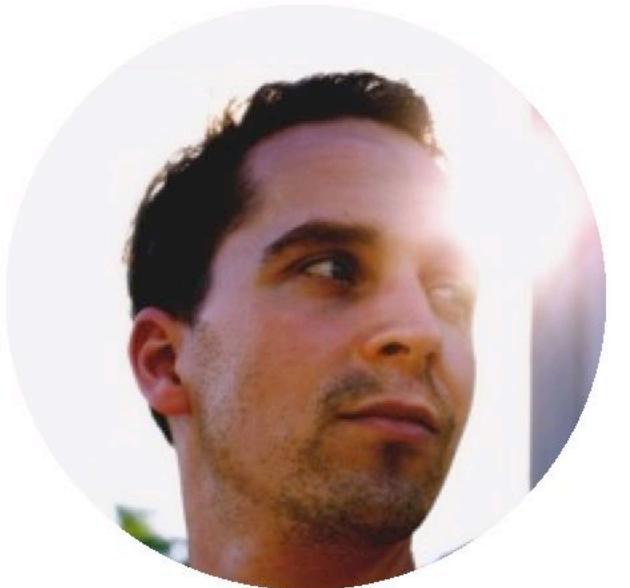


# Runtimes and Tools

Efforts to cope with ~~package &~~ version management



Henry Stamerjohann, MDO:YVR - June 16 2022

MAC  
DEV  
OPS  
YVR

G  a i

MAC  
DEV  
OPS  
YVR

Goal

**Create and maintain a local development environment**

# Who, why, and how?

**Create and maintain a local development environment**

MAC  
DEV  
OPS  
YVR

Administrators, Developers, DevOps

Day by day, collaboration, and evaluation

Take care of version conflicts and constraints

Tools help to manage runtimes and dependencies

# What is the issue here?

## Platform obstacles

- ✓ Linux - existing Package Managers
  - DPKG, RPM, Pacman, Zypper
- ✗ macOS - no built-in Package Manager
  - Historically grown approaches

# Packages, management?

Apple specific helpers



NixOS



Mac Ports



FINK



# MDO:YVR 2018: Let's talk about brew!



Mac Ports

guide.macports.org

### 3.1.11. port install

The action `install` is used to install a port. Once you determined the name of a port you want (possibly using [port search](#)), you can install it using this command. See [Section 3.2.1, “Invoking Variants”](#) on how to choose variants when installing a new port. For example,

```
$ sudo port install apache2 -preforkmpm +workermpm
```

installs the `apache2` port without the `preforkmpm`, but with the `workermpm` variant.

If the installation of a port fails, you can enable verbose or debug output by giving the `-v` or `-d` flag to `port`:

```
$ sudo port -v install apache2
```

All debug information is also kept in `main.log` for the port you installed. Its path will be printed automatically if the installation fails. You can manually get the path using `port logfile portname`. Note that logfiles will automatically be deleted on successful installation.

If the installation of a port fails, you should always clean and try again, i.e., run

```
$ sudo port clean portname
```

and re-execute the command you ran before.

You might also want to try enabling trace mode, which can prevent conflicts caused by files installed by other ports or in common system locations, such as `/usr/local`. To do that, re-run the installation with the `-t` flag, i.e.,

```
$ sudo port -t install portname
```

If the port still fails to install after you have followed these steps, please [file a ticket](#) and attach the `main.log` of a clean attempt.

**Note**

The installation of a single port consists of multiple phases. These phases are fetch, extract, patch, configure, build, destroot, archive, and finally install. You may break up a port's installation into smaller steps for troubleshooting by using the name of one of these phases as action rather than `install`. For example

```
$ sudo port destroot apache2
```

will run the installation of `apache2` until the `destroot` phase. See [Section 5.3, “Port Phases”](#) for a complete list of phases and a detailed description.

install takes the following switches:

**--no-rev-upgrade**

By default, a binary sanity check called `rev-upgrade` is run automatically after each successful installation. Pass this flag, if you want to avoid running this step, for example if you want to run it explicitly later after a number of installations using `sudo port rev-upgrade`, or if you know it will detect problems but want to defer dealing with them.

**--unrequested**

By default, each port you install using the `install` explicitly (contrary to ports installed as a dependency of a different port) is marked as “requested”. If you want MacPorts to treat a port you installed manually as if it was automatically installed as a dependency (e.g., if a dependency failed to build and you re-tried installing the dependency only), pass this flag.

### 3.1.12. port notes

The `notes` action is used to display any notes that a port's author included. These can contain anything, but by convention are brief, and typically contain quick start steps for configuring and using the port, pitfalls to watch out for, or other information that users should be aware of. These same notes are also displayed after installing a port. Many ports have no notes. More extensive documentation can often be found at a port's homepage, or in its installed files.

```
$ port notes xinit
```

```
---> xinit has the following notes:  
To use MacPorts' X11 as the default server, install xorg-server, log out, and  
log back in.
```

### 3.1.13. port clean

The `-l` option deletes intermediate files created by MacPorts while installing a port. A `port clean` is often necessary when builds fail and should be

Mac Ports Search for ports... Home Ports Builds Statistics Watchlist +

# py39-python-install

(python/py-py-pyInstall)

A simple, correct PEP427 wheel installer.

Version: 0.0.3 License: MIT GitHub

Details Builds Installation Stats Trac Tickets 0 Report an Issue with this port API Always skip to Details page

Maintainers	jmroot
Categories	python
Homepage	<a href="https://pypi.python.org/pypi/python-install/">https://pypi.python.org/pypi/python-install/</a>
Platforms	darwin
Variants	-

Support(s) (5)

[py310-python-install](#) [py36-python-install](#) [py37-python-install](#) [py38-python-install](#)  
[py-pyInstall](#)

"py39-python-install" depends on

[thon39](#)  
[ild \(3\)](#)  
[ang-9.0](#) [py-bootstrap-modules](#) [py39-setuptools](#)

Port Health:

✓ Monterey (arm64)	Files (45)
? Monterey (x86_64)	
✓ Big Sur (arm64)	Files (45)
✓ Big Sur (x86_64)	Files (45)
✓ Catalina	Files (45)
✓ Mojave	Files (45)
✓ High Sierra	Files (45)
✓ Sierra	Files (45)
✓ El Capitan	Files (45)
✓ Yosemite	Files (45)
✓ Mavericks	Files (45)
✓ Mountain Lion	Files (45)
✓ Lion	Files (45)
✓ Snow Leopard (x86_64)	Files (45)
✓ Snow Leopard (i386)	Files (45)
? Leopard (ppc legacy)	

? - No history in app's database

INSTALLATIONS (30 DAYS)  
66

REQUESTED INSTALLATIONS (30 DAYS)

Mac Ports

Back to nixos.org Packages Options Flakes Experimental

# Search more than 80 000 packages

python

Channel: 21.11 22.05 (Beta) unstable

Showing results 1-50 of 9894 packages.

