Implementación de una Verificación de Proxy en Windows

Para implementar una verificación de proxy en Windows utilizando Git-Bash y PowerShell, es necesario configurar mecanismos en cada shell para detectar cuándo se ejecutan comandos relacionados con la red y mostrar la configuración del proxy si están configurados. A continuación se presentan los pasos y el código para ambos Git-Bash y PowerShell. El "terminal"mencionado probablemente se refiere a Windows Terminal que hospeda estos shells, por lo que nos centraremos en las implementaciones de Git-Bash y PowerShell.

Para Git-Bash

Git-Bash es una emulación de Bash en Windows, y podemos usar una trampa `DEBUG` para ejecutar una función antes de la ejecución de cada comando. El objetivo es verificar si el comando está relacionado con la red y si se han configurado los ajustes del proxy, luego mostrar estos ajustes.

Pasos:

1. **Definir la lista de comandos relacionados con la red.**
2. **Crear una función para mostrar los ajustes del proxy.**
3. **Crear una función para verificar el comando y los ajustes del proxy.**
4. **Configurar la trampa `DEBUG` para ejecutar la verificación antes de cada comando.**
5. **Definir una función manual `checkproxy` para mostrar los ajustes del proxy bajo demanda.**
6. **Agregar todas las configuraciones al archivo `.bashrc`.**

Implementación: Agregue el siguiente código a su archivo `~/.bashrc` (créelo si no existe):

```
# Lista de comandos relacionados con la red
network_commands=(
    "gpa"
    "git"
    "ssh"
    "scp"
    "sftp"
    "rsync"
    "curl"
    "wget"
    "apt"
    "yum"
    "dnf"
```

```
    "npm"
    "yarn"
    "pip"
    "pip3"
    "gem"
    "cargo"
    "docker"
    "kubectl"
    "ping"
    "traceroute"
    "netstat"
    "ss"
    "ip"
    "ifconfig"
    "dig"
    "nslookup"
    "nmap"
    "telnet"
    "ftp"
    "nc"
    "tcpdump"
    "adb"
    "bundle"
    "brew"
    "cpanm"
    "bundle exec jekyll"
    "make"
    "python"
    "glcoud"
)

# Función para mostrar los ajustes del proxy
display_proxy() {
    echo -e "  **Ajustes de Proxy Detectados:**"
    [ -n "$HTTP_PROXY" ] && echo "   - HTTP_PROXY: $HTTP_PROXY"
    [ -n "$http_proxy" ] && echo "   - http_proxy: $http_proxy"
    [ -n "$HTTPS_PROXY" ] && echo "   - HTTPS_PROXY: $HTTPS_PROXY"
    [ -n "$https_proxy" ] && echo "   - https_proxy: $https_proxy"
    [ -n "$ALL_PROXY" ] && echo "   - ALL_PROXY: $ALL_PROXY"
    [ -n "$all_proxy" ] && echo "   - all_proxy: $all_proxy"
```

```
    echo ""
}

# Función para verificar si el comando está relacionado con la red y si los proxies están configurados
proxy_check() {
    local cmd
    # Extraer la primera palabra del comando
    cmd=$(echo "$BASH_COMMAND" | awk '{print $1}')

    for network_cmd in "${network_commands[@]}"; do
        if [[ "$cmd" == "$network_cmd" ]]; then
            # Verificar si alguna variable de entorno de proxy está configurada
            if [ -n "$HTTP_PROXY" ] || [ -n "$http_proxy" ] || \
               [ -n "$HTTPS_PROXY" ] || [ -n "$https_proxy" ] || \
               [ -n "$ALL_PROXY" ] || [ -n "$all_proxy" ]; then
                display_proxy
            fi
            break
        fi
    done
}

# Configurar la trampa DEBUG para ejecutar proxy_check antes de cada comando
trap 'proxy_check' DEBUG

# Función para verificar manualmente los ajustes del proxy
checkproxy() {
    echo "HTTP_PROXY: $HTTP_PROXY"
    echo "HTTPS_PROXY: $HTTPS_PROXY"
    echo "Proxy HTTP de Git:"
    git config --get http.proxy
    echo "Proxy HTTPS de Git:"
    git config --get https.proxy
}
```

Cómo Funciona:

- El array `network_commands` enumera los comandos relacionados con la red.
- `display_proxy` muestra todas las variables de entorno de proxy relevantes si están configuradas.
- `proxy_check` usa `BASH_COMMAND` (disponible en la trampa `DEBUG`) para obtener el comando que se está

ejecutando, extrae la primera palabra y verifica si coincide con algún comando de red. Si las variables de proxy están configuradas, las muestra.

