Scripts útiles

Scripts que me ahorran tiempo, clics y neuronas. Si algo puede automatizarse, aquí lo dejo apuntado.

Conectividad

Conectividad

Construcción de Caddy con xcaddy y módulos personalizados

Introducción

Este artículo documenta la **construcción personalizada de Caddy** mediante **xcaddy**, incorporando únicamente los módulos necesarios para el uso habitual en la infraestructura. El objetivo es disponer de un binario controlado, reproducible y alineado con las necesidades reales, evitando dependencias externas o configuraciones parciales.

El proceso se automatiza mediante un script que gestiona la compilación, la instalación del binario resultante, la validación de la configuración y el reinicio del servicio. Este enfoque permite actualizar Caddy de forma coherente, sin instalaciones manuales ni pasos intermedios frágiles.

¿Por qué usar xcaddy?

Caddy no incluye todos los módulos posibles en su binario oficial. Determinadas funcionalidades, como el **DNS challenge con proveedores externos** o módulos HTTP específicos, requieren que el binario se compile explícitamente con ellos.

`xcaddy` es la herramienta oficial que permite:

- Construir un binario único de Caddy con módulos adicionales.
- Mantener control total sobre qué capacidades se incluyen.
- Evitar binarios genéricos con dependencias innecesarias.
- Reproducir el mismo binario en distintos entornos.

Este enfoque es especialmente útil cuando Caddy actúa como **pieza central del frontend HTTPS**.

Características del enfoque

- Compilación automatizada mediante `xcaddy`.
- Inclusión explícita de los módulos necesarios.
- Sustitución directa del binario de Caddy existente.
- Validación y formateo del `Caddyfile`.
- Reinicio controlado del servicio y comprobación de estado.

Módulos incluidos

El binario se construye con los siguientes módulos:

- **DNS Cloudflare**: permite realizar validaciones ACME mediante DNS challenge, necesario cuando no se exponen puertos públicos o se quiere evitar HTTP-01.
- **transform-encoder**: se utiliza para transformar y personalizar el formato de logs, facilitando formatos compatibles con nginx u otros sistemas de análisis.

El script actual **no incluye** módulos adicionales; cualquier ampliación debe declararse explícitamente en la fase de build.

Requisitos previos

Para ejecutar el script de construcción es necesario:

- Go instalado en el sistema.
- `xcaddy` disponible en el entorno.
- Caddy gestionado como servicio mediante `systemd`.

Si alguno de estos elementos falta, la compilación no podrá completarse.

Instalación manual de Go

Se instala Go de forma manual para asegurar la versión utilizada durante la compilación:

```
sudo rm -rf /usr/local/go && sudo tar -C /usr/local -xzf go1.23.3.linux-amd64.tar.gz
export PATH=$PATH:/usr/local/go/bin
echo 'export PATH=$PATH:/usr/local/go/bin' >> ~/.bashrc
source ~/.bashrc
```

```
go version
```

Este método evita depender de versiones empaquetadas potencialmente desactualizadas.

Instalación de xcaddy desde el repositorio oficial

```
sudo apt install -y debian-keyring debian-archive-keyring apt-transport-https
curl -1sLf 'https://dl.cloudsmith.io/public/caddy/xcaddy/gpg.key' | sudo gpg --dearmor -o
/usr/share/keyrings/caddy-xcaddy-archive-keyring.gpg
curl -1sLf 'https://dl.cloudsmith.io/public/caddy/xcaddy/debian.deb.txt' | sudo tee
/etc/apt/sources.list.d/caddy-xcaddy.list
sudo apt update && sudo apt install xcaddy
xcaddy version
```

La instalación desde el repositorio oficial asegura compatibilidad con las versiones actuales de Caddy.

Script de construcción

La lógica completa de compilación, instalación y validación se encuentra en el siguiente repositorio:

- UpdateCaddy.sh

El script realiza las siguientes acciones, de forma secuencial:

- Construye Caddy mediante `xcaddy build`, incluyendo únicamente los módulos definidos.
- Verifica que el binario generado responde correctamente (`caddy --version`).
- Reemplaza el binario existente en `/usr/bin/caddy` sin interacción.
- Ejecuta el formateo del `Caddyfile` mediante `caddy fmt`.
- Reinicia el servicio de Caddy gestionado por `systemd`.
- Comprueba que el servicio queda activo tras el reinicio.

El script está diseñado para fallar de forma temprana si ocurre cualquier error crítico durante el proceso, evitando dejar el sistema en un estado inconsistente.

Errores comunes y decisiones importantes

- Si Go o `xcaddy` no están disponibles, la compilación falla.
- El servicio de Caddy debe estar gestionado por `systemd`.
- El formateo del `Caddyfile` no es obligatorio, pero evita errores sutiles.
- El binario se reemplaza sin confirmación interactiva, asumiendo que se desea actualizar.