Sort: Best match ▾

Package sets	
python38Packages	4750
python39Packages	4742
No package set	269
haskellPackages	33
emacs27Packages	21
kodiPackages	9
vimPlugins	8
pythonDocs	8
ocamlPackages	4
matrix-synapse-plugins	4
nodePackages_latest	3
nodePackages	3
gnome2	3
gnome	3
apacheHttpdPackages	3
clcklispPackagesClisp	2
over	2
oceanusPackages	2

**pythonFull**  
A high-level dynamically-typed programming language  
Name: [python](#) Version: 2.7.18 [Homepage](#) [Source](#) License: Python Software Foundation License version 2

**python38**  
A high-level dynamically-typed programming language  
Name: [python3](#) Version: 3.8.13 Outputs: [out](#) [debug](#) [Homepage](#) [Source](#)  
License: Python Software Foundation License version 2

▲▲▲ Hide package details ▲▲▲

**How to install python38?**

[On non-NixOS](#) [On NixOS](#)

```
$ nix-env -iA nixos.python38
```

Python is a remarkably powerful dynamic programming language that is used in a wide variety of application domains. Some of its key distinguishing features include: clear, readable syntax; strong introspection capabilities; intuitive object orientation; natural expression of procedural code; full modularity, supporting hierarchical packages; exception-based error handling; and very high level dynamic data types.

**Maintainers**

- Frederik Rietdijk <[fridh@fridh.nl](mailto:fridh@fridh.nl)>

**Platforms**

- aarch64-linux
- i686-linux
- x86\_64-linux
- x86\_64-darwin
- aarch64-darwin



The screenshot shows a web browser window displaying the Homebrew Documentation for Python. The page has a dark background with orange text and icons. At the top is a navigation bar with standard icons. Below it is a search bar labeled "Search Documentation". A large orange "Homebrew Documentation" title is centered, above which is a cartoon illustration of a beer mug with foam and an apple logo. In the top right corner, there's a "Fork me on GitHub" button with a ribbon icon. On the left side, there's a yellow rounded rectangle containing the "Homebrew" logo with its signature beer mug icon. On the right side, there's a yellow flashlight icon pointing upwards.

# Homebrew Documentation

Search Documentation

## Python

This page describes how Python is handled in Homebrew for users. See [Python for Formula Authors](#) for advice on writing formulae to install packages written in Python.

Homebrew should work with any **CPython** and defaults to the macOS system Python.

Homebrew provides formulae to brew Python 3.y. A `python@2` formula was provided until the end of 2019, at which point it was removed due to the Python 2 deprecation.

**Important:** If you choose to use a Python which isn't either of these two (system Python or brewed Python), the Homebrew team cannot support any breakage that may occur.

### Python 3.y

Homebrew provides formulae for maintained releases of Python 3.y (`python@3.y`).

**Important:** Python may be upgraded to a newer version at any time. Consider using a version manager such as `pyenv` if you require stability of minor or patch versions for virtual environments.

github.com

# Simple Python Version Management: pyenv

gitter join chat

pyenv lets you easily switch between multiple versions of Python. It's simple, unobtrusive, and follows the UNIX tradition of single-purpose tools that do one thing well.

This project was forked from [rbenv](#) and [ruby-build](#), and modified for Python.

```
$ pyenv versions
 2.7.10
* 3.5.0 (set by /Users/yuu/.pyenv/version)
  miniconda3-3.16.0
  pypy-2.6.0

$ python --version
Python 3.5.0

$ pyenv global pypy-2.6.0

$ python --version
Python 2.7.9 (295ee98b69288471b0fcf2e0ede82ce5209eb90b, Jun 01 2015, 17:30:13)
[PyPy 2.6.0 with GCC 4.9.2]

$ cd /Volumes/treasuredata/jupyter

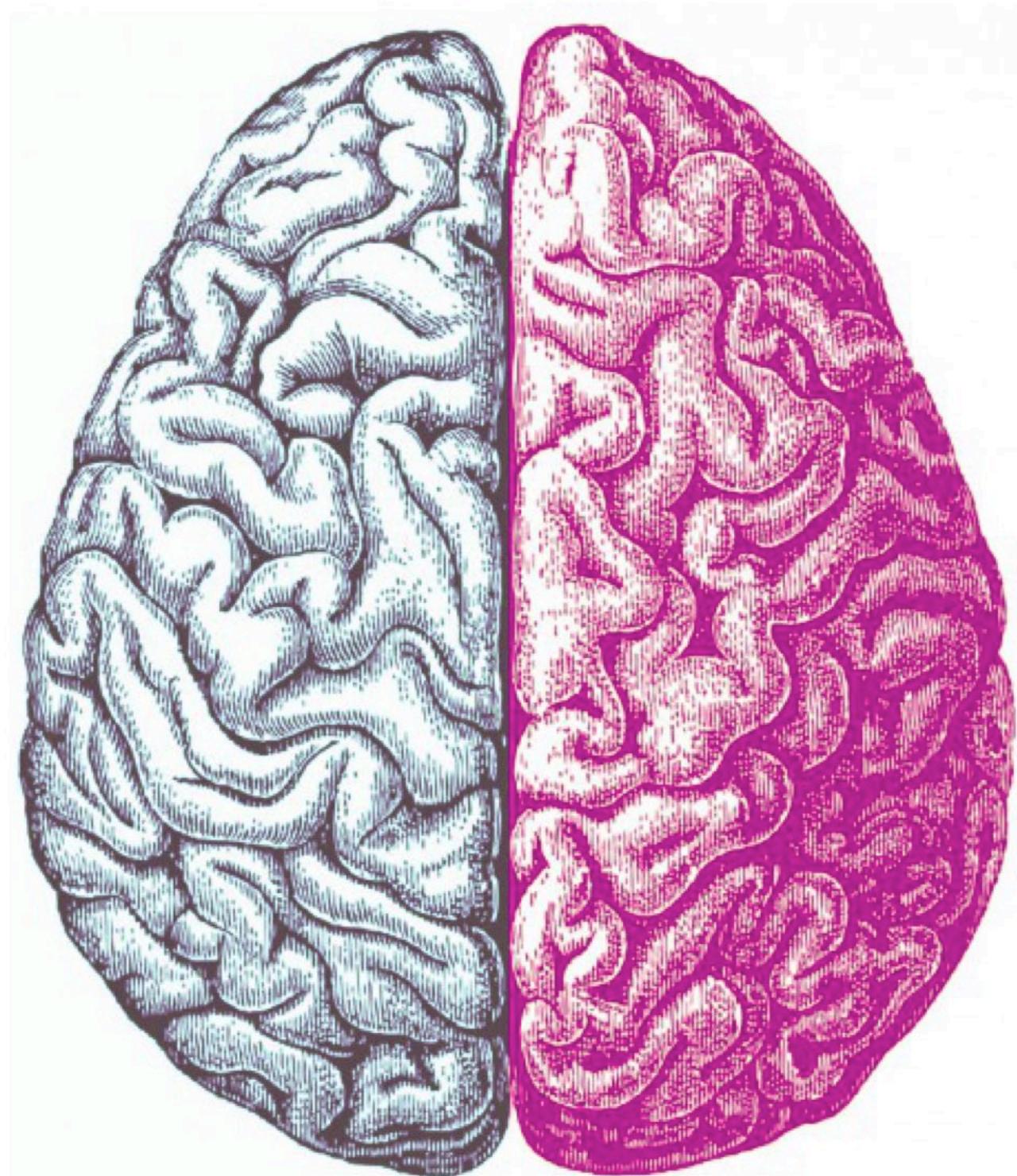
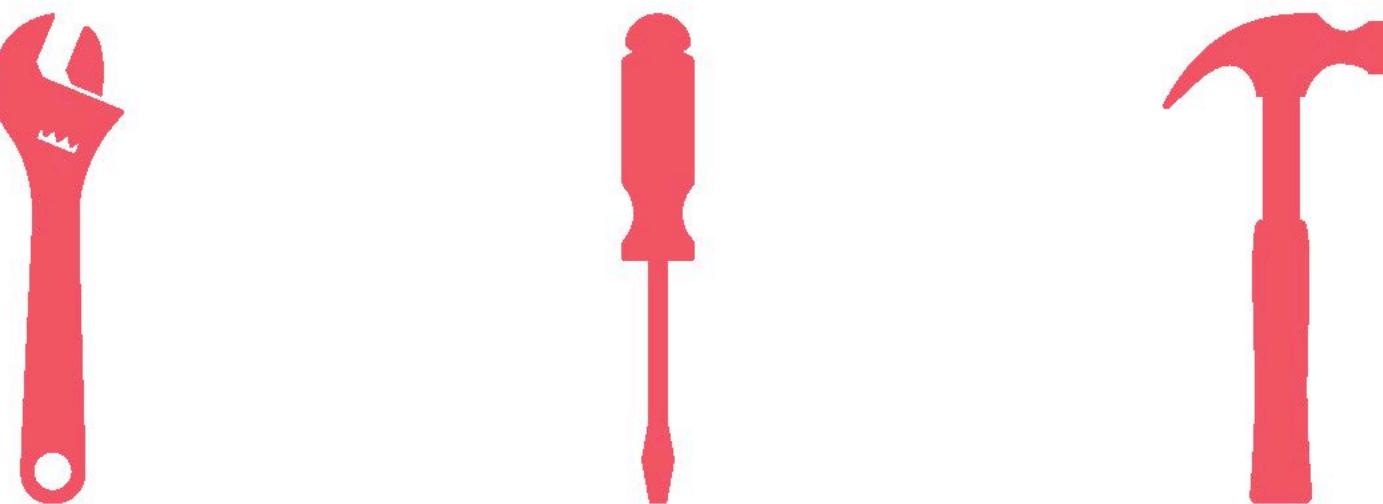
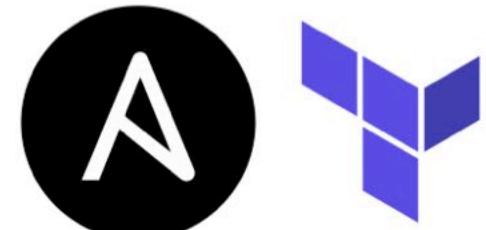
$ pyenv version
miniconda3-3.16.0 (set by /Volumes/treasuredata/.python-version)

$ python --version
Python 3.4.3 :: Continuum Analytics, Inc.
```