- La línea `trap 'proxy_check' DEBUG` asegura que `proxy_check` se ejecute antes de cada comando.
- `checkproxy` permite ver manualmente los ajustes del proxy, incluyendo las configuraciones específicas de Git.
- Después de agregar esto a `.bashrc`, reinicie Git-Bash o ejecute `source ~/.bashrc` para aplicar los cambios.

Uso:

- Cuando ejecute un comando de red (por ejemplo, `git clone`, `curl`), si los ajustes del proxy están configurados, se mostrarán antes de que se ejecute el comando.
- Ejecute `checkproxy` para ver manualmente los ajustes del proxy.

Para PowerShell

PowerShell no tiene un equivalente directo a la trampa `DEBUG` de Bash, pero podemos usar el manejador `CommandValidationHandler` del módulo `PSReadLine` para lograr una funcionalidad similar. Este manejador se ejecuta antes de cada comando, permitiéndonos verificar los comandos y los ajustes del proxy.

Pasos:

1. **Definir la lista de comandos relacionados con la red.**
2. **Crear una función para mostrar los ajustes del proxy.**
3. **Configurar el `CommandValidationHandler` para verificar comandos y ajustes del proxy.**
4. **Definir una función manual `checkproxy` para mostrar los ajustes del proxy bajo demanda.**
5. **Agregar todas las configuraciones a su perfil de PowerShell.**

Implementación: Primero, localice su archivo de perfil de PowerShell ejecutando `$PROFILE` en PowerShell. Si no existe, créelo:

```
New-Item -Type File -Force $PROFILE
```

Agregue el siguiente código a su perfil de PowerShell (por ejemplo, `Microsoft.PowerShell_profile.ps1`):

```
# Lista de comandos relacionados con la red
$networkCommands = @(
    "gpa",
```

```
    "git",
    "ssh",
    "scp",
    "sftp",
    "rsync",
    "curl",
    "wget",
    "apt",
    "yum",
    "dnf",
    "npm",
    "yarn",
    "pip",
    "pip3",
    "gem",
    "cargo",
    "docker",
    "kubectl",
    "ping",
    "traceroute",
    "netstat",
    "ss",
    "ip",
    "ifconfig",
    "dig",
    "nslookup",
    "nmap",
    "telnet",
    "ftp",
    "nc",
    "tcpdump",
    "adb",
    "bundle",
    "brew",
    "cpanm",
    "bundle exec jekyll",
    "make",
    "python",
    "glcoud"
)
```

```powershell
# Función para mostrar los ajustes del proxy
function Display-Proxy {
    Write-Host "  **Ajustes de Proxy Detectados:**"
    if ($env:HTTP_PROXY) { Write-Host "    - HTTP_PROXY: $env:HTTP_PROXY" }
    if ($env:http_proxy) { Write-Host "    - http_proxy: $env:http_proxy" }
    if ($env:HTTPS_PROXY) { Write-Host "    - HTTPS_PROXY: $env:HTTPS_PROXY" }
    if ($env:https_proxy) { Write-Host "    - https_proxy: $env:https_proxy" }
    if ($env:ALL_PROXY) { Write-Host "    - ALL_PROXY: $env:ALL_PROXY" }
    if ($env:all_proxy) { Write-Host "    - all_proxy: $env:all_proxy" }
    Write-Host ""
}

# Configurar el CommandValidationHandler para verificar comandos antes de la ejecución
Set-PSReadLineOption -CommandValidationHandler {
    param($command)
    # Extraer la primera palabra del comando
    $cmd = ($command -split ' ')[0]

    if ($networkCommands -contains $cmd) {
        # Verificar si alguna variable de entorno de proxy está configurada
        if ($env:HTTP_PROXY -or $env:http_proxy -or $env:HTTPS_PROXY -or $env:https_proxy -or $env:ALL_PROXY -
            Display-Proxy
        }
    }
    # Siempre devolver verdadero para permitir la ejecución del comando
    return $true
}

# Función para verificar manualmente los ajustes del proxy
function checkproxy {
    Write-Host "HTTP_PROXY: $env:HTTP_PROXY"
    Write-Host "HTTPS_PROXY: $env:HTTPS_PROXY"
    Write-Host "Proxy HTTP de Git:"
    git config --get http.proxy
    Write-Host "Proxy HTTPS de Git:"
    git config --get https.proxy
}
```