Este comportamiento es intencionado para favorecer automatización y consistencia.

Resumen breve

- Instalar Go y `xcaddy`.
- Ejecutar el script de construcción.
- Verificar que el servicio arranca correctamente.

Notas personales

El script puede adaptarse fácilmente para incluir otros módulos o ajustarse a distintos escenarios. La selección actual cubre las necesidades habituales sin añadir complejidad innecesaria.

Referencias

- Caddy – Build documentation
- Repositorio oficial de xcaddy

Monitorización

Monitorización

Actualizar agente Beszel (No necesario actualmente)

Script de actualización para Beszel

Este script te ahorra dolores de cabeza: automatiza la actualización del agente **Beszel** en tu servidor. Olvídate de actualizar a mano: con esto, siempre estarás usando la última versión disponible.

El script completo

> Repositorio en Gitea

¿Qué hace exactamente este script?

- Se mueve al directorio donde tienes Beszel.
- Para el servicio para no liarla mientras actualiza.
- Descarga el binario nuevo adaptado a tu sistema.
- Sustituye el antiguo y le da permisos para que pueda ejecutarse.
- Verifica que todo haya salido bien.
- Arranca de nuevo el servicio.
- Te va avisando de cada paso, para que sepas que no ha explotado nada.

Paso a paso, con lupa

1. Configuración de variables

Define dónde está Beszel y cómo se llama su servicio:

```
DIR="/ruta/a/tu/directorio/Beszel"
SERVICE="beszel-agent.service"
```

2. Cambio de directorio

Se mete en el directorio. Si falla, te avisa y corta la ejecución:

```
cd "$DIR" || { echo "No se pudo cambiar al directorio $DIR"; exit 1; }
```

3. Parar el servicio

Detiene el agente para actualizar sin problemas:

```
sudo systemctl stop "$SERVICE"
```

4. Descargar y reemplazar el binario

Usa `curl` para traer la última versión, extrae el ejecutable y lo deja listo:

```
curl -sL "https://github.com/henrygd/beszel/releases/latest/download/beszel-agent_$(uname -s)_$(uname -m | sed 's/x86_64/amd64/' | sed 's/armv7l/arm/' | sed 's/aarch64/arm64/').tar.gz" | tar -xz -O beszel-agent | tee ./beszel-agent >/dev/null && chmod +x beszel-agent
```

5. Comprobación de la descarga

Se asegura de que el archivo esté donde debe:

```
if [ -f "$DIR/beszel-agent" ]; then
  echo "El binario se actualizó correctamente."
else
  echo "Error: El binario no se descargó correctamente."
  exit 1
fi
```

6. Reiniciar el servicio

Levanta de nuevo el servicio para que trabaje con la versión nueva:

```
sudo systemctl start "$SERVICE"
```

7. Mensaje final

Te confirma que todo ha ido bien:

```
echo "Actualización completada."
```

Cómo usarlo

1. Guarda el script como `Update_Beszel.sh`.
2. Dale permisos de ejecución:

   ```
   chmod +x Update_Beszel.sh
   ```

3. Ejecútalo cuando quieras actualizar:

   ```
   ./Update_Beszel.sh
   ```

Ventajas de este script

- **Automatización total:** No tienes que ir a GitHub a mano.
- **Velocidad:** Lo actualiza todo en segundos.
- **Compatibilidad inteligente:** Detecta tu sistema y arquitectura solo.
- **Seguridad:** Para el servicio antes de tocar nada, como debe ser.

Nota importante

Desde las últimas versiones, **Beszel** ya incluye una función de autoactualización integrada.

¿Quiere decir que este script ya no sirve? No exactamente: **puedes seguir usándolo si prefieres forzar la actualización manualmente o asegurarte de tener el último binario en situaciones especiales**.

Monitorización

Monitorización simple de SAI (UPS) con Python

Este script permite consultar el estado de tu SAI (*Uninterruptible Power Supply*) usando `upsc`, mostrando la información en consola con colores, alertas y detalles básicos del sistema. Es una forma ligera y práctica de saber si todo va bien con tu SAI, sin necesidad de dashboards pesados.

El código está disponible aquí: CheckSAI.sh en Gitea

Características

- Consulta parámetros clave del SAI: carga, voltaje, frecuencia, carga de salida...
- Muestra alertas si algún valor está fuera del rango normal.
- Incluye información del sistema (hostname, procesador, plataforma).
- Salida en colores para facilitar la lectura.
- Ayuda integrada con `-h` o `--help`.