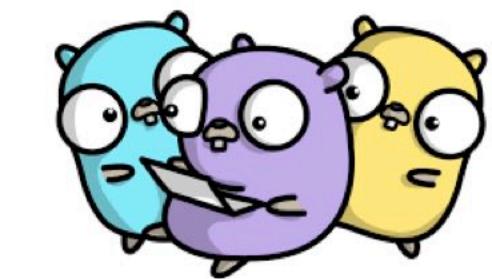
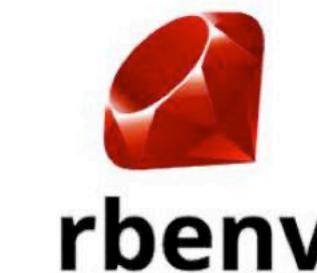
Languages

Language	Percentage
Python	51.6%
Java	0.1%

# Packages



# Versions +Dependencies



Pipenv

Poetry

Keep it stupid simple

The screenshot shows a web browser window with the URL `asdf-vm.com` in the address bar. The page content is as follows:

# asdf

Manage multiple runtime versions with a single CLI tool

[Get Started](#) [Introduction](#)

---

<b>One Tool</b> Manage each of your project runtimes with a single CLI tool and command interface.	<b>Plugins</b> Large ecosystem of existing runtimes & tools. Simple API to add support for new tools as you need!	<b>Backwards Compatible</b> Support for existing config files <code>.nvmrc</code> , <code>.node-version</code> , <code>.ruby-version</code> for smooth migration!
<b>One Config File</b> .tool-versions to manage all your tools, runtimes and their versions in a single, sharable place.	<b>Shells</b> Supports Bash, ZSH, Fish & Elvish with completions available.	<b>GitHub Actions</b> Provides a GitHub Action to install and utilize your <code>.tool-versions</code> in your CI/CD workflows.

asdf

Getting Started Reference ▾ Plugins ▾ Contribute ▾ Learn More ▾ Languages ▾ GitHub ▾ ☰ Search

## Homebrew

The Missing Package Manager for macOS (or Linux)

Homebrew manages your packages and their upstream dependencies. `asdf` does not manage upstream dependencies, it is not a package manager, that burden is upon the user, though we try and keep the dependency list small.

See [Homebrew docs](#) for more.

## NixOS

Nix is a tool that takes a unique approach to package management and system configuration

NixOS aims to build truly reproducible environments by managing exact versions of packages up the entire dependency tree of each tool, something `asdf` does not do. NixOS does this with its own programming language, many CLI tools and a package collection of over 60,000 packages.

Again, `asdf` does not manage upstream dependencies and is not a package manager.

See [NixOS docs](#) for more.

