

# Setting up Alpaca on MacOS

## Purpose

This guide will walk users through installing and setting up [Alpaca](#).

📖 This article is generic (not specific to Snyk) but it is a requirement for Snyk as it is for any other CLIs.

For any package installation, always try Self-Service first, as it has a dedicated team for better support. [Why writing a MacOS guide if Self-Service offers it?](#)

At the time of writing the MacOS article, for the MacOS SOE (not the EPZ one), JAMF On-Prem (the former Self-Service) **did** not have a functional Alpaca: It did not detect the PAC URL ([I fixed that in 2.0.11](#)) so all connections defaulted to DIRECT (where 95% should be forwarded to Prisma proxy). Then JAMF Cloud (the new/current Self-Service) **did** not have Alpaca packaged... but it now has!

📌 If you are a Windows user, head this way → [🏠 Setting up Alpaca on Windows](#)

This will help ensure a stable experience when running the Snyk CLI via Command Line and the Snyk IDE Extension.

---

## Installing Alpaca on different OSs

### On Mac OS

Guide adapted from the [Engineering Handbook Guide](#) (I'll need to update the Handbook too)

#### Step 1: Installing Alpaca

1. Run the command:

```
1 brew tap samuong/alpaca
2 brew install samuong/alpaca/alpaca
```

⚠️ Do not install it via Self Service because you then won't be able to make it a daemon (autostart)

Or install it via Self-Service and build your own Daemon, by skipping Step 2. and creating the whole file in Step 3.

## Step 2: Enable Alpaca Service (LaunchDaemon/Agent a.k.a autostart)

1. Run the command:

```
1 brew services start alpaca
```

## Step 3: Fix the daemon

There was a tiny (but pesky) bug on Alpaca, where it failed retrieving the PAC URL in MacOS. I've fixed it in version [2.0.11](#). Keeping the below for learnings (how to tweak homebrew).

~~In the meantime, while waiting for that fix, we need to tweak the Daemon and hardcode the PAC URL. To do so edit~~

```
~/Library/LaunchAgents/homebrew.mxcl.alpaca.plist
```

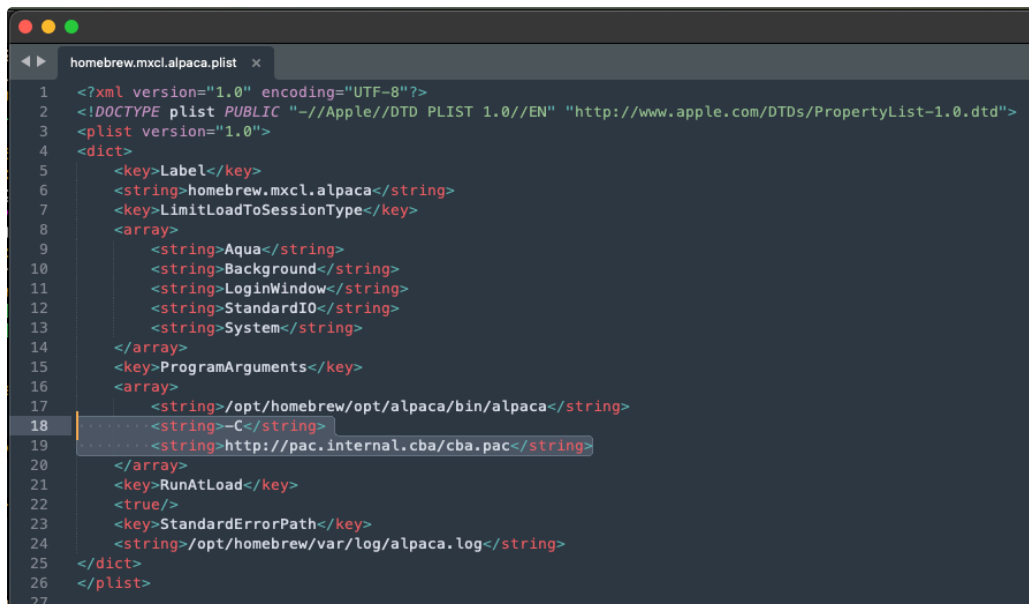
~~and add~~

```
<string>-C</string>
```

```
<string><http://pac.internal.cba/cba.pac</string>
```