Requisitos

Antes de usar el script, necesitas:

- Python 3.x instalado.
- El paquete `colorama`:

```
pip install colorama
```

- Tener el comando `upsc`, que forma parte de **NUT (Network UPS Tools)**.

Adaptaciones necesarias

Antes de ejecutar el script, asegúrate de editar:

- **Nombre del SAI (`ups_name`)**: por defecto está como `nutdev1`. Sustitúyelo por el nombre real de tu SAI.
- **Dirección y puerto del servicio**: si no usas `localhost:4500`, edita el endpoint en la función que lanza `upsc` para reflejar la IP/puerto correctos.

¿Qué hace exactamente?

1. Ejecuta `upsc` para obtener los datos en bruto del SAI.
2. Extrae y organiza esa información en un formato legible.
3. Comprueba si los valores críticos están dentro de los rangos normales.
4. Muestra alertas si algo está fuera de lo previsto.
5. Añade información sobre el sistema donde se ejecuta.

Alternativa visual

Si prefieres una interfaz gráfica sencilla en el navegador, puedes probar con PeaNUT, también basado en NUT.

Sistema

Sistema

Actualizar wireguard-tools desde el repositorio oficial

Las versiones de `wireguard-tools` incluidas en Ubuntu permanecen sin actualizar desde hace años (`v1.0.20210914` a fecha de esta nota). Este script permite **actualizar directamente las utilidades de espacio de usuario desde el repositorio oficial de Jason Donenfeld (`git.zx2c4.com`)**, compilando los binarios desde la fuente y reemplazando los del sistema.

Por qué usar una versión actualizada

Las versiones distribuidas por Ubuntu se detuvieron en `v1.0.20210914`, mientras que la versión oficial más reciente (`v1.0.20250521`) incluye numerosas mejoras:

- Correcciones en la gestión de `resolvconf` y DNS.
- Ajustes de MTU automático más precisos.
- Compatibilidad ampliada con FreeBSD, Android y Windows.
- Eliminación de fugas momentáneas de tráfico al levantar el túnel.
- Limpieza y refactorización del código (`setconf` → `addconf` , mejor manejo de memoria, etc.).

En resumen: mantener `wireguard-tools` actualizado mejora la compatibilidad, estabilidad y seguridad del sistema. Los cambios completos pueden consultarse aquí: Registro oficial de cambios

---

Script utilizado

Repositorio en Gitea: `UpdateWireguard.sh`

El script:

- Comprueba si existen las herramientas y dependencias necesarias (`git` , `make` , `gcc` , etc.).
- Instala los paquetes faltantes mediante `apt` .
- Clona el repositorio oficial de `wireguard-tools` .
- Compila e instala los binarios (`wg` , `wg-quick` , etc.) sobre el sistema.

- Muestra la versión instalada al finalizar.

Ejemplo de uso

Instalar primero `wireguard` desde los repositorios del sistema para disponer de la base necesaria:

```
sudo apt install wireguard -y
```

Luego ejecutar el script para actualizar las herramientas:

```
chmod +x UpdateWireguard.sh
sudo ./UpdateWireguard.sh
```

Salida esperada:

```
Installing missing build dependencies: ...
Cloning into 'wireguard-tools'...
Installed version:
wireguard-tools v1.0.2025xxxx - https://git.zx2c4.com/wireguard-tools
```

Detalles técnicos

Este script solo afecta a las **utilidades de espacio de usuario** (`wg`, `wg-quick`, etc.), **no al módulo del kernel**. El kernel mantiene su versión estable; los binarios pueden actualizarse sin riesgo.

Referencias

- Repositorio oficial de wireguard-tools
- Instalación de WireGuard
- Script en Gitea

Sistema

Cambiar el plan de CPU según enchufe o batería

Introducción

Apunte para automatizar el cambio de plan de energía (governor) en un portátil según si está enchufado o funcionando con batería. La idea es olvidarse de cambiarlo manualmente y que se ajuste solo.

Características

- Detecta si el portátil está en corriente o en batería.
- Cambia el governor de CPU automáticamente.
- Funciona como servicio systemd para que siempre esté activo.

Requisitos previos

- Linux (probado en distros con `/sys/class/power_supply`, funciona bien en Fedora 42).
- Tener sudo configurado para poder escribir en `scaling_governor` sin pedir contraseña (o configurar `sudoers`).
- Acceso root para instalar y habilitar servicios.

Script principal

> “ □□ Puedes consultarlo en Gitea

- Se revisa cada 3 minutos (`sleep 180`).

- Modificar los governors según preferencias.

