# Cross-platform Package Management for Modern C++ Development with Pixi

RUBEN ARTS

Cppcon
The C++ Conference
2025
September 13 - 19

## About Us

- Europe based Startup
- 3 Years old
- Solving our own problems
- Focus on package management

Prefix.dev

# Pixi is not a C++ package manager

Nor Python or Rust or any language

# Why should I still care about Pixi?

Generic package management

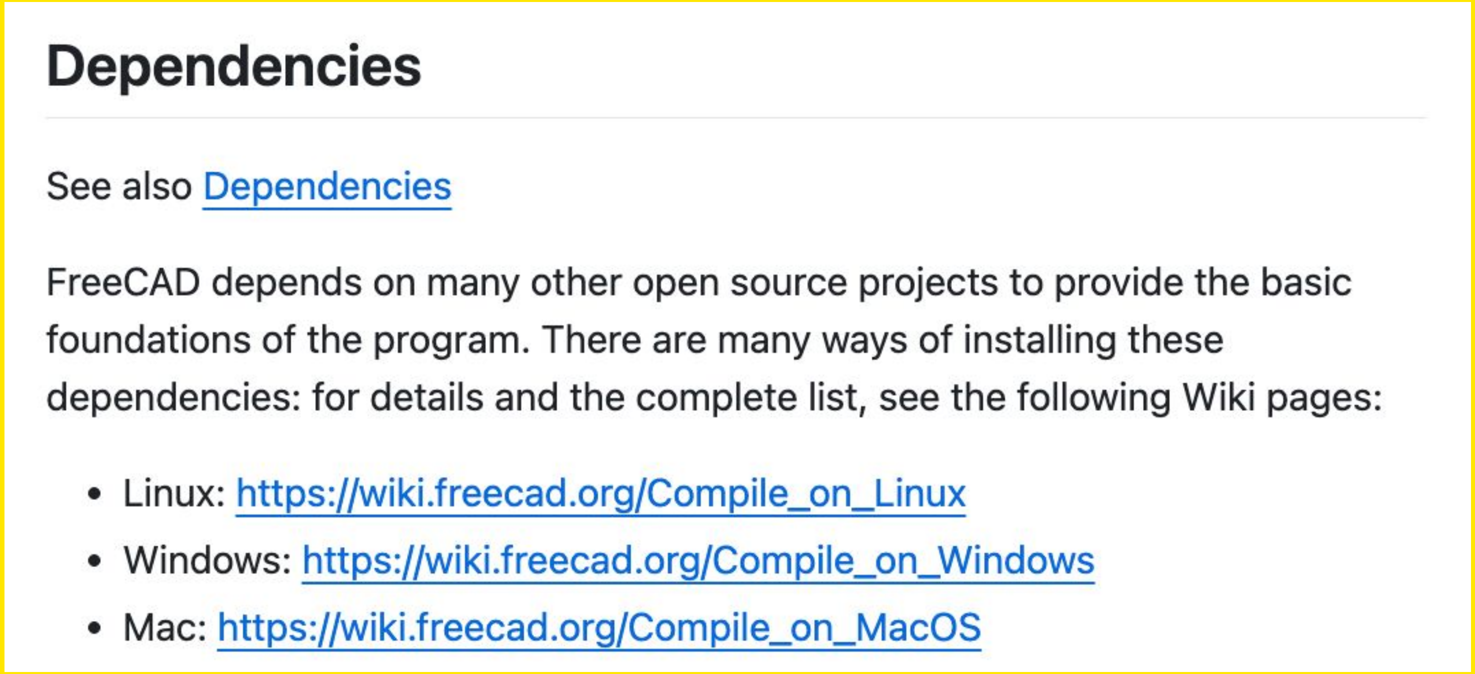
Support multiple languages

Support for Windows, macOS, and Linux

Binary and Source dependencies

Reproducibility

FreeCAD



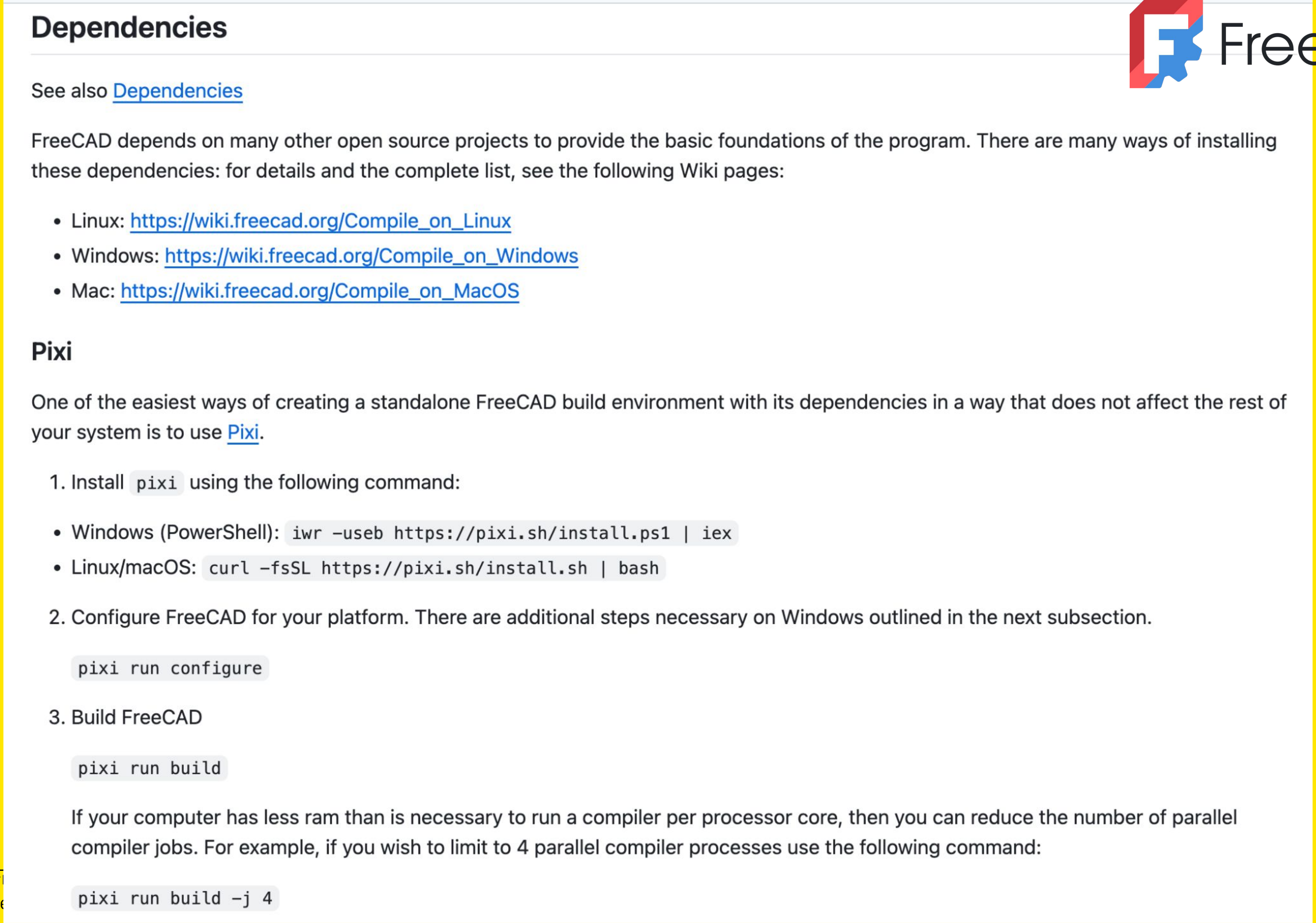
## Dependencies

See also Dependencies

FreeCAD depends on many other open source projects to provide the basic foundations of the program. There are many ways of installing these dependencies: for details and the complete list, see the following Wiki pages:

- Linux: https://wiki.freecad.org/Compile_on_Linux
- Windows: https://wiki.freecad.org/Compile_on_Windows
- Mac: https://wiki.freecad.org/Compile_on_MacOS

https://github.com/FreeCAD/DevelopersHandbook/blob/main/gettingstarted/index.md
https://upload.wikimedia.org/wikipedia/commons/a/a9/Freecad_logo_large.svg



## Dependencies

See also Dependencies

FreeCAD depends on many other open source projects to provide the basic foundations of the program. There are many ways of installing these dependencies: for details and the complete list, see the following Wiki pages:

- Linux: https://wiki.freecad.org/Compile_on_Linux
- Windows: https://wiki.freecad.org/Compile_on_Windows
- Mac: https://wiki.freecad.org/Compile_on_MacOS

### Pixi

One of the easiest ways of creating a standalone FreeCAD build environment with its dependencies in a way that does not affect the rest of your system is to use Pixi.

