# Let Zsh Display Proxy Settings Before Running Network Commands

`<img class="responsive" src="/assets/images/preexec/pe1.png" alt="prexec" />`

Living in China or working within companies that use VPNs and proxies can complicate software development. Forgetting to configure these settings often leads to connectivity issues. To streamline your workflow, I created a simple Zsh script with the help of ChatGPT that automatically displays your proxy settings when you run specific network-dependent commands.

## Why Display Proxy Settings?

Proxies and VPNs are essential for accessing external resources securely. Displaying your proxy settings before executing network-dependent commands helps you quickly identify and troubleshoot connectivity issues.

## The Script

This script utilizes Zsh' s `preexec` function to check if the upcoming command is network-dependent. If it is and proxy environment variables are set, it displays the current proxy settings.

```
# Function to check and display proxy settings before certain commands
preexec() {
    # Define network-dependent commands
    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"
    # Add more commands as needed
)

# Extract the first word (command) from the command line
local cmd
cmd=$(echo "$1" | awk '{print $1}')

# Function to display proxy variables
display_proxy() {
    echo -e "\n  **Proxy Settings Detected:**"

    [ -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 ""
    }

    # Check if the command is network-dependent
    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
}
```

## Setting Up the Script in Zsh

### 1. Open Your `.zshrc` File

Use your preferred text editor to open the `.zshrc` configuration file. For example:

```
nano ~/.zshrc
```

### 2. Add the `preexec` Function

Paste the script above at the end of the file.

### 3. Save and Close

Press `CTRL + O` to save and `CTRL + X` to exit.

#### 4. Apply the Changes

Reload your `.zshrc` to apply the new configuration immediately:

```
source ~/.zshrc
```

## Testing the Setup

### 1. With Proxy Enabled

Set a proxy variable temporarily and run a network-dependent command using `pip`:

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

**Expected Output:**

```

  **Proxy Settings Detected:**
   - 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. Without Proxy Enabled

Unset the proxy variable and run the same `pip` command:

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

**Expected Output:**

```
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)
...
```

*(No proxy notification should appear.)*

### 3. Non-Network Command

Run a local command like `ls`:

```
ls
```

**Expected Output:**

```
[List of files and directories]
```

*(No proxy notification should appear.)*

## Customization

- **Extend `network_commands`:** Add any additional network-dependent commands to the `network_commands` array.
- **Handle Aliases:** Ensure that any aliases for network-dependent commands are included in the `network_commands` list.

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

  Add `"gpa"` to the `network_commands` array to trigger proxy notifications when using this alias.
- **Enhance Visibility with Colors:**

  For better visibility, especially in cluttered terminals, you can add color to the proxy notifications:

  ```
  # Add color codes at the top of your .zshrc
  GREEN='\033[0;32m'
  NC='\033[0m' # No Color

  display_proxy() {
      echo -e "\n${GREEN} **Proxy Settings Detected:**${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 ""
    }
```

## Conclusion

Managing proxy settings is crucial for smooth software development in restricted network environments. This Zsh script ensures you’ re always informed about your proxy configurations when running commands that require network access, enhancing your workflow and troubleshooting efficiency.

**Happy Coding!**