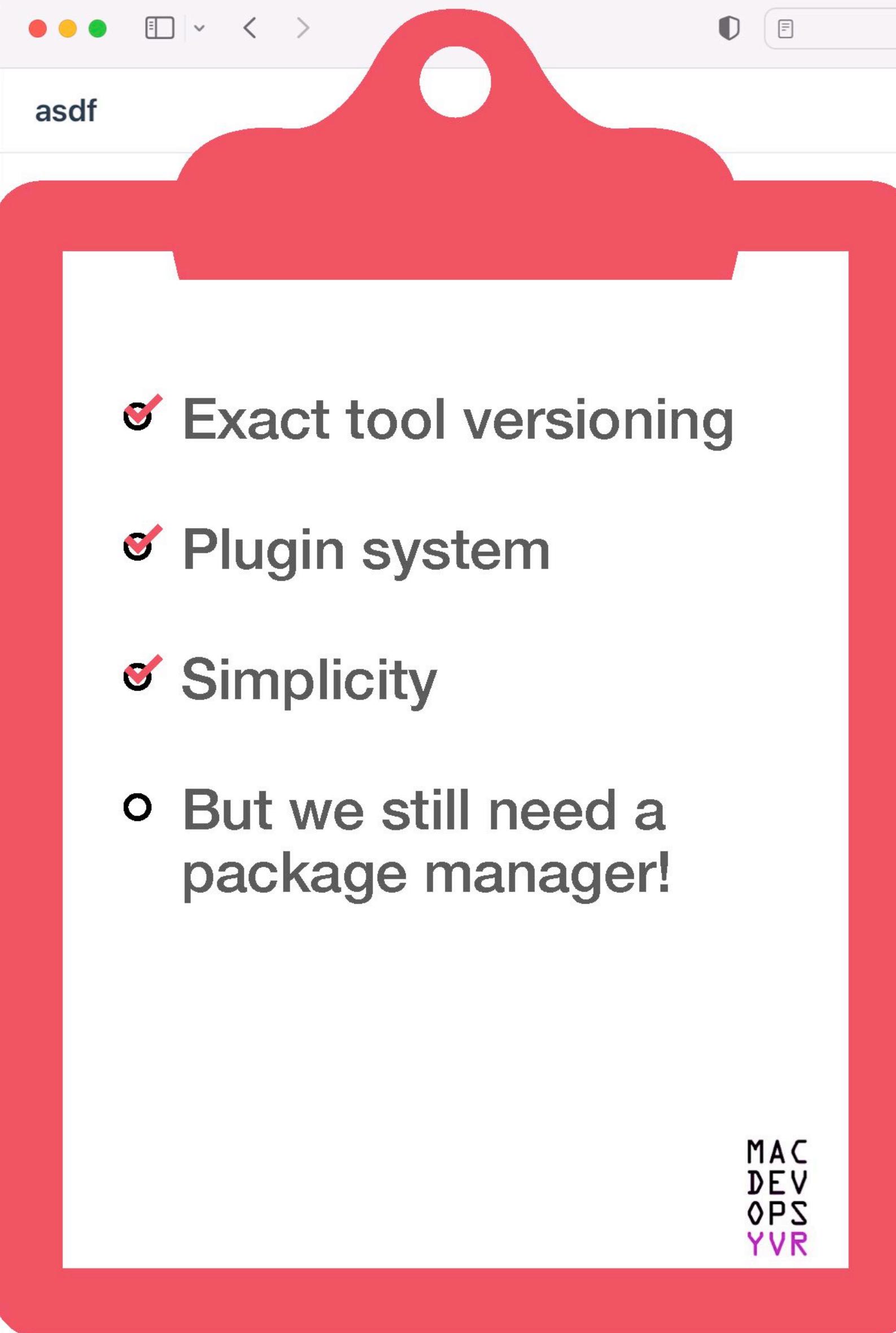
## Why use asdf?

`asdf` ensures teams are using the **exact** same versions of tools, with support for **many** tools via a plugin system, and the *simplicity and familiarity* of being a single **Shell** script you include in your Shell config.

**Note**

`asdf` is not intended to be a system package manager. It is a tool version manager. Just because you can create a plugin for any tool and manage its versions with `asdf`, does not mean that is the best course of action for that specific tool.





asdf

Getting Started Reference ▾ Plugins ▾ Contribute ▾ Learn More ▾ Languages ▾ GitHub ▾ ☰ Search

## Homebrew

The Missing Package Manager for macOS (or Linux)

Homebrew manages your packages and their upstream dependencies. `asdf` does not manage upstream dependencies, it is not a package manager, that burden is upon the user, though we try and keep the dependency list small.

See [Homebrew docs](#) for more.

## NixOS

Nix is a tool that takes a unique approach to package management and system configuration

NixOS aims to build truly reproducible environments by managing exact versions of packages up the entire dependency tree of each tool, something `asdf` does not do. NixOS does this with its own programming language, many CLI tools and a package collection of over 60,000 packages.

Again, `asdf` does not manage upstream dependencies and is not a package manager.

See [NixOS docs](#) for more.

## Why use asdf?

`asdf` ensures teams are using the **exact** same versions of tools, with support for **many** tools via a plugin system, and the *simplicity and familiarity* of being a single **Shell** script you include in your Shell config.

**Note**

`asdf` is not intended to be a system package manager. It is a tool version manager. Just because you can create a plugin for any tool and manage its versions with `asdf`, does not mean that is the best course of action for that specific tool.

asdf

Getting Started Reference ▾ Plugins ▾ Contribute ▾ Learn More ▾ Languages ▾ GitHub ▾ ☰ Search

**Introduction**

**Getting Started**

1. Install Dependencies

2. Download asdf

3. Install asdf

Core Installation Complete!

4. Install a Plugin

Plugin Dependencies

Install the Plugin

5. Install a Version

6. Set a Version

Global

Local

Using Existing Tool Version Files

Guide Complete!

asdf installation involves:

1. Installing dependencies
2. Downloading asdf core
3. Installing asdf
4. Installing a plugin for each tool/runtime you wish to manage
5. Installing a version of the tool/runtime
6. Setting global and project versions via .tool-versions config files

## 1. Install Dependencies

Linux:

**Note**

`sudo` may be required depending on your system configuration.

Package Manager	Command
Aptitude	<code>apt install curl git</code>
DNF	<code>dnf install curl git</code>
Pacman	<code>pacman -S curl git</code>
Zypper	<code>zypper install curl git</code>

macOS:

Package Manager	Command
Homebrew	Dependencies will be automatically installed by Homebrew.
Spack	<code>spack install coreutils curl git</code>



