Jamf Pro_

Guía de atajos · administración de Macs

```
admin@dajomar:~$ sudo jamf policy
```

Cómo funciona Jamf

Jamf gestiona los Mac por **dos canales** que trabajan juntos:

Canal	Qué hace
MDM (Apple)	Perfiles de configuración, comandos remotos (bloquear, borrar), apps del App Store. Usa APNs de Apple.
Binario jamf	Políticas, scripts, paquetes, inventario. Corre en el Mac y hace check-in con el servidor.

Si fallan los perfiles, el problema suele estar en el MDM. Si fallan las políticas, revisa el binario `jamf`.

Conceptos clave

Policy	Acción a ejecutar: instalar, correr script, configurar.
Config Profile	Ajustes de macOS vía MDM (Wi-Fi, FileVault, restricciones).
Smart Group	Grupo dinámico por criterios (versión de macOS, app instalada).
Static Group	Grupo manual: tú eliges los equipos.
Scope	A quién aplica: **Targets**, **Limitations** y **Exclusions**.
PreStage	Configuración de enrolamiento automático (ADE / Apple Business Manager).
Ext. Attribute	Dato personalizado del inventario, calculado con un script.
Self Service	Tienda interna donde el usuario instala apps y corre tareas.
Check-in	El Mac pregunta al servidor si hay políticas pendientes.
Recon	Envío de inventario del Mac a Jamf Pro.

Binario jamf (en el Mac)

Conexión y estado

```
# ¿llega al servidor?
sudo jamf checkJSSConnection
jamf version
sudo jamf help
sudo jamf help policy  # ayuda de un verbo
```

Políticas

```
sudo jamf policy           # check-in manual
# trigger custom
sudo jamf policy -event instalar_zoom
sudo jamf policy -id 123   # una política
sudo jamf policy -verbose  # con detalle
# reinicia "once per"
sudo jamf flushPolicyHistory
```

Inventario y gestión

```
sudo jamf recon            # envía inventario
sudo jamf recon -endUsername dmaza
sudo jamf manage           # reaplica el framework
sudo jamf removeFramework  # ⚠ quita Jamf del Mac
```

Pro Tip:
Después de cambiar algo en Jamf Pro, no esperes al check-in: corre `sudo jamf policy` para las políticas y `sudo jamf recon` para que los Smart Groups se actualicen al momento.

Perfiles y MDM

```
profiles status -type enrollment     # ¿enrolado?
sudo profiles show -type enrollment  # info ADE
# re-enrolar ADE
sudo profiles renew -type enrollment
sudo profiles list  # perfiles instalados
```

Enrolled via DEP: Yes y **MDM enrollment: Yes (User Approved)** es el estado sano. Si el MDM no está aprobado por el usuario, muchos perfiles no se aplican.

Logs

```
tail -f /var/log/jamf.log  # políticas en vivo
grep -i error /var/log/jamf.log
log stream --info --predicate \
  'subsystem == "com.apple.ManagedClient"'  # MDM
log show --last 1h --predicate \
  'process == "mdmclient"'
```

En Jamf Pro: **Computers → equipo → History** muestra los logs de políticas y comandos MDM pendientes o fallidos.

Políticas: triggers y frecuencia

Trigger	Cuándo corre
Recurring Check-in	En cada check-in periódico
Enrollment Complete	Al terminar el enrolamiento
Login / Startup	Al iniciar sesión / al encender
Network State Change	Cambia la red o la IP
Custom	Con `jamf policy -event nombre`
Self Service	Cuando el usuario la ejecuta

Frecuencia	Uso típico
Once per computer	Instalaciones iniciales
Once per user per comp.	Ajustes por usuario
Once every day/week	Inventario, mantenimiento
Ongoing	Self Service o triggers custom

⚠ **Ongoing** + **Recurring Check-in** = la política corre cada 15 minutos (o lo que dure tu check-in) para siempre.

Scripts en Jamf

$1	Punto de montaje del disco de destino
$2	Nombre del equipo
$3	Usuario con sesión (si aplica)
$4 a $11	Parámetros personalizados de la política

```
#!/bin/bash
app_name="$4"                       # parámetro 4
usuario=$(stat -f%Su /dev/console)  # usuario real
if [[ -z "$app_name" ]]; then
  echo "Falta el parámetro 4"; exit 1
fi
echo "Instalando $app_name para $usuario"
exit 0
```

El código de salida importa: `exit 0` = Completed · otro valor = Failed en los logs.

Extension Attribute (dato de inventario)

```
#!/bin/bash
estado=$(fdesetup status | head -1)
echo "<result>$estado</result>"
```

Jamf guarda lo que esté entre `<result>`. Se actualiza con cada `recon`.

Comandos macOS para soporte

Equipo y usuarios

```
# modelo, serial
system_profiler SPHardwareDataType
sw_vers                  # versión de macOS
scutil --get ComputerName
sudo scutil --set ComputerName "MAC-DMAZA"
stat -f%Su /dev/console  # quién está logueado
dscl . -list /Users | grep -v '^_'
id dmaza                 # grupos (admin = 80)
```

Seguridad

```
fdesetup status     # FileVault
sudo fdesetup list  # usuarios habilitados
sysadminctl -secureTokenStatus dmaza
spctl --status      # Gatekeeper
csrutil status      # SIP
# firewall
sudo /usr/libexec/ApplicationFirewall/\
socketfilterfw --getglobalstate
```

Software y paquetes

```
softwareupdate -l        # actualizaciones
sudo softwareupdate -ia  # instalar todas
sudo installer -pkg app.pkg -target /
pkgutil --pkgs | grep -i zoom
pkgutil --pkg-info com.ejemplo.app
mdls -name kMDItemVersion /Applications/Zoom.us.app
```

Servicios

```
sudo launchctl list | grep -i jamf
ls /Library/LaunchDaemons/ | grep jamf
```

API de Jamf Pro

Con un **API Client** (Settings → API Roles and Clients):

```
url="https://tuempresa.jamfcloud.com"
token=$(curl -s -X POST "$url/api/oauth/token" \
  -d "grant_type=client_credentials" \
  -d "client_id=$CLIENT_ID" \
  -d "client_secret=$CLIENT_SECRET" \
  | jq -r .access_token)

curl -s -H "Authorization: Bearer $token" \
  "$url/api/v1/computers-inventory?page-size=10"
```

Da al API Client solo los permisos que necesita y nunca guardes el secreto dentro de un script en Jamf. Documentación en `/api/doc` de tu instancia.

Pro Tip: un Mac no recibe políticas

`sudo jamf checkJSSConnection` → revisa `/var/log/jamf.log` → `profiles status -type enrollment` → `sudo jamf manage` → `sudo jamf recon` → confirma en Jamf Pro que el equipo está en el **scope** y no en una exclusión.