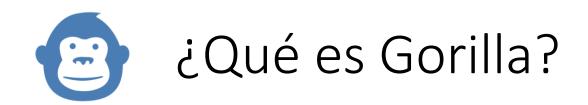
# Automatización de tareas en Windows con Gorilla



Un modelo cliente-servidor sobre HTTP/HTTPS para desplegar Scripts y Software de forma desatendida y escalable



Gorilla se define en su proyecto GitHub como:

Munki-like Application Management for Windows

Munki es una herramienta software diseñada por Walt Disney para el entorno macOS que se utiliza para desplegar software y/o scripts en las máquinas clientes de forma desatendida.

Y esto mismo es lo que hace Gorilla, pero en Windows. Es una herramienta Open-Source con la misma idea pero en otro entorno. Tiene licencia **Apache License 2.0** 



## ¿Qué podemos hacer con Gorilla?

- Realizar instalaciones de software en caliente y de forma desatendida con pocas líneas de código y sólo en los equipos deseados (escalabilidad).
- Solucionar errores que hayamos descubierto en nuestra imagen mediante scripts en Powershell sin tener que volver a clonar o ir uno a uno.



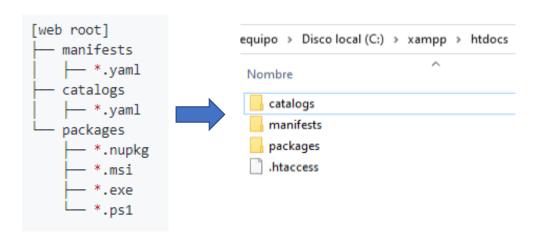
### Ejemplos reales:

- Ejecutar un script con alguna configuración específica sólo en ciertos equipos (los de los profesores).
- Nos traen una impresora nueva, y reconfiguramos el sistema para que borre la antigua y añada la actual (instalando los nuevos drivers, predeterminando, cambiando IP si es necesario, etc.). Y sólo en cierta aula.
- Cambiar el fondo de pantalla de todos los equipos por uno nuevo en cualquier momento.
- Crear/habilitar una cuenta de usuario para un examen en una fecha determinada.
- Crear cierta tarea programada en el equipo para que realice trabajos con una periodicidad específica.



## ¿Qué estructura tiene Gorilla?

 SERVIDOR: servidor web con la siguiente estructura (y nada más)



 CLIENTE: un archivo de configuración que lo vincula al servidor (y nada más)

```
C:\ProgramData\gorilla\config.vaml • - Sublime Text (UNI
url: https://YourWebServer/gorilla/
                                                       File Edit Selection Find View Goto Tools Project
manifest: example
catalogs:
                                                               config.yaml
  - alpha
  - beta
                                                              url: http://10.1.XX.XX/
app data path: C:/gorilla/cache
                                                              manifest: nombre equipo manifest
auth user: GorillaRepoUser
auth_pass: pizzaisyummy
                                                                - mi catalog
                                                              app_data_path: C:/gorilla/cache
tls auth: true
                                                              auth user: tecnico
tls client cert: c:/certs/client.pem
tls client key: c:/certs/client.key
tls server cert: c:/certs/server.pem
```



## ¿Qué necesito saber para empezar con Gorilla?

No se requieren grandes conocimientos específicos. Puede hacerlo cualquier persona que sepa crear/interpretar un script básico.

- Sistemas: Montar un servidor Web

- Scripting: Manejarse en línea de comandos (Powershell y CMD).

### ¿Qué seré capaz de hacer con Gorilla?

A partir de un Windows 10 base (con unas configuraciones mínimas) y un catálogo de aplicaciones y scripts en mi servidor de gorilla seré capaz de crear una imagen casi completa de aulas (con aplicaciones en su última versión, actualizaciones de sistema, cuentas de usuario, impresoras, y mucho más).

Y si se te olvida alguna configuración o encuentras cualquier error tras clonar el aula, ¡no te preocupes! en caliente puedes solucionarlo.

## ¿Qué haríamos en el curso?

#### **Máquina Servidor**:

- Montar un servidor web con una instancia de Multipass que contenga la estructura de carpetas de gorilla (catalogs, manifests y packages).
- Crear varios manifests
- Definir un catalog y crear un script básico. Por ejemplo, que cree una carpeta en el escritorio y deje una marca para que podamos ver que ya se ha ejecutado (aunque borrásemos la carpeta)

#### Máquina/s Cliente/s:

- Definir en cada cliente su archivo de configuración (a qué manifest del servidor lo vamos a asociar)
- Desplegar tareas en estos
- Observar logs en tiempo real o tras la ejecución
- Crear nuestros propios logs (muy útil)
- Controlar tareas que sólo queremos que se ejecuten una vez
- Procedimientos para depurar errores
- Herencia de manifests



## Desplegar manifests en el cliente con Gorilla

#### ¿Cómo conectan cliente-servidor?

Con el binario de Gorilla. Siempre desde el cliente hacia el servidor.

#### ¿Cada cuanto tiempo conectan?

Con la periodicidad que yo quiera. Debemos crear un script que indique al cliente cuándo debe conectarse.

¿Y si quiero ejecutar cierta tarea de un manifest sólo una vez? Si ejecutamos gorilla de forma periódica, necesitaremos que ciertas tareas sean Idempotentes. Un procedimiento de trabajo con scripts soluciona esto.