# Untangling Dependencies

Streamlined installation procedure



Method A - Homebrew

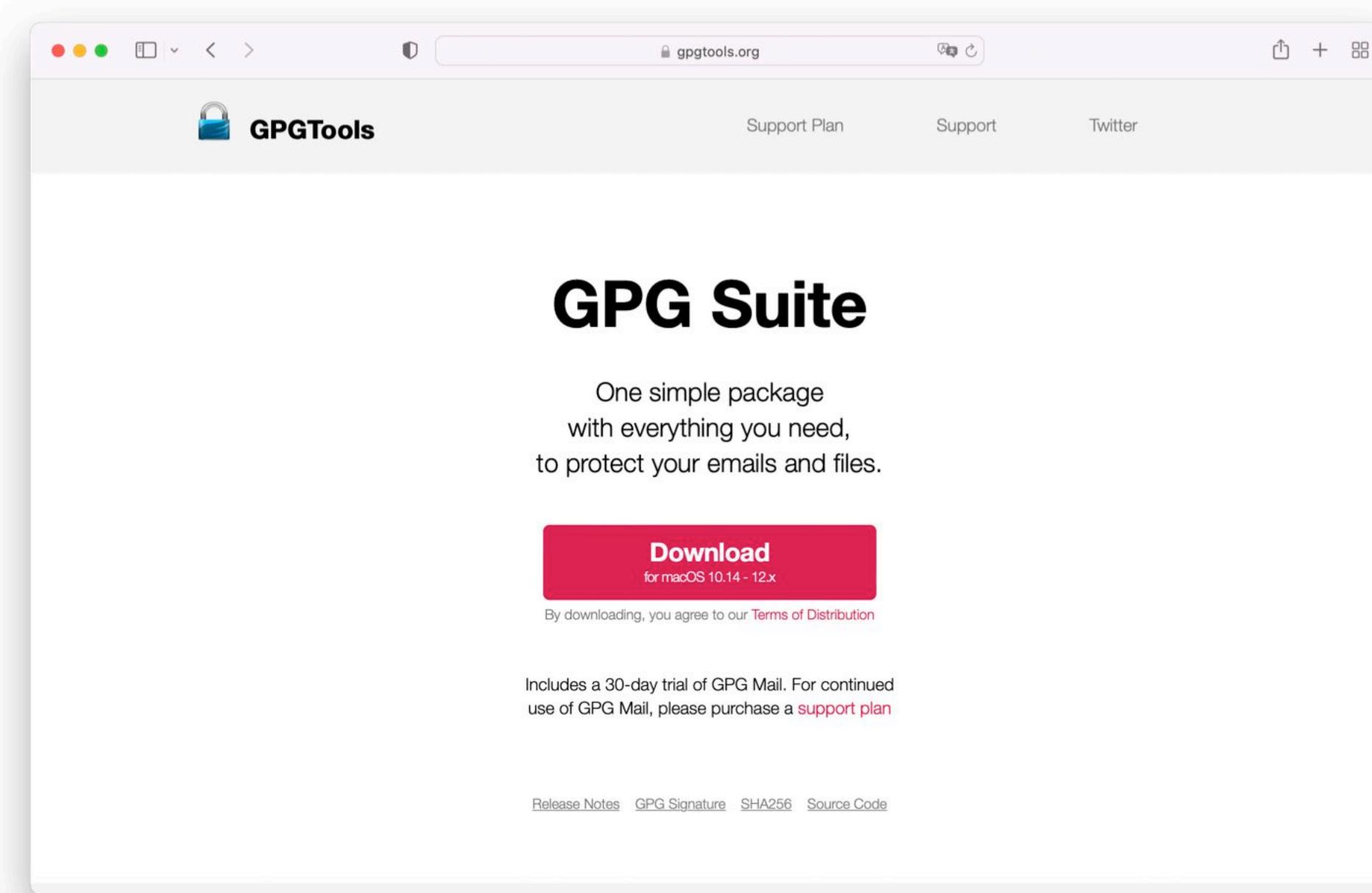
# Untangling Dependencies

## Streamlined installation procedure

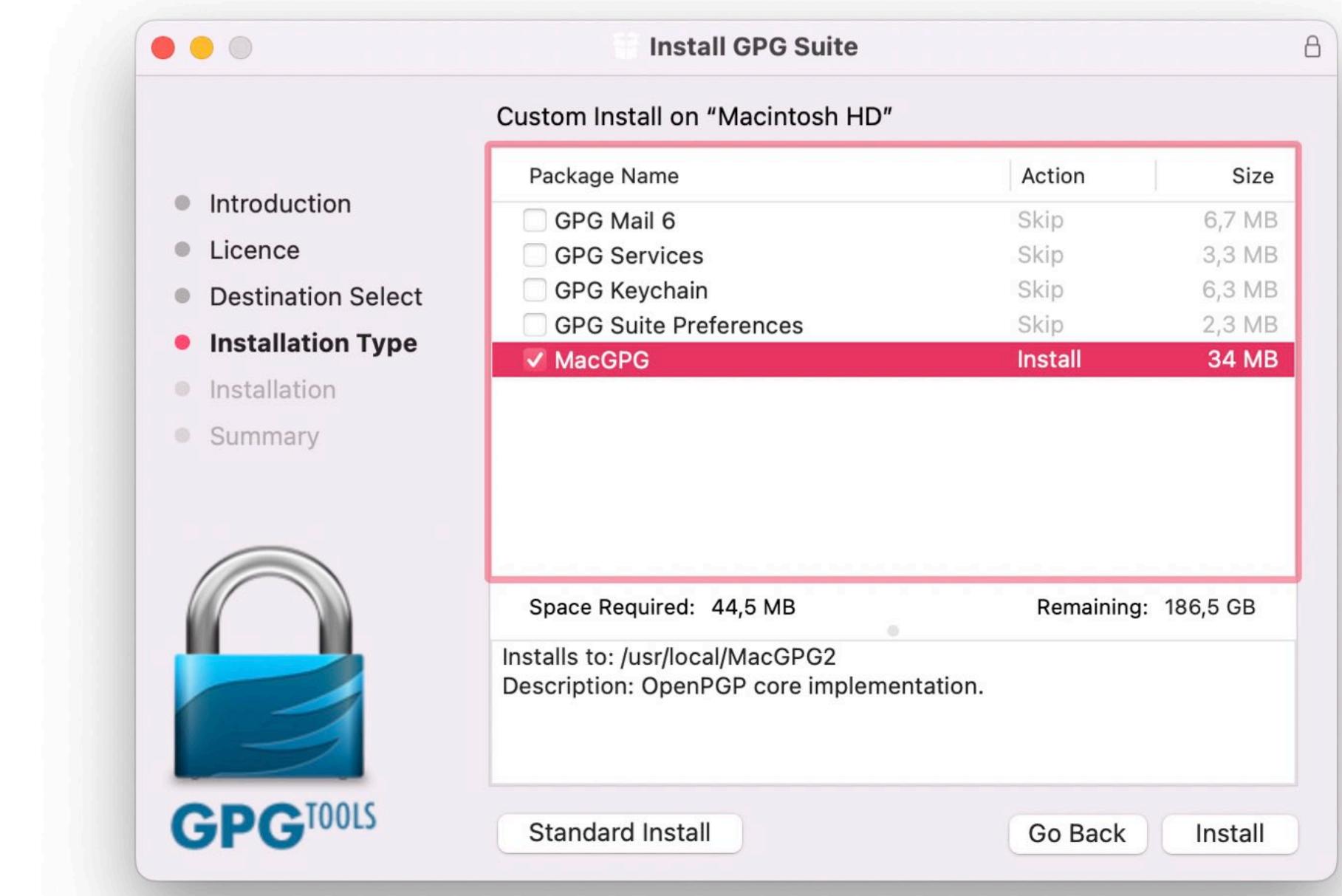
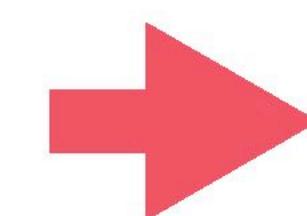


Method A - Homebrew

Method B - Utilize GPG Tools for the remainder dependency [coreutils]



The screenshot shows the GPG Suite download page on the GPGTools website. It features a large "GPG Suite" heading, a brief description of the suite, and a prominent red "Download" button. Below the download button, there's a note about a 30-day trial of GPG Mail and links to release notes, GPG Signature, SHA256, and Source Code.



