コマンド実行前にプロキシ設定を表示

中国に住んでいたり、VPN やプロキシを使用する企業で働いている場合、ソフトウェア開発が複雑になることがあります。これらの設定を忘れると、接続の問題が発生しがちです。ワークフローを効率化するために、ChatGPT の助けを借りて、特定のネットワーク依存コマンドを実行したときにプロキシ設定を自動的に表示する簡単な Zsh スクリプトを作成しました。

プロキシ設定を表示する理由

プロキシと VPN は、外部リソースに安全にアクセスするために不可欠です。ネットワークに依存するコマンドを実行する前にプロキシ設定を表示することで、接続の問題を迅速に特定し、トラブルシューティングを行うことができます。

スクリプト

このスクリプトは、Zsh の `preexec` 関数を利用して、実行しようとしているコマンドがネットワークに依存するかどうかをチェックします。もし依存しており、かつプロキシ環境変数が設定されている場合、現在のプロキシ設定を表示します。

```
# 特定のコマンド実行前にプロキシ設定を確認して表示する関数
preexec() {
    # ネットワーク依存のコマンドを定義
    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"
        # 必要に応じてさらにコマンドを追加
    )

# コマンドラインから最初の単語（コマンド）を抽出する
local cmd
cmd=$(echo "$1" | awk '{print $1}')

# プロキシ変数を表示する関数
display_proxy() {
    echo -e "\n  検出されたプロキシ設定:"
```

```
[ -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 ""

    # コマンドがネットワーク依存かどうかをチェック
    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
}
```

Zsh でのスクリプト設定

1. .zshrc ファイルを開く

お好みのテキストエディタを使用して、.zshrc 設定ファイルを開きます。例えば：

```
nano ~/.zshrc
```

2. preexec 関数を追加する

上記のスクリプトをファイルの末尾に貼り付けてください。

3. 保存して閉じる

CTRL + O を押して保存し、CTRL + X を押して終了します。

4. 変更を適用する

新しい設定をすぐに適用するために、`.zshrc` をリロードします:

```
source ~/.zshrc
```

セットアップのテスト

1. プロキシを有効にする場合

一時的にプロキシ変数を設定し、`pip` を使用してネットワーク依存のコマンドを実行する:

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

このコードは、HTTP プロキシを設定し、`selenium`、`beautifulsoup4`、`urllib3` という Python パッケージをインストールするためのコマンドです。プロキシの設定は、`HTTP_PROXY` 環境変数に `http://127.0.0.1:7890` を指定しています。その後、`pip` コマンドを使って指定されたパッケージをインストールしています。

期待される出力:

☒ プロキシ設定が検出されました: - 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

```
seleniumの収集
  selenium-4.x.x-py2.py3-none-any.whlをダウンロード中 (xxx kB)
beautifulsoup4の収集
  beautifulsoup4-4.x.x-py3-none-any.whlをダウンロード中 (xxx kB)
urllib3の収集
  urllib3-1.x.x-py2.py3-none-any.whlをダウンロード中 (xxx kB)
...
```

2. プロキシが有効でない場合

プロキシ変数を解除し、同じ `pip` コマンドを実行します:

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

このコードは、HTTP プロキシの設定を解除し、`selenium`、`beautifulsoup4`、`urllib3` という Python パッケージをインストールするためのコマンドです。

期待される出力:

```
selenium を収集中
  selenium-4.x.x-py2.py3-none-any.whl をダウンロード中 (xxx kB)
beautifulsoup4 を収集中
  beautifulsoup4-4.x.x-py3-none-any.whl をダウンロード中 (xxx kB)
urllib3 を収集中
  urllib3-1.x.x-py2.py3-none-any.whl をダウンロード中 (xxx kB)
...
```

(プロキシ通知は表示されないはずです。)

3. 非ネットワークコマンド

ローカルコマンド (例: `ls`) を実行する:

```
ls
```

期待される出力:

```
[ファイルとディレクトリのリスト]
```

(プロキシ通知は表示されないはずです。)

カスタマイズ

- `network_commands` の拡張: ネットワークに依存する追加のコマンドを `network_commands` 配列に追加します。

- エイリアスの処理: ネットワーク依存のコマンドに対するエイリアスが network_commands リストに含まれていることを確認します。

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

このコードは、gpa というエイリアスを定義しています。このエイリアスを使用すると、git push all というコマンドを簡単に実行できます。git push all は、通常、すべてのリモートリポジトリに変更をプッシュするためのコマンドです。ただし、git push all というコマンドは標準の Git コマンドではなく、カスタムスクリプトや設定によって定義されている可能性があります。

"gpa" を network_commands 配列に追加して、このエイリアスを使用する際にプロキシ通知をトリガーします。

- 色で視認性を向上させる:

特に混雑したターミナルでの視認性を向上させるために、プロキシ通知に色を追加することができます：

```
  # .zshrc の先頭にカラーコードを追加
  GREEN='\033[0;32m'
  NC='\033[0m' # カラーなし

display_proxy() {
  echo -e "\n${GREEN}  プロキシ設定が検出されました:${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

プロキシ設定の管理は、制限されたネットワーク環境でのソフトウェア開発を円滑に行うために重要です。この Zsh スクリプトは、ネットワークアクセスを必要とするコマンドを実行する際

に、常にプロキシ設定を確認できるようにし、ワークフローとトラブルシューティングの効率を向上させます。

ハッピーコーディング！