Flox 치트시트

Flox는 언어에 구애받지 않는 패키지 및 환경 관리자로, 다양한 플랫폼(macOS, Linux, x86, Arm)에서 재현 가능하고 일관된 개발 환경을 만드는 데 도움을 줍니다. 단일 매니페스트를 사용하여 패키지, 도구, 환경 변수, 서비스를 포함한 모든 프로젝트 종속성을 정의합니다.

핵심 개념

Flox의 핵심 개념은 **환경** 입니다. 환경은 프로젝트에 대한 패키지, 도구 및 특정 구성을 설치할 수 있는 격리된 공간입니다. 이 격리는 컨테이너가 아닌 미리 구성된 하위 셸을 통해 이루어지므로 flox가 기존 도구 및 셸과 원활하게 작동할 수 있습니다. 환경을 활성화하면 flox는 `PATH`를 수정하여 환경의 특정 패키지를 가리키도록 합니다.

Flox와 Nix Flakes

Nix Flake는 `flake.nix` 파일을 포함하는 소프트웨어 패키징용 독립형 장치입니다. 이 파일은 종속성과 소프트웨어를 빌드하거나 실행하는 방법을 정의합니다. Flake는 종속성을 `flake.lock` 파일의 특정 버전에 고정하여 재현성을 향상시킵니다. Flox는 Nix를 기반으로 구축되어 Nix 전문가가 아닐 수도 있는 개발자가 Nix의 강력한 기능에 더 쉽게 액세스할 수 있도록 합니다. Flox는 `manifest.toml` 파일에서 환경을 정의할 수 있으며 Nix Flake를 사용하여 소프트웨어 패키지를 정의하고 설치할 수 있습니다.

Flakes를 사용하여 Flox 설치

`manifest.toml` 파일을 편집하여 flox에 설치할 flake를 알려줍니다. `flox edit`를 실행하여 매니페스트를 편집할 수 있습니다.

```
[install]
cowsay = "github:flox-examples/cowsay"
```

그런 다음 환경을 활성화합니다.

```
flox activate
```

Flox가 `cowsay` flake를 다운로드하고 빌드하여 환경에 추가합니다.

Flox 프로필

Flox 프로필을 사용하면 `manifest.toml` 파일 내에서 셸별 별칭, 함수 및 스크립트를 정의할 수 있습니다. 이러한 사용자 정의는 Flox 환경을 활성화할 때 셸에서 소싱되므로 필요에 맞게 환경을 조정할 수 있습니다.

```
[profile]
alias l="ls -alh"
```

기본 워크플로우

1. 환경 초기화

프로젝트용 디렉토리를 만들고 해당 디렉토리로 이동한 다음 `flox init` 명령을 사용하여 새 flox 환경을 만듭니다.

```
mkdir my-project
cd my-project
flox init
```

2. 패키지 검색

`flox search` 명령을 사용하여 패키지를 검색합니다.

```
flox search cowsay
```

3. 패키지 설치