The screenshot shows the "Install GPG Suite" window in the GPGTools application. It displays a "Custom Install on 'Macintosh HD'" dialog. On the left, a sidebar lists steps: Introduction, Licence, Destination Select, Installation Type (which is selected), Installation, and Summary. The main area shows a table of packages:

Package Name	Action	Size
GPG Mail 6	Skip	6,7 MB
GPG Services	Skip	3,3 MB
GPG Keychain	Skip	6,3 MB
GPG Suite Preferences	Skip	2,3 MB
<b>MacGPG</b>	<b>Install</b>	<b>34 MB</b>

Below the table, it says "Space Required: 44,5 MB" and "Remaining: 186,5 GB". The "MacGPG" row is highlighted with a pink box. At the bottom, there are buttons for "Standard Install", "Go Back", and "Install".

# Fast-track ...

## Installation & Setup

- 1) Install Xcode CLI tools
- 2) Install coreutils
- 3) Clone asdf repository
- 4) Add entries to `~/.zshrc`

```
● ● ●

# 1 - Install Xcode CLI tools
# 2 - Install coreutils via brew or GPG Tools

# 3 - clone asdf repository
git clone https://github.com/asdf-vm/asdf.git ~/.asdf --branch v0.10.1

# 4 - Add the following to ~/.zshrc
. $HOME/.asdf/asdf.sh
# append completions
fpath=(${ASDF_DIR}/completions $fpath)
# compinit ZSH completions
autoload -Uz compinit && compinit
```



# Manage Plugins

Efficient installation process

asdf plugin add <name>

asdf plugin add <name> <url>



```
# Add plugins via short-name
```

```
asdf plugin add python
```

```
asdf plugin add golang
```

```
asdf plugin add nodejs
```

```
asdf plugin add terraform
```

```
asdf plugin add poetry
```

```
# Add plugins via their Git URL
```

```
asdf plugin add cloudfared https://github.com/threkk/asdf-cloudfared.git
```

# Install runtime versions

## Download from source

asdf install <name> <vers>

asdf install <name> latest

```
● ● ●

# 1 - Install Xcode CLI tools
asdf install python 3.10.5
python-build 3.10.5 /Users/zentral/.asdf/install/python/3.10.5
Downloading openssl-1.1.1o.tar.gz ...
-> https://www.openssl.org/source/openssl-1.1.1o.tar.gz
Installing openssl-1.1.1o ...
Installed openssl-1.1.1o to /Users/zentral/.asdf/install/python/3.10.5

Downloading readline-8.1.tar.gz ...
-> https://ftpmirror.gnu.org/readline/readline-8.1.tar.gz
Installing readline-8.1 ...
Installed readline-8.1 to /Users/zentral/.asdf/install/python/3.10.5

Downloading Python-3.10.5.tar.xz ...
-> https://www.python.org/ftp/python/3.10.5/Python-3.10.5.tar.xz
Installing Python-3.10.5 ...
python-build: use zlib from xcode sdk
```

# Set version

Set global or local version

asdf global <name> <vers>

asdf local <name> <vers>



```
# set local installed version as global
asdf global python 3.10.5

# validate global set versions
cat /Users/zentral/.tool-versions
golang 1.18.3
nodejs 17.5.0
python 3.10.5
```

# 3 step process

It's simple as that

1) Install a Plugin

2) Install a Version\*

3) Set a Version

```
●●●  
# 1 - Add/update desired plugin  
asdf plugin add python  
updating plugin repository ... HEAD is now at a1c0673
```

\*optional: include default packages



So what's more to come?

# Folders structure

## Content

~/.asdf

```
• • •  
# directory structure  
tree ~/.asdf -L 1  
/Users/zentral/.asdf  
├── CHANGELOG.md  
├── CONTRIBUTING.md  
├── LICENSE  
├── README.md  
├── SECURITY.md  
├── asdf.elv  
├── asdf.fish  
├── asdf.sh  
├── bin  
├── completions  
├── defaults  
├── docs  
├── downloads  
├── help.txt  
└── installs ←  
├── lib  
├── plugins  
├── repository  
├── scripts  
├── shims  
├── test  
└── tmp  
version.txt
```

# Folder structure

## Versions

`~/.asdf/install/<name>`

```
tree ~/.asdf/install -L 2
/Users/zentral/.asdf/install
├── cloudfared
│   └── 2022.5.3
├── golang
│   └── 1.18.3
├── nodejs
│   ├── 17.5.0
│   └── 18.3.0
├── python
│   ├── 3.10.5
│   └── 3.9.13 ←
├── tart
│   └── 0.7.1
├── terraform
│   └── 1.2.2
└── terragrunt
    └── 0.37.1
```

# Folder structure

## Binaries and Libraries

Common directory structure

Runtime executable files/binary

Core dependencies

```
tree ~/.asdf/install/python/3.9.13 -L 2
/Users/zentral/.asdf/install/python/3.9.13
├── bin
├── etc
├── include
├── lib
├── openssl
└── readline
└── share
```

# Neat features

## Plugin specific default packages

\$HOME/.default-python-packages



## Pipenv



```
cat $HOME/.default-python-packages  
ansible  
pipenv
```

# Install runtime versions

## Locate versions

asdf list <name>

asdf list all <name>

```
# List Installed Versions
asdf list python
3.10.4
3.10.5
3.9.13

# List All Available Versions
asdf list all python
3.9.13
..
3.10.4
3.10.5
3.11.0b3
3.11-dev
3.12-dev

# Show Latest Stable Version
asdf latest python
3.10.5
```

# Update tooling

## Extra tools & dependencies

asdf current

asdf current		
cloudfared	2022.5.3	/Users/zentral/Desktop/Project-A/.tool-versions
gcloud	_____	No version is set. Run "asdf <global shell local> gcloud <version>"
golang	1.18.3	/Users/zentral/.tool-versions
nodejs	17.5.0	/Users/zentral/.tool-versions
poetry	_____	No version is set. Run "asdf <global shell local> poetry <version>"
python	3.10.5	/Users/zentral/.tool-versions
ripgrep	_____	No version is set. Run "asdf <global shell local> ripgrep <version>"
shellcheck	_____	No version is set. Run "asdf <global shell local> shellcheck <version>"
shfmt	_____	No version is set. Run "asdf <global shell local> shfmt <version>"
tart	0.7.1	/Users/zentral/.tool-versions
terraform	_____	No version is set. Run "asdf <global shell local> terraform <version>"
terragrunt	_____	No version is set. Run "asdf <global shell local> terragrunt <version>"