~~⚠ This plist file will only exist if you ran Step 2. already.~~

~~Also it will be overwritten if you ever brew service stop Alpaca (do not)~~



```
homebrew.mxcl.alpaca.plist
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE plist PUBLIC "-//Apple/DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
3 <plist version="1.0">
4   <dict>
5     <key>Label</key>
6     <string>homebrew.mxcl.alpaca</string>
7     <key>LimitLoadToSessionType</key>
8     <array>
9       <string>Aqua</string>
10      <string>Background</string>
11      <string>LoginWindow</string>
12      <string>StandardIO</string>
13      <string>System</string>
14    </array>
15    <key>ProgramArguments</key>
16    <array>
17      <string>/opt/homebrew/opt/alpaca/bin/alpaca</string>
18      <string>-C</string>
19      <string>http://pac.internal.cba/cba.pac</string>
20    </array>
21    <key>RunAtLoad</key>
22    <true/>
23    <key>StandardErrorPath</key>
24    <string>/opt/homebrew/var/log/alpaca.log</string>
25  </dict>
26 </plist>
27
```

~~We do need to reload the Daemon once and for all:~~

```
1 launchctl unload ~/Library/LaunchAgents/homebrew.mxcl.alpaca.plist
2 launchctl load ~/Library/LaunchAgents/homebrew.mxcl.alpaca.plist
```

~~Once again DO NOT brew service stop alpaca or you'll need to tweak the daemon-  
plist again...~~

~~Voila, your machine can reboot, and Alpaca will always run (persistent) making use of our PAC-  
file (unless our PAC URL changes)~~

#### Step 4: Instruct your CLI to use Alpaca (Persistent)

1. Open your `~/.zshrc` or `~/.bashrc` and copy and paste these commands into there.

```
1 # Alias to run Alpaca with pac file and login
2 alias run-alpaca="GODEBUG=netdns=go alpaca -C "http://pac.internal.cba/cba.pac" -u
  $(whoami) -d au"
3 export {all,http,https}_proxy=http://localhost:3128
4
5 # Alias to turn on and off local proxy (point to localhost:3128)
6 alias AlpacaProxyOn="export {all,http,https}_proxy=http://localhost:3128"
7 alias CharlesProxyOn="export {all,http,https}_proxy=http://localhost:8888"
8 alias PrismaProxyOn="export
  {all,http,https}_proxy=http://cba.proxy.prismaaccess.com:8080"
9 alias EDSProxyOn="export {all,http,https}_proxy=http://proxy.au.eds.com:8080"
10 alias ProxyOff="unset {all,http,https}_proxy"
```

More info on proxies: [CLI - Why you should not set CNTLM nor Prisma](#)

[Proxy and SSL Certificate issues](#)

#### Troubleshooting:

- Say alpaca doesn't work, or you just don't want it anymore:

```
1 brew service stop alpaca
2 brew uninstall alpaca
```

revert the changes in your `~/.zshrc` and point `{all,http,https}_proxy` to CNTLM or prisma ([cba.proxy.prismaaccess.com:8080](http://cba.proxy.prismaaccess.com:8080))

Understand, this is not ideal because you then no longer make use of the PAC file, so you need to manually set the `no_proxy` variable for your CLI to know when to go through the corporate proxy and when not to...

- If you do not want a persistent proxy set, you can comment line 3 in Step 4:

`export {all,http,https}_proxy=http://localhost:3128` so you can invoke it manually.