Servicio systemd

Archivo en `/etc/systemd/system/power_manager.service`:

```
[Unit]
Description=Power Manager Script
After=network.target

[Service]
Type=simple
ExecStart=/bin/bash -c "sudo /home/usuario/Documentos/Scripts/power_manager.sh"
Restart=always
User=usuario
Environment=DISPLAY=:0
Environment=HOME=/home/usuario

[Install]
WantedBy=multi-user.target
```

- Cambiar `usuario` por el nombre real si se copia literal.
- Importante revisar permisos y rutas.

Configuración en sudoers

Añadir al archivo sudoers (con `visudo`) para permitir ejecutar el script sin contraseña:

```
# Custom
usuario ALL=(ALL) NOPASSWD: /home/usuario/Documentos/Scripts/power_manager.sh
```

Errores comunes o decisiones importantes

- **Permisos sudo**: hay que permitir `sudo tee` sin contraseña para que funcione sin bloqueo.
- **Ruta AC0**: depende del hardware, a veces es `AC`, `ACAD`, etc. Verificar en `/sys/class/power_supply/`.

- **Intervalo**: se podría reducir el tiempo de espera (sleep), pero puede consumir más recursos.

Resumen breve

- Script en bash que revisa si está enchufado.
- Cambia automáticamente el governor de CPU.
- Servicio systemd para que se inicie solo.
- Configurar sudoers para no pedir contraseña.
- Revisar nombre del adaptador AC.

Notas personales

Quizá se podría mejorar para no usar un bucle infinito y en su lugar usar udev o eventos ACPI, pero el loop es más sencillo y robusto para no complicarse.

Sistema

Cambiar el plan de CPU según enchufe o batería (v2)

Introducción

Sistema para aplicar perfiles de energía dinámicos en Linux según el estado de alimentación (AC o batería). La idea es automatizar el cambio de governor, EPP y boost de CPU sin depender de herramientas externas.

Requisitos previos

- Kernel con soporte para `cpufreq` y governors (`performance`, `schedutil`, `powersave`, etc.).
- Opcional: soporte `amd-pstate-epp` para Energy Performance Preference.
- `systemd` y `udev` disponibles (entornos modernos ya lo traen).
- Script personalizado Change_Power_v2.sh.

Flujo general

1. **Regla udev** detecta conexión/desconexión de corriente.
2. **udev dispara units systemd** específicas para cada estado.
3. **systemd ejecuta script Bash**, que aplica el perfil de energía correspondiente.

Configuración

1. Regla udev

Archivo `/etc/udev/rules.d/99-power-profile.rules`:

```
# Detectar cambio en el estado de AC
SUBSYSTEM=="power_supply", KERNEL=="AC0", ATTR{type}=="Mains", ACTION=="change",
ATTR{online}=="1", TAG+="systemd", ENV{SYSTEMD_WANTS}+="power-profile-ac.service"
SUBSYSTEM=="power_supply", KERNEL=="AC0", ATTR{type}=="Mains", ACTION=="change",
ATTR{online}=="0", TAG+="systemd", ENV{SYSTEMD_WANTS}+="power-profile-bat.service"
```

Con esto, cualquier cambio de estado en el adaptador de corriente dispara la unit correspondiente.

---

2. Units systemd

AC → `/etc/systemd/system/power-profile-ac.service`

```
[Unit]
Description=Perfil de energía: AC (enchufado)

[Service]
Type=oneshot
ExecStart=/usr/local/bin/Change_Power_v2.sh ac
```

Batería → `/etc/systemd/system/power-profile-bat.service`

```
[Unit]
Description=Perfil de energía: Batería

[Service]
Type=oneshot
ExecStart=/usr/local/bin/Change_Power_v2.sh bat
```

---

3. Script Bash

El script principal se mantiene versionado en Gitea:

Change_Power_v2.sh en Gitea

Testeado en un portátil con **Fedora 41**, donde funciona correctamente el cambio de perfiles al enchufar o desenchufar el cargador.

Errores comunes o decisiones importantes

- **Script genérico**: no contiene información personal ni rutas específicas. Solo depende del nombre del dispositivo de alimentación (`AC0` por defecto), que puede variar según el hardware.
- **Detección de AC**: en algunos equipos el dispositivo no se llama `AC0` sino `AC`, `ACAD` o similar. Conviene verificar en `/sys/class/power_supply/`.
- **Permisos**: escribir en `/sys/devices/system/cpu/*/cpufreq/*` puede requerir root. Se ejecuta vía systemd, así que no suele ser problema.
- **Drivers**: si `amd-pstate-epp` no está disponible, las opciones EPP se ignoran de forma silenciosa.

Resumen breve

- udev escucha cambios en el estado de corriente.
- systemd ejecuta units `power-profile-ac` o `power-profile-bat`.
- El script aplica governors, EPP y boost con fallbacks robustos.