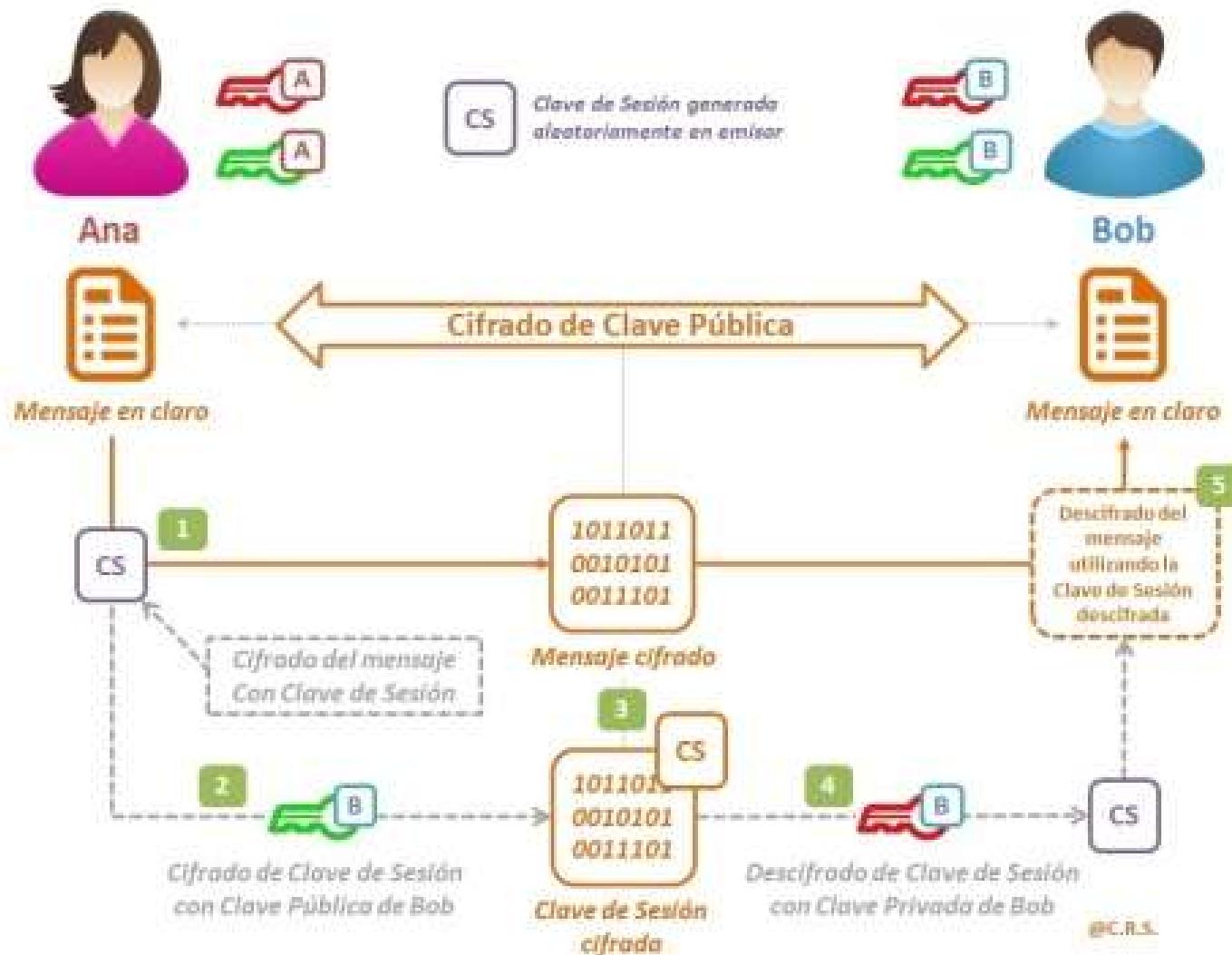
```
ssh-copy-id -i dir_origen usuario@dir_ip
```

- ❖ Nos pedirá el password por última vez al copiarla

5. Ver la clave pública en el servidor

```
user@equipo:/etc/ssh$
```

```
sudo ssh-keygen -lv -f /etc/ssh/ssh_host_rsa_key
```



FASES



- 1: NEGOCIACIÓN**
 - 2: AUTENTICACIÓN**
-

FICHEROS

En cliente: **ls -l ~/.ssh/**

En servidor: **ls -l /etc/ssh/**



Actividad

P21 – FTP y SSH