# pipx Python app installer

## CONCEPTS

pipx is for applications (black, httpie), not libraries: each gets its own venv, only its commands land on PATH.

**pipx** manager
- **One venv per app** ~/.local/share/pipx/venvs
  - **App + its deps** isolated
    - **Command link** ~/.local/bin

Use pip or uv to add libraries to a project; use pipx for command-line tools you run everywhere.

## INSTALL

`pipx install <package>`
Install an app in its own venv

`pipx install <package> --python <python>`
Pick the Python version

`pipx install "<package>[<extra>]"`
Install with an extra

`pipx install git+<url>`
Install from a git repository

`pipx install ./<path>`
Install a local project

`pipx install --editable ./<path>`
Editable local install

`pipx install --include-deps <package>`
Also expose dependency commands

`pipx install --suffix @<tag> <package>`
Install a second copy side by side

`pipx install --force <package>`
Overwrite an existing install

`pipx install --pip-args="--pre" <package>`
Pass options to pip

`pipx install --global <package>`
Install for all users

## RUN WITHOUT INSTALLING

`pipx run <package>`
Run an app in a temporary venv

`pipx run <package> <args>`
Pass arguments to the app

`pipx run --spec "<package>==<version>" <command>`
Run a specific version

`pipx run --python <python> <package>`
Run on a chosen Python

`pipx run <script>.py`
Run a script with its inline dependencies

`pipx run git+<url>`
Run straight from git

## MANAGE

`pipx list`
Show installed apps and versions

`pipx list --short`
One line per app

`pipx upgrade <package>`
Upgrade one app

`pipx upgrade-all`
Upgrade every app

`pipx uninstall <package>`
Remove an app and its venv

`pipx uninstall-all`
Remove every app

`pipx reinstall <package>`
Rebuild the venv

`pipx reinstall-all`
Rebuild every venv

`pipx upgrade --include-injected <package>`
Upgrade injected libraries too

## INJECT AND PIN

`pipx inject <package> <dep>`
Add a library to an app's venv

`pipx inject <package> -r <file>`
Inject from a requirements file

`pipx uninject <package> <dep>`
Remove an injected library

## INJECT AND PIN (cont.)

`pipx runpip <package> <pip-args>`
Run pip inside an app's venv

`pipx pin <package>`
Block upgrades of an app

`pipx unpin <package>`
Allow upgrades again

## SETUP AND ENVIRONMENT

`pipx ensurepath`
Put the app directory on PATH

`python3 -m pip install --user pipx`
Install pipx itself

`pipx upgrade pipx`
Update a self-installed pipx

`pipx environment`
Show paths and settings

`pipx environment --value <name>`
Print one value

### Environment variables

| Variable | Default |
|---|---|
| PIPX_HOME | ~/.local/share/pipx |
| PIPX_BIN_DIR | ~/.local/bin |
| PIPX_DEFAULT_PYTHON | Python running pipx |
| PIPX_GLOBAL_HOME | /opt/pipx |

On Debian/Ubuntu 23.04+ and Fedora, pip install --user is blocked; install pipx from the system package manager.

## RECIPES

### First setup on Ubuntu

```
sudo apt install pipx
pipx ensurepath
# open a new terminal
pipx install black
```

### Ansible with extra modules

```
pipx install --include-deps ansible
pipx inject ansible <extra-package>
```

### Move tools to a new Python

```
pipx reinstall-all --python python3.13
```