- If you want an interactive session to see what's happening but the daemon is running, you can


stop it gracefully: `launchctl unload`

`~/Library/LaunchAgents/homebrew.mxcl.alpaca.plist`

or kill it: `pkill alpaca` and run it manually.

```
pkill alpaca
GODEBUG=netdns=go alpaca -C "http://pac.internal.cba/cba.pac" -u bidabefl -d au
Password (for au\bidabefl):
2025/05/15 19:25:17.166439 netmonitor.go:45: Network changes detected: [127.0.0.1/8 ::1/128 fe80::1/64 fe80::14c2:68ed:36a2:13c9/64 192.168.10.110/24
2025/05/15 19:25:17.166544 pacfetcher.go:98: Attempting to download PAC from http://pac.internal.cba/cba.pac
2025/05/15 19:25:17.642035 main.go:92: Listening on tcp6 localhost:3128
2025/05/15 19:25:17.642064 main.go:92: Listening on tcp4 localhost:3128
2025/05/15 19:26:33.836415 proxyfinder.go:135: [1] GET http://169.254.169.254/metadata/instance/compute via "PROXY cba.proxy.prismaaccess.com:8080"
2025/05/15 19:26:33.301154 proxyfinder.go:113: [2] POST https://westus-0.in.applicationinsights.azure.com/v2.1/track via "DIRECT"
2025/05/15 19:26:44.202185 proxyfinder.go:135: [3] GET http://169.254.169.254/metadata/instance/compute via "PROXY cba.proxy.prismaaccess.com:8080"
2025/05/15 19:26:44.304725 proxyfinder.go:113: [4] POST https://westus-0.in.applicationinsights.azure.com/v2.1/track via "DIRECT"
2025/05/15 19:26:51.473372 proxyfinder.go:113: [5] POST https://copilot-telemetry.githubusercontent.com/telemetry via "DIRECT"
2025/05/15 19:27:04.881574 proxyfinder.go:113: [6] POST https://copilot-telemetry.githubusercontent.com/telemetry via "DIRECT"
```

## On Windows

 Same as above roughly...

1. Download the Windows binary at: [Releases · samuong/alpaca](#)
2. Store it in the Dev folder to bypass Application whitelisting
3. Then run it with `./alpaca.exe`

a. If it does not detect the PAC URL, run it with

```
./alpaca.exe -C http://pac.internal.cba/cba.pac
```

4. If it works, optionally: make it a service, or create a background scheduled task (or play in CurrentVersion / Run in Regedit)
5. Set your Env Variables (no bash expansion here...):

```
all_proxy=http://localhost:3128
```

```
http_proxy=http://localhost:3128
```

```
https_proxy=http://localhost:3128
```

## Configure Proxy within VS Code

1. **Open VS Code:** Launch your Visual Studio Code application.
2. **Access Settings:** Go to the bottom left and click the cog icon, then click **Settings**.
3. **Search for Proxy Settings:** In the search bar at the top of the Settings UI, type `proxy`.
4. **Update HTTP Proxy:**

- Look for the setting labeled **HTTP: Proxy**.
  - Enter `http://localhost:3128` in the input field.
5. **Save Changes:** Your changes should be saved automatically. You can close the Settings UI.
  6. **Restart VS Code:** To ensure the new proxy settings take effect, restart Visual Studio Code.

Now your VS Code should be configured to use `http://localhost:3128` as the proxy.

### Step 3: Test your extensions work

1. If you have the Snyc Extension installed, try and run a scan via the extension, you should receive results from your Snyc scan.
2. If you have GitHub Copilot installed, try and type a sample chat message and you should receive a response.

## Further Information

### Available Guides

[Expand all](#) [Collapse all](#)

- [Snyk - General Availability Info](#)
- [🧐 Snyk - Cheat Sheet](#)
- › [📄 Snyk - Onboarding & Access Guides](#)
- › [📖 Snyk - General Guides](#)
- › [🩹 Snyk - Findings Management & Remediation](#)
- › [🏠 Snyk - Pipeline Scanning Guides](#)
- [🐪 Setting up Alpaca on MacOS](#)