# Shim?

## Rescan dependencies

asdf reshim <name> <vers>

```
● ● ●

pip install scrapy
Collecting scrapy
  Downloading Scrapy-2.6.1-py2.py3-none-any.whl (264 kB)
    264.3/264.3 KB 3.5 MB/s eta 0:00:00

Collecting itemadapter>=0.1.0
  Downloading itemadapter-0.6.0-py3-none-any.whl (10 kB)
Requirement already satisfied: cryptography>=2.0 in
./asdf/install/python/3.10.5/lib/python3.10/site-packages (from scrapy) (37.0.2)

Collecting PyDispatcher>=2.0.5
  Downloading PyDispatcher-2.0.5.zip (47 kB)
    47.6/47.6 KB 1.6 MB/s eta 0:00:00

  Preparing metadata (setup.py) ... done
Collecting service-identity>=16.0.0
  Downloading service_identity-21.1.0-py2.py3-none-any.whl (12 kB)
Collecting cssselect>=0.9.1
  Downloading cssselect-1.1.0-py2.py3-none-any.whl (16 kB)
Collecting Twisted>=17.9.0
  Downloading Twisted-22.4.0-py3-none-any.whl (3.1 MB)
    3.1/3.1 MB 10.4 MB/s eta 0:00:00

Collecting pyOpenSSL>=16.2.0
  Downloading pyOpenSSL-22.0.0-py2.py3-none-any.whl (55 kB)
    55.8/55.8 KB 1.8 MB/s eta 0:00:00

Collecting w3lib>=1.17.0
  Downloading w3lib-1.22.0-py2.py3-none-any.whl (20 kB)
Collecting lxml>=3.5.0
  Downloading lxml-4.9.0.tar.gz (3.4 MB)
    3.4/3.4 MB 10.4 MB/s eta 0:00:00

  Preparing metadata (setup.py) ... done
```

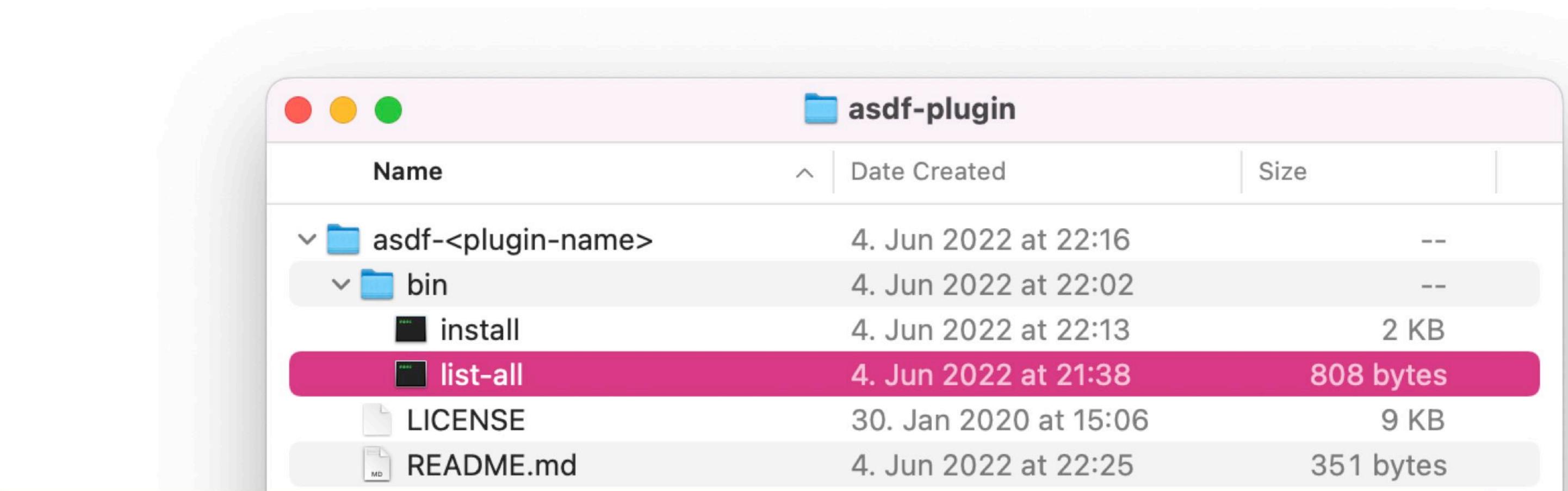
# Plugin ecosystem

## Easy to create own plugins

At least two shell scripts:

→ list-all

→ install



Name	Date Created	Size
asdf-<plugin-name>	4. Jun 2022 at 22:16	--
bin	4. Jun 2022 at 22:02	--
install	4. Jun 2022 at 22:13	2 KB
list-all	4. Jun 2022 at 21:38	808 bytes
LICENSE	30. Jan 2020 at 15:06	9 KB
README.md	4. Jun 2022 at 22:25	351 bytes



```
#!/usr/bin/env bash
github_coordinates=owner/reponame
releases_path="https://api.github.com/repos/${github_coordinates}/releases"
cmd="curl -sSL"
if [ -n "$GITHUB_API_TOKEN" ]; then
    cmd="$cmd -H 'Authorization: token $GITHUB_API_TOKEN'"
fi
cmd="$cmd $releases_path"
# stolen from https://github.com/rbenv/ruby-build/pull/631/files#diff-fdcfb8a18714b33b07529b7d02b54f1dR942
function sort_versions() {
    sed 'h; s/[+-]/./g; s/.p\([[:digit:]]*\)\.z\1/; s/$/.z/; G; s/\n/ /'
    LC_ALL=C sort -t. -k 1,1 -k 2,2n -k 3,3n -k 4,4n -k 5,5n | awk '{print $2}'
}
# Fetch all tag names, and get only second column. Then remove all unnecessary characters.
versions=$(eval "$cmd" | grep -oE "tag_name\" : \"[^\"]{1,15}\", " | sed 's/tag_name\" : \"//;s/\", //' | sort_versions)
# shellcheck disable=SC2086
echo $versions
rm -rf $HOME/.tool-versions $HOME/.asdfrc
```

# Update asdf built in command

asdf update --head

```
● ● ●

asdf update --head
remote: Enumerating objects: 282, done.
remote: Counting objects: 100% (282/282), done.
remote: Compressing objects: 100% (206/206), done.
remote: Total 282 (delta 126), reused 179 (delta 70), pack-reused 0
Receiving objects: 100% (282/282), 147.14 KiB | 3.27 MiB/s, done.
Resolving deltas: 100% (126/126), completed with 6 local objects.
From https://github.com/asdf-vm/asdf
 * branch           master      -> FETCH_HEAD
   aafe1e5..738306b  master      -> origin/master
Previous HEAD position was 9ee24a3 chore: release 0.9.0 (#994)
Branch 'master' set up to track remote branch 'master' from 'origin'.
Switched to a new branch 'master'
HEAD is now at 738306b fix: updating references to legacy github.io site (#1240)
Updated asdf to latest on the master branch
```

# Uninstall

## Simple removal task

```
rm -rf $HOME/.asdf
```

```
rm -rf $HOME/.tool-versions
```

```
# in ~/.zshrc remove source asdf.sh and completions
. $HOME/.asdf/asdf.sh
fpath=(${ASDF_DIR}/completions $fpath)
autoload -Uz compinit

# Remove $HOME/.asdf directory
rm -rf ${ASDF_DATA_DIR:-$HOME/.asdf}

# Remove all asdf config files
rm -rf $HOME/.tool-versions $HOME/.asdfrc
```

