Crashcourse selfhosting

Esta charla requiere de acceso a un entorno remoto preparado, puedes pedirlo por Signal.

Gracias :)

SSH

```
ssh -p 20022 root@chipburners.club
    puerto   usuario      host
```

Package manager

- Instalar un paquete → `dnf install paquete`
- Actualizar repositorios → `dnf update`
- Actualizar paquetes → `dnf upgrade`

Package manager

- Instalar un paquete → `dnf install paquete`
- Actualizar repositorios → `dnf update`
- Actualizar paquetes → `dnf upgrade`

`dnf` es el package manager de **Fedora**:

- en sistemas Debian (o Ubuntu) → `apt`
- … Arch → `pacman`
- etc.

Caddy

HTTP daemon multifunción:

- Terminación SSL/TLS
- Archivos estáticos
- Proxy reverso

Caddy

HTTP daemon multifunción:

- Terminación SSL/TLS
- Archivos estáticos
- Proxy reverso

Alternativas:

- Apache
- Nginx
- Haproxy

Caddy

HTTP daemon multifunción:

- Terminación SSL/TLS
- Archivos estáticos
- Proxy reverso

Alternativas:

- Apache
- Nginx
- Haproxy

Instalación

En Fedora, paquete `caddy`

Configuración

`/etc/caddy/Caddyfile`

FHS

- /usr
  - bin → binarios (ejecutables)
  - lib → librerías
  - share → documentación, ejemplos, ...
- /etc → configuración
- /var → datos persistentes de servicios

FHS

- /usr
  - bin → binarios (ejecutables)
  - lib → librerías
  - share → documentación, ejemplos, ...
- /etc → configuración
- /var → datos persistentes de servicios

- /opt → programas externos
- /home o /root → carpetas de usuario (o root)

Service manager

Funciones:

- Iniciar servicios en arranque
- Dependencias de servicios (e.g. red)
- Estado de servicios

Service manager

Funciones:

- Iniciar servicios en arranque
- Dependencias de servicios (e.g. red)
- Estado de servicios

En el caso de **systemd**:

- Logging unificado (journalctl)
- Sandboxing
- init (PID 1)

Service manager

Control servicios → `systemctl`:

- Iniciar → `systemctl start miservicio`
- Parar → `systemctl stop miservicio`
- Estado → `systemctl status miservicio`
- Activar en arranque → `systemctl enable miservicio`

Ejemplos:

```
systemctl status                journalctl -u caddy
systemctl start caddy           systemctl restart ssh
```

SSH Tunneling

Permite exponer puertos del servidor al cliente (**local**) o puertos del cliente al servidor (**remoto**)

```
ssh -L 8080:localhost:80 user@mihost
```

	local	puerto cliente	host expuesto	puerto expuesto

Resultado: localhost:8080 → mihost:80 (`localhost` es desde el punto de vista del servidor al que nos conectamos)

DNS

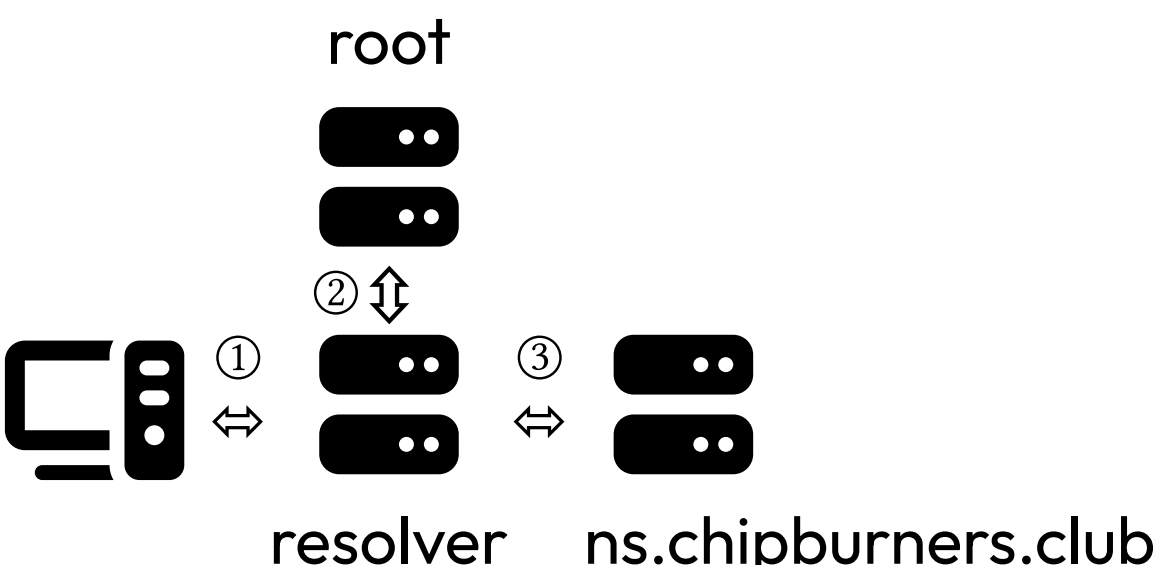
Key/value store distribuido para resolución de hostnames

chipburners.club → A 79.43.123.3

DNS

Key/value store distribuido para resolución de hostnames

chipburners.club → A 79.43.123.3



Zone file

```
$TTL 1m
$ORIGIN change.chipburners.club.

@ IN SOA ns1.chipburners.club. admin.chipburners.club. (
  2025022602     ; serial
  3H             ; refresh
  1800           ; retry
  1W             ; expire
  1H )           ; minimum

@ IN NS ns1.chipburners.club.
ns1.chipburners.club IN A 93.177.67.23
ns1.chipburners.club IN AAAA 2a03:4000:38:59b::1

fotos IN A 77.203.67.41
```

Caddyfile

Conjunto de directivas según el sitio visitado

```
fresa.chipburners.club {
  file_server
  root * /var/www/fresa
  reverse_proxy /api/* :2020
}

photos.chipburners.club {
  reverse_proxy :4043
}
```

Caddyfile

Conjunto de directivas según el sitio visitado

```
fresa.chipburners.club {
  file_server
  root * /var/www/fresa
  reverse_proxy /api/* :2020
}

photos.chipburners.club {
  reverse_proxy :4043
}
```

Documentación

https://caddyserver.com/docs

Contenedores OCI

- Formato distro-agnostic para programas
- Sandboxing por defecto (como una VM, pero comparte kernel)
- Formado por capas (“diffs” apilados)
- Un “engine” administra los contenedores (Docker/**Podman**/K8s)

Ejemplos

```
podman run -it alpine:latest   podman ps
podman stop a329540fd9c5       podman build --tag myapp
podman exec -it banana bash
```

(Docker) Compose

Formato simple para orquestrar múltiples contenedores para una aplicación; archivo `docker-compose.yml`:

```yaml
name: immich
services:
  immich-server:
    image: immich-server:release
    ports:
      - '2283:2283'
    depends_on:
      - database
```

```yaml
# Continuando services:
  database:
    image: postgres:14
    environment:
      POSTGRES_PASSWORD: mypw
      POSTGRES_USER: postgres
    volumes:
      - ./postgres:/var/lib/
postgresql/data
```

(Docker) Compose

- `podman-compose up -d` (-d arranca los contenedores en 2o plano)
- `podman-compose down`

Ejemplo → Immich

https://chipburners.club/docker-compose.yml

1. Arrancar contenedores
2. Configuración inicial con SSH tunneling
3. Exponer por HTTPS en el subdominio fotos

https://chipburners.club