在運行命令前顯示代理設置

在中國生活或在需要使用 VPN 和代理的公司工作，可能會使軟件開發變得複雜。忘記配置這些設置通常會導致連接問題。為了簡化您的工作流程，我在 ChatGPT 的幫助下創建了一個簡單的 Zsh 腳本，當您運行特定的依賴網絡的命令時，該腳本會自動顯示您的代理設置。

為什麼要顯示代理設置？

代理和 VPN 對於安全訪問外部資源至關重要。在執行依賴網絡的命令之前顯示您的代理設置，可以幫助您快速識別和解決連接問題。

腳本

這個腳本利用 Zsh 的 `preexec` 函數來檢查即將執行的命令是否依賴網絡。如果是並且設置了代理環境變量，它將顯示當前的代理設置。

```
# 函數用於在執行某些命令之前檢查並顯示代理設置
preexec() {
    # 定義依賴網絡的命令
    local 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"
    # 根據需要添加更多命令
)

# 從命令行中提取第一個單詞（命令）
local cmd
cmd=$(echo "$1" | awk '{print $1}')

# 函數用於顯示代理變量
display_proxy() {
    echo -e "\n 檢測到代理設置:"

    [ -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 ""
    }

    # 檢查命令是否依賴網絡
    for network_cmd in "${network_commands[@]}"; do
        if [[ "$1" == "$network_cmd"* ]]; then
            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
}
```

在 Zsh 中設置腳本

1. 打開您的 `.zshrc` 文件

使用您喜歡的文本編輯器打開 `.zshrc` 配置文件。例如：

```
nano ~/.zshrc
```

2. 添加 `preexec` 函數

將上面的腳本粘貼到文件的末尾。

3. 保存並關閉

按 CTRL + O 保存，然後按 CTRL + X 退出。

4. 應用更改

重新加載您的.zshrc 以立即應用新配置：

```
source ~/.zshrc
```

測試設置

1. 啟用代理

臨時設置一個代理變量並使用 pip 運行一個依賴網絡的命令：

```
export HTTP_PROXY="http://127.0.0.1:7890"
pip install selenium beautifulsoup4 urllib3
```

預期輸出：

```
  檢測到代理設置:
    - HTTP_PROXY: http://127.0.0.1:7890
    - http_proxy: 127.0.0.1:7890
    - HTTPS_PROXY: 127.0.0.1:7890
    - https_proxy: 127.0.0.1:7890
    - ALL_PROXY: 127.0.0.1:7890
    - all_proxy: 127.0.0.1:7890

Collecting selenium
  Downloading selenium-4.x.x-py2.py3-none-any.whl (xxx kB)
Collecting beautifulsoup4
  Downloading beautifulsoup4-4.x.x-py3-none-any.whl (xxx kB)
Collecting urllib3
  Downloading urllib3-1.x.x-py2.py3-none-any.whl (xxx kB)
...
```

2. 未啟用代理

取消設置代理變量並運行相同的 `pip` 命令：

```
unset HTTP_PROXY
pip install selenium beautifulsoup4 urllib3
```

預期輸出：

```
Collecting selenium
  Downloading selenium-4.x.x-py2.py3-none-any.whl (xxx kB)
Collecting beautifulsoup4
  Downloading beautifulsoup4-4.x.x-py3-none-any.whl (xxx kB)
Collecting urllib3
  Downloading urllib3-1.x.x-py2.py3-none-any.whl (xxx kB)
...
```

(不應出現代理通知。)

3. 非網絡命令

運行一個本地命令，如 `ls`：

```
ls
```

預期輸出：

```
[文件和目錄列表]
```

(不應出現代理通知。)

自定義

- 擴展 `network_commands`：將任何其他依賴網絡的命令添加到 `network_commands` 數組中。
- 處理別名：確保任何依賴網絡的命令的別名都包含在 `network_commands` 列表中。

  ```
  alias gpa='git push all'
  ```

將"gpa" 添加到 network_commands 數組中，以便在使用此別名時觸發代理通知。

- 使用顏色增強可見性：

  為了在雜亂的終端中更好地可見性，您可以為代理通知添加顏色：

```
# 在.zshrc 的頂部添加顏色代碼
GREEN='\033[0;32m'
NC='\033[0m' # 無顏色

display_proxy() {
    echo -e "\n${GREEN} 檢測到代理設置:${NC}"

    [ -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 ""
}
```

結論

在受限的網絡環境中，管理代理設置對於順利的軟件開發至關重要。這個 Zsh 腳本確保您在運行需要網絡訪問的命令時始終了解您的代理配置，從而提高您的工作流程和故障排除效率。

祝您編程愉快！⌧