Cómo Funciona:

- `$networkCommands` es un array de comandos relacionados con la red.
- `Display-Proxy` muestra todas las variables de entorno de proxy relevantes si están configuradas.
- `Set-PSReadLineOption -CommandValidationHandler` define un bloque de script que se ejecuta antes de cada comando:
  - Extrae la primera palabra del comando.
  - Verifica si está en `$networkCommands`.
  - Si las variables de proxy están configuradas, llama a `Display-Proxy`.
  - Devuelve `$true` para asegurar que se ejecute el comando.
- `checkproxy` permite ver manualmente los ajustes del proxy, incluyendo los proxies específicos de Git.
- Después de agregar a su perfil, reinicie PowerShell o ejecute `. $PROFILE` para aplicar los cambios.

Requisitos:

- Se requiere el módulo `PSReadLine`, que está incluido por defecto en PowerShell 5.1 y versiones posteriores.
- Si usa una versión anterior, es posible que necesite actualizar PowerShell o encontrar un método alternativo (no se cubre aquí, ya que la mayoría de los sistemas usan versiones más recientes).

Uso:

- Cuando ejecute un comando de red (por ejemplo, `git pull`, `curl`), si los ajustes del proxy están configurados, se mostrarán antes de que se ejecute el comando.
- Ejecute `checkproxy` para ver manualmente los ajustes del proxy.

Notas sobre el "Terminal"

- Si "terminal"se refiere a Windows Terminal, es simplemente un host para shells como Git-Bash, PowerShell o Command Prompt (cmd.exe).
- Las implementaciones anteriores funcionan dentro de sesiones de Git-Bash o PowerShell en Windows Terminal.
- Implementar una funcionalidad similar en Command Prompt (cmd.exe) no es práctico debido a sus limitadas capacidades de scripting. Se recomienda usar Git-Bash o PowerShell en su lugar.

Consideraciones Adicionales

- **Análisis de Comandos:**

  - Ambas implementaciones verifican solo la primera palabra del comando contra la lista de comandos de red. Por ejemplo, `git clone` dispara porque `git` está en la lista.
  - Comandos de varias palabras como `bundle exec jekyll` dispararán si `bundle` está en la lista, lo cual es suficiente para la mayoría de los casos.
  - Si es necesario, podría modificar el código para verificar todas las palabras en el comando, pero esto podría llevar a falsos positivos y generalmente es innecesario.

- **Variables de Proxy:**
  - Ambas implementaciones verifican `HTTP_PROXY`, `http_proxy`, `HTTPS_PROXY`, `https_proxy`, `ALL_PROXY` y `all_proxy` para cubrir las variaciones comunes.
  - En Windows, las variables de entorno no distinguen entre mayúsculas y minúsculas, pero se verifican ambas para mantener la consistencia con el comportamiento de Unix (especialmente en Git-Bash).

- **Configuraciones de Proxy de Git:**
  - La función `checkproxy` en ambos shells muestra las configuraciones específicas de proxy de Git usando `git config --get`.

Pruebas

- **Establecer Variables de Proxy:**
  - En Git-Bash: `export HTTP_PROXY=http://proxy.example.com:8080`
  - En PowerShell: `$env:HTTP_PROXY = "http://proxy.example.com:8080"`

- **Ejecutar Comandos de Red:**
  - Intente comandos como `git --version`, `curl -V`, etc.
  - Los ajustes del proxy deberían mostrarse antes de la salida del comando.

- **Usar `checkproxy`:**
  - Ejecute `checkproxy` en cualquiera de los shells para ver manualmente los ajustes del proxy.

Esta implementación proporciona una verificación robusta de proxy tanto en Git-Bash como en PowerShell, adecuada para su uso dentro de Windows Terminal o de forma independiente.