1. Install `pixi` using the following command:

- Windows (PowerShell): `iwr -useb https://pixi.sh/install.ps1 | iex`
- Linux/macOS: `curl -fsSL https://pixi.sh/install.sh | bash`

2. Configure FreeCAD for your platform. There are additional steps necessary on Windows outlined in the next subsection.

```
pixi run configure
```

3. Build FreeCAD

```
pixi run build
```

If your computer has less ram than is necessary to run a compiler per processor core, then you can reduce the number of parallel compiler jobs. For example, if you wish to limit to 4 parallel compiler processes use the following command:

```
pixi run build -j 4
```

https://github.com/F
https://upload.wikime

## Pixi global

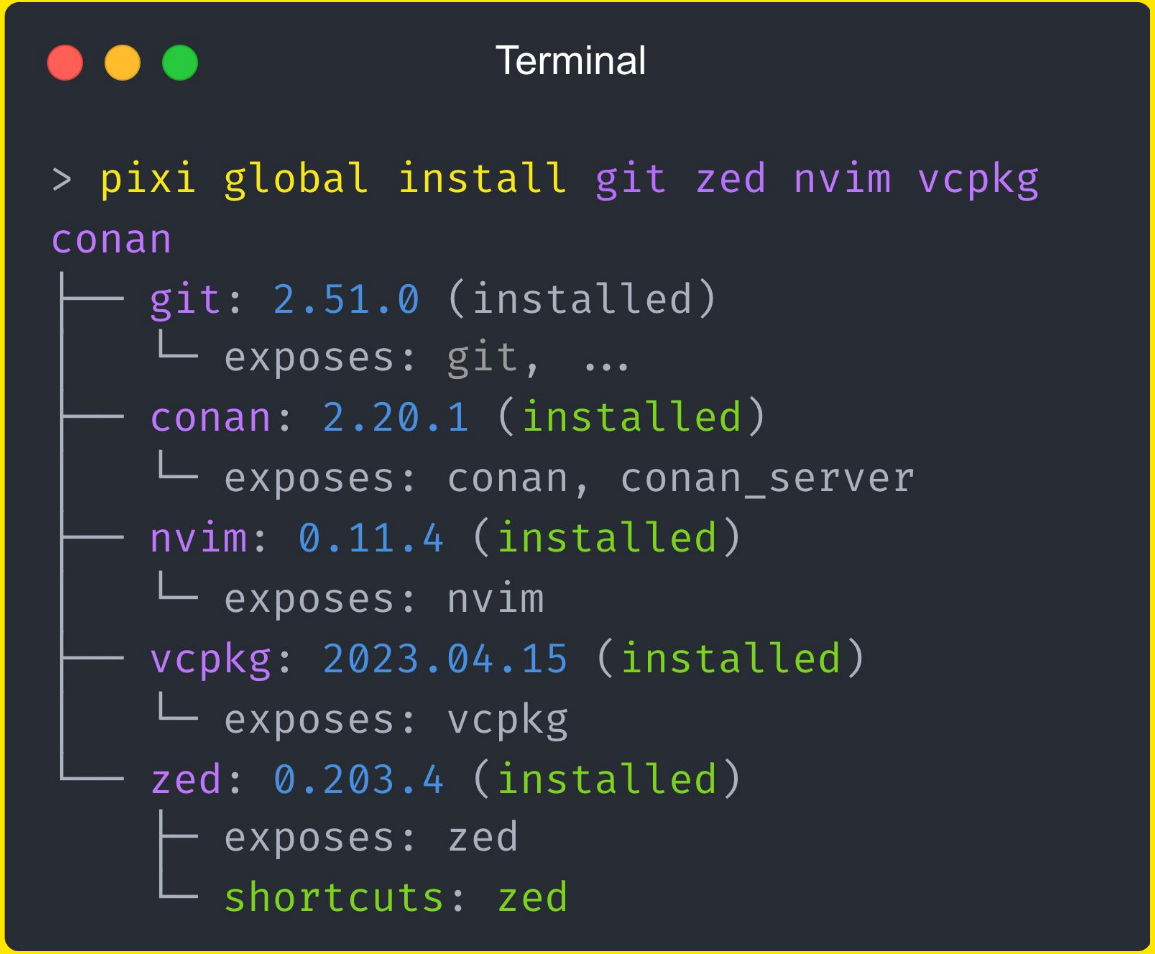
Isolated

No activation

Shortcuts

Autocompletion

Tools



```
> pixi global install git zed nvim vcpkg
conan
├── git: 2.51.0 (installed)
│   └─ exposes: git, ...
├── conan: 2.20.1 (installed)
│   └─ exposes: conan, conan_server
├── nvim: 0.11.4 (installed)
│   └─ exposes: nvim
├── vcpkg: 2023.04.15 (installed)
│   └─ exposes: vcpkg
└── zed: 0.203.4 (installed)
    ├─ exposes: zed
    └─ shortcuts: zed
```