# Asdf man page

## Manage Plugins

# Asdf man page

## Manage Packages

```
● ● ●

asdf -h

MANAGE PACKAGES
asdf install                                     Install all the package versions listed in the .tool-versions file
asdf install <name>                                Install one tool at the version specified in the .tool-versions file
asdf install <name> <version>                      Install a specific version of a package
asdf install <name> latest[:<version>]            Install the latest stable version of a package
asdf uninstall <name> <version>                    Remove a specific version of a package
asdf current                                       Display current version set or being used for all packages
asdf current <name>                                 Display current version set or being used for package
asdf where <name> [<version>]                      Display install path for an installed or current version
asdf which <command>                                Display the path to an executable
asdf local <name> <version>                          Set the package local version
asdf local <name> latest[:<version>]                Set the package local version to the latest provided version
asdf global <name> <version>                          Set the package global version
asdf global <name> latest[:<version>]              Set the package global version to the latest provided version
asdf shell <name> <version>                           Set the package version to `ASDF_${LANG}_VERSION` in the current shell
asdf latest <name> [<version>]                      Show latest stable version of a package
asdf latest --all                                    Show latest stable version of all the packages and if they are installed
asdf list <name> [version]                            List installed versions of a package and optionally filter the versions
asdf list all <name> [<version>]                     List all versions of a package and optionally filter the returned version
asdf help <name> [<version>]                         Output documentation for plugin and tool
```

# Asdf man page

## Manage Utils

```
● ● ●

asdf -h

UTILS
asdf exec <command> [args ... ]           Executes the command shim for current version
asdf env <command> [util]                   Runs util (default: `env`) inside the environment used for command shim
execution.
asdf info                                     Print OS, Shell and ASDF debug information.
asdf reshim <name> <version>              Recreate shims for version of a package
asdf shim-versions <command>               List the plugins and versions that provide a command
asdf update                                    Update asdf to the latest stable release
asdf update --head                            Update asdf to the latest on the master branch
```

# asdf - brief summary

## A simple tool version manager

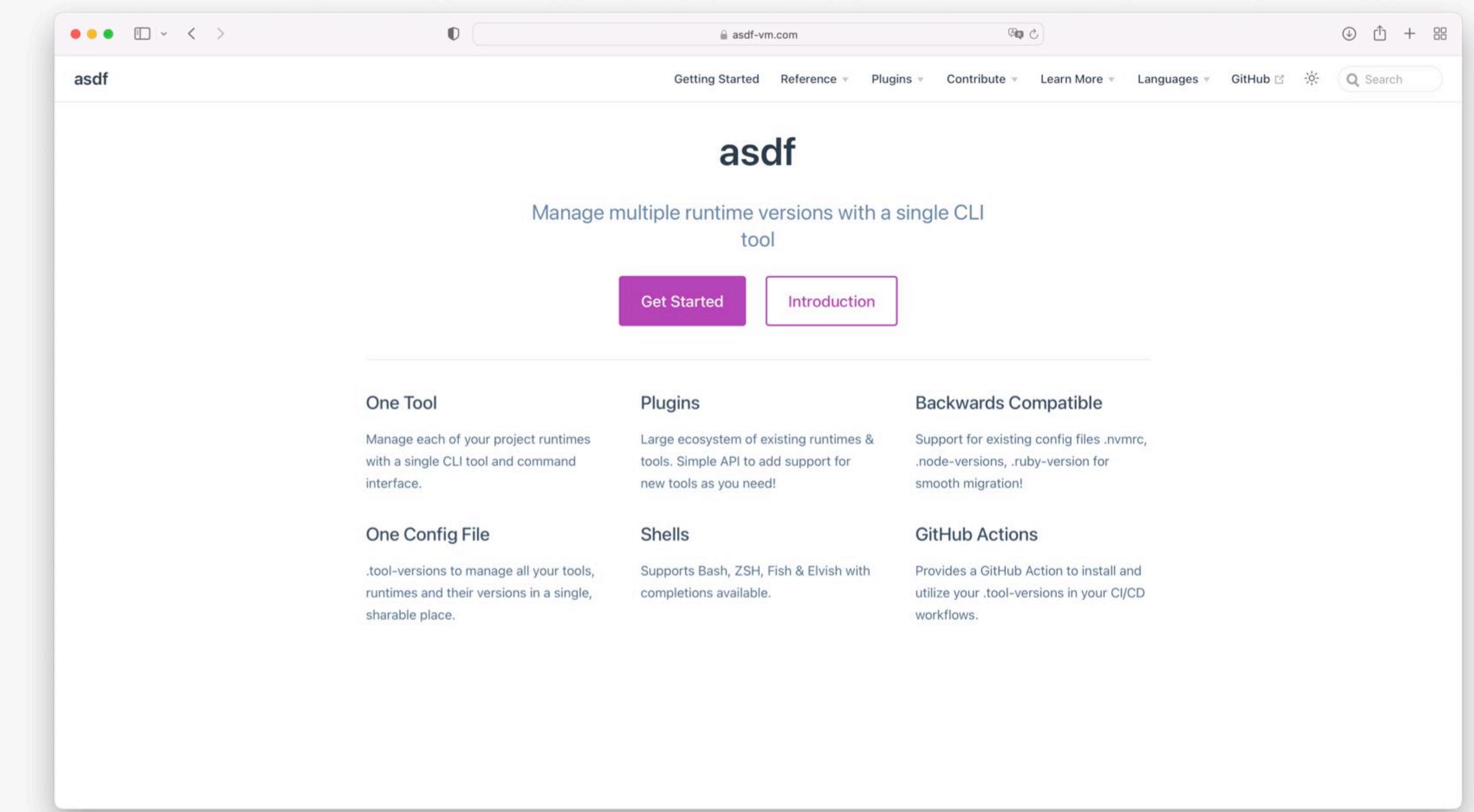
Super simple approach

Linux, macOS support

User-specific only

Plugin ecosystem

Global/local `.tool-versions` file



Install and switch runtime versions as you like!

# Caveats?

## Sufficient safeguarding

Supply chain control/hygiene

Pin to commits in source repository ([Plugins, etc.](#))

Auditing [\\$HOME/.asdf/](#) directory

- File integrity monitoring ([Osquery](#))
- Collect hashes, SPDX ([SBOMs](#))



# Links

## Learn more

Project: <https://asdf-vm.com>

Plugins repo: <https://github.com/asdf-vm/asdf-plugins>

GitHub Actions: <https://github.com/asdf-vm/actions>

- MDO:YVR: <https://blog.macadmin.me/posts/asdf-on-macos/>
- macOS: <https://www.wiserfirst.com/blog/how-to-use-asdf-on-macos/>
- Linux/ WSL: [https://www.joshfinnie.com/blog/setting\\_up\\_wsl\\_with\\_asdf/](https://www.joshfinnie.com/blog/setting_up_wsl_with_asdf/)

# Thank you

Slack: [@headmin\(henry\)](#)

Twitter: [@head\\_min](#)



Zentral: <https://zentral.io>