¿Cómo puedo depurar errores si no se ejecuta como espero? Conociendo la forma de trabajo de Gorilla. También los archivos de log y su significado. Además, el editor de Powershell también nos puede ayudar bastante.

Administrador: c:\windows\system32\cmd.exe - powershell PS C:\Users\tecnico\scripts> gorilla -help gorilla v1.0.0.5 Gorilla - Munki-like Application Management for Windows https://github.com/1dustindavis/gorilla Usage: gorilla.exe [options] Options: c, -config path to configuration file in yaml format C, -checkonly enable check only mode -verbose enable verbose output -debug enable debug output -about displays the version number and other build info -version display the version number h, -help display this help message PS C:\Users\tecnico\scripts>



#### Ejemplos:

- Paso 1/3 - Implementando el catálogo de aplicaciones del servidor

Aquí defino todo lo PUEDO hacer en mis clientes, lo haga o no



```
! todos los clientes manifest.yaml • ! mi_catalogo_de_aplicaciones.yaml ×
 EXPLORADOR

✓ EDITORES ABIERTOS 1 SIN GUARDAR

                                                               catalogs > ! mi_catalogo_de_aplicaciones.yaml

    ! todos_los_clientes_manifest.yaml manifests

  × ! mi_catalogo_de_aplicaciones.yaml catalogs
                                                                        nstalar OGClient:
                                                                        display_name: instalar_OGClient

✓ catalogs

 ! mi_catalogo_de_aplicaciones.yaml
                                                                          - path: C:\Program Files (x86)\OGAgent\OGAgentService.exe
 ! cliente1 manifest.yaml
                                                                                                                                                                         Instalación software (.exe)
 ! cliente2_manifest.yaml
                                                                          hash: EE950047C7FB18BDECCA183BE03B049F077B39C57F35AECC29E4FCD50DA5A63B
 ! todos_los_clientes_manifest.yaml
                                                                          location: packages/OGAgent/OGAgentSetup-1.1.1b.exe

→ packages

  > crear_carpeta_escritorio
                                                                          - /server 10.1.XX.X
.htaccess
                                                                          type: exe
                                                                          location: packages/OGAgent/uninstall_ogagent.ps1
                                                                          hash: 9361AEB7EB57635720AD0D345CE79C6AF85D212968D132CFE49E1735C6760B7C
                                                                        version: 1.1.1b
                                                                            exit 0
                                                                          location: packages/scripts/crear carpeta escritorio/crear carpeta escritorio.ps1
                                                                          hash: 1DECF48D02AE19969FE7694AE2F5D295405AA1CB7EAD51FA43D5033B9C388C7E
                                                                          type: ps1
                                                                        version: 1.0
                                                                                                                                                                            Powershell scripts (.ps1)
                                                                      cambiar fondo escritorio:
                                                                        display name: cambiar fondo escritorio
                                                                            exit 0
                                                                          location: packages/scripts/cambiar fondo escritorio/cambiar fondo escritorio.ps1
                                                                          hash: 8A6FDBF004E6BFB8C98D8D3BEBA078E2EF014EC5580662D337497E52066796B7
                                                                          type: ps1
                                                                        version: 1.0
```



Ejemplos:
- Paso 2/3 - vinculando un cliente con su manifest

A un cliente concreto, le digo: TÚ vas a hacer lo que diga "cliente1 manifest"



¿Tengo que ir equipo a equipo configurando la línea "manifest" de su archivo de configuración "config.yaml"?

La respuesta es NO.

Con un juego de scripts y manifest auxiliares podemos automatizar esta configuración, para que cada equipo obtenga su manifest correspondiente de forma automatizada en función de un dato concreto, como por ejemplo su IP.

```
config.yaml X
config.yaml
     url: https://10.1.XX.XX
    manifest: cliente1 manifest
    catalogs:
      - mi catalogo de aplicaciones
    app_data_path: C:/gorilla/cache
    auth user: XXXXXXX
    auth pass: XXXXXXX
```

Configuración de un pc cualquiera del aula. Lo estoy asociando a "cliente1 manifest"



## Ejemplos: - Paso 3/3 - implementando **manifests** en clientes

Indico qué tareas de mi catálogo quiero implementar en cada cliente, o en ambos (included manifests)







```
cliente1_manifest.yaml X
manifests > ! cliente1_manifest.yaml
       name: cliente1_manifest
       managed installs:
         - instalar OGClient
       managed uninstalls:
       managed updates:
       included manifests:

    todos los clientes manifest

       catalogs:
         - mi catalogo de aplicaciones
```

```
cliente2_manifest.yaml ×
manifests > ! cliente2_manifest.yaml
      name: cliente2 manifest
      managed installs:
       - crear carpeta escritorio
      managed uninstalls:
      managed updates:
      included manifests:

    todos los clientes manifest

      catalogs:
         - mi_catalogo_de_aplicaciones
 10
```

```
todos_los_clientes_manifest.yaml •
manifests > ! todos_los_clientes_manifest.yaml
       name: todos los clientes manifest
       managed installs:
         - cambiar fondo escritorio
       managed uninstalls:
       managed updates:
       included manifests:
       catalogs:
         - mi catalogo de aplicaciones
```

Asociado con cliente1

Asociado con cliente2

Sin asociación