## Pixi workspaces

Demo

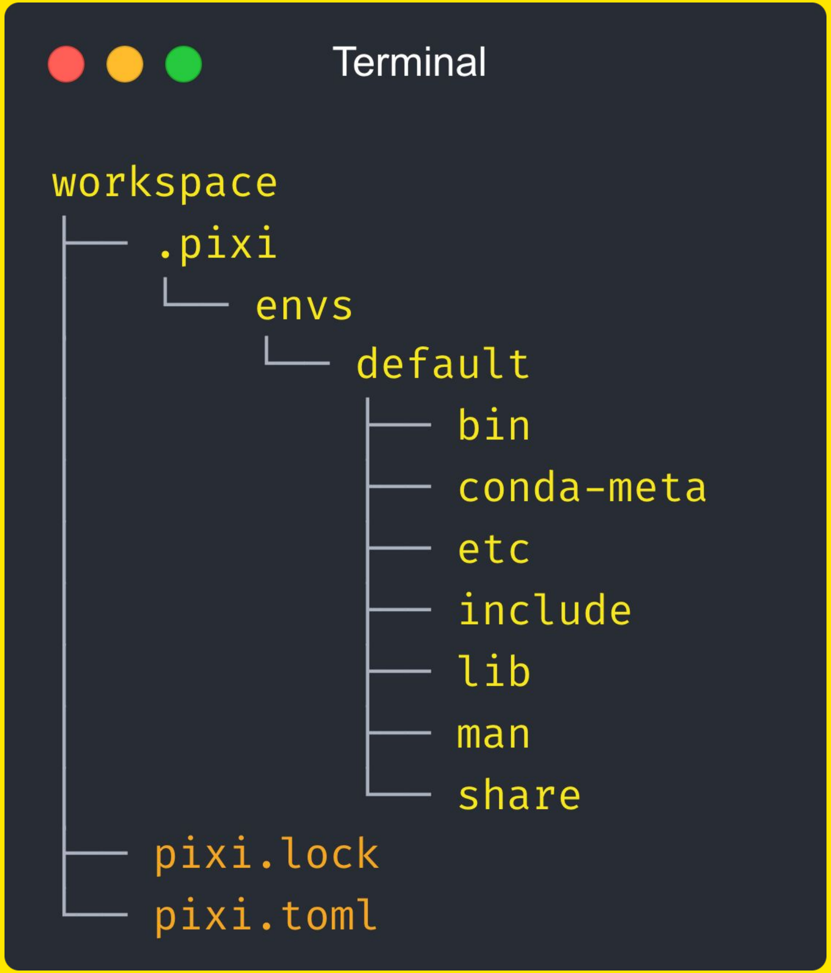
## Environments

No global installation

Requires activation

- pixi run
- pixi shell

Linked from cache

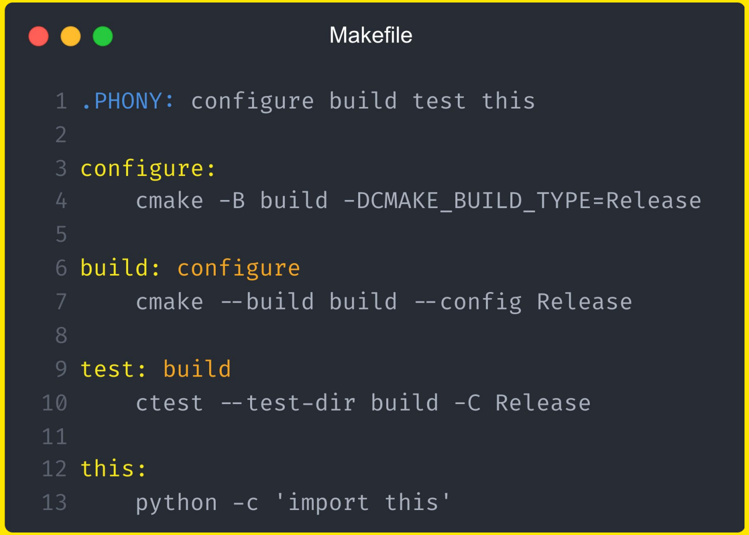


```
workspace
├── .pixi
│   └── envs
│       └── default
│           ├── bin
│           ├── conda-meta
│           ├── etc
│           ├── include
│           ├── lib
│           ├── man
│           └── share
├── pixi.lock
└── pixi.toml
```

# Tasks

Makefile but cross-platform

Shipping integrated shell


Makefile

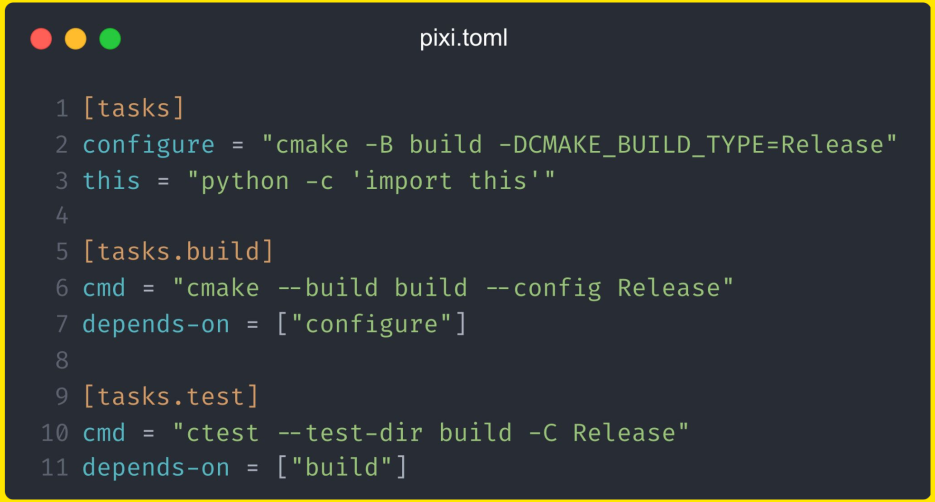
```
.PHONY: configure build test this

configure:
	cmake -B build -DCMAKE_BUILD_TYPE=Release

build: configure
	cmake --build build --config Release

test: build
	ctest --test-dir build -C Release

this:
	python -c 'import this'
```


pixi.toml

```
[tasks]
configure = "cmake -B build -DCMAKE_BUILD_TYPE=Release"
this = "python -c 'import this'"

[tasks.build]
cmd = "cmake --build build --config Release"
depends-on = ["configure"]

[tasks.test]
cmd = "ctest --test-dir build -C Release"
depends-on = ["build"]
```

## Lockfiles

Reproducibility

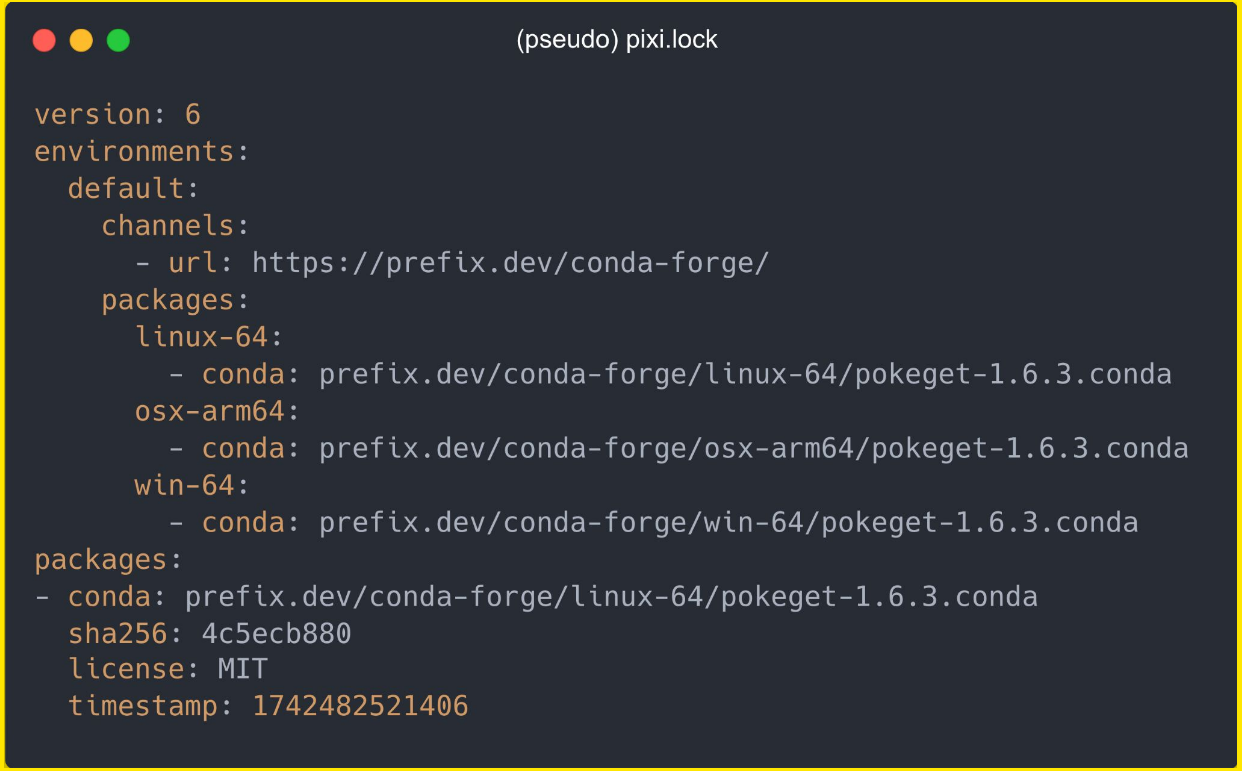
All platforms

Deterministic CI/CD

Auditable

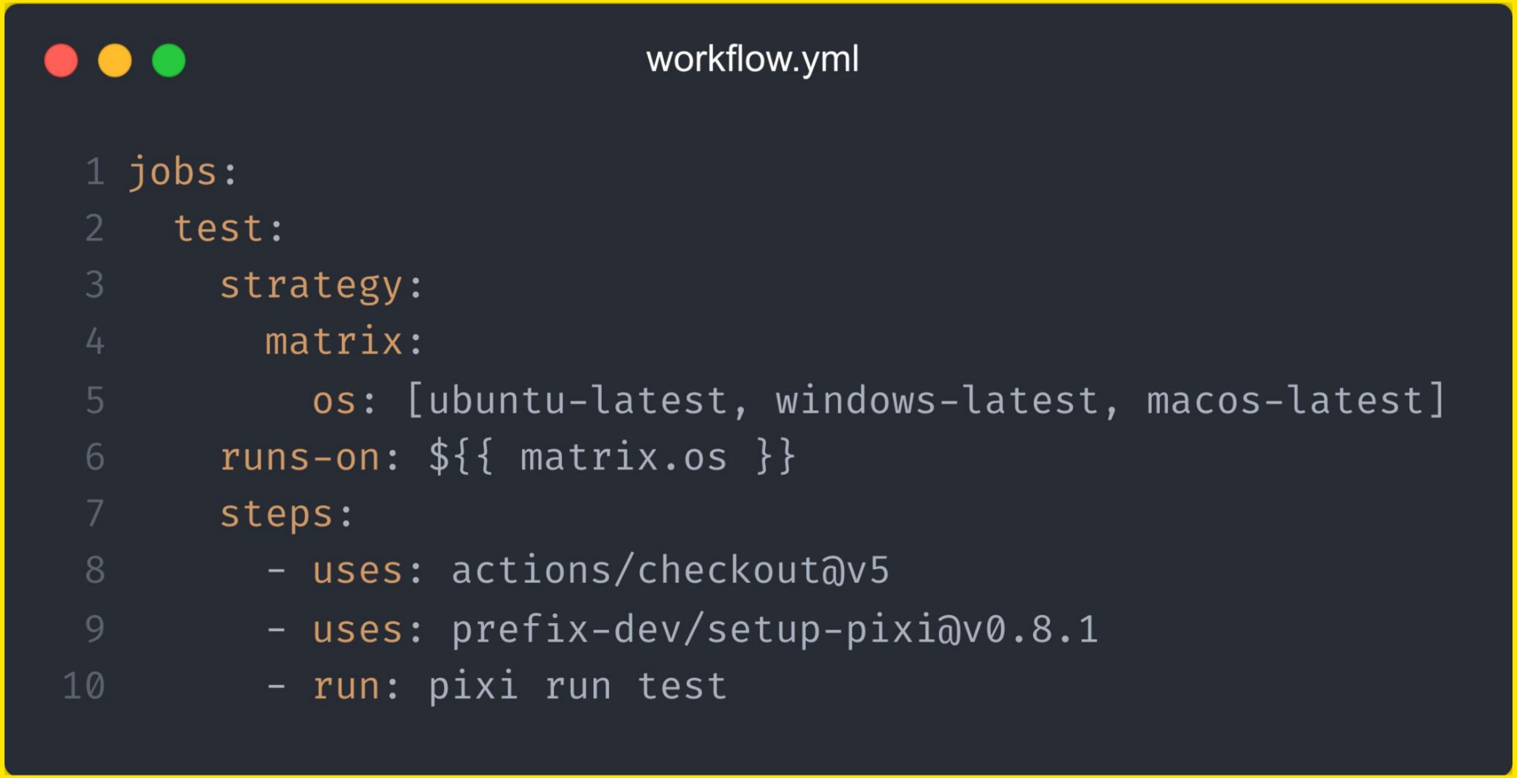
No solving

Rollbacks



```yaml
version: 6
environments:
  default:
    channels:
      - url: https://prefix.dev/conda-forge/
    packages:
      linux-64:
        - conda: prefix.dev/conda-forge/linux-64/pokeget-1.6.3.conda
      osx-arm64:
        - conda: prefix.dev/conda-forge/osx-arm64/pokeget-1.6.3.conda
      win-64:
        - conda: prefix.dev/conda-forge/win-64/pokeget-1.6.3.conda
packages:
- conda: prefix.dev/conda-forge/linux-64/pokeget-1.6.3.conda
  sha256: 4c5ecb880
  license: MIT
  timestamp: 1742482521406
```

# GitHub Actions



```yaml
jobs:
  test:
    strategy:
      matrix:
        os: [ubuntu-latest, windows-latest, macos-latest]
    runs-on: ${{ matrix.os }}
    steps:
      - uses: actions/checkout@v5
      - uses: prefix-dev/setup-pixi@v0.8.1
      - run: pixi run test
```

## Conda-forge

CONDA-FORGE

Default binary package source

Packaging multiple languages: C/C++, Rust, Python, Go, Fortran, etc.

Available packages:

| | |
|---|---|
| • Conda-forge: | **30K** Packages |
| • Conan: | **9K** Packages |
| • Vcpkg: | **3K** Packages |

https://vcpkg.io/en/packages
https://conan.io/center
https://conda-forge.org/

## CONDA

13+ years old

Binary Distribution of compiled libraries

## Source builds

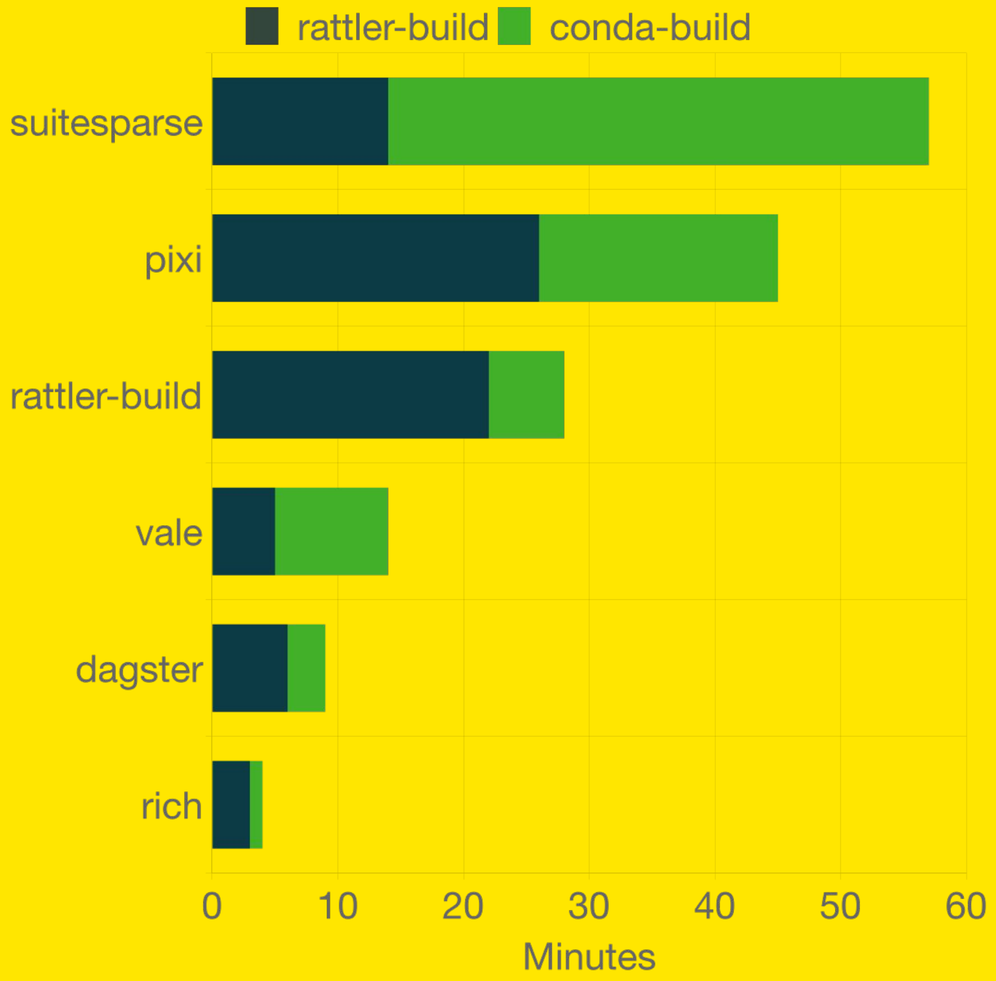
Demo

## Rattler-build

conda-build replacement

Pure yaml recipe

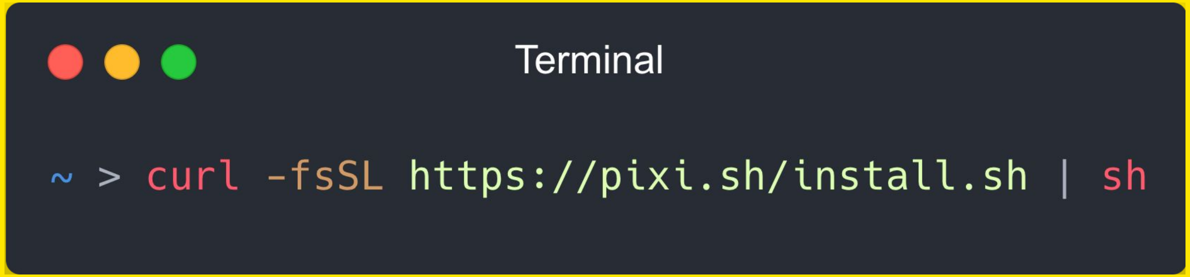
More info: https://rattler.build



## When to use Pixi?

- Reproducible developer workflows
- Not breaking other projects
- Easy CI setup
- Easy Distribution
- Cross-platform projects and applications

## Try it today!



Terminal

```
~ > curl -fsSL https://pixi.sh/install.sh | sh
```

https://pixi.sh

Prefix.dev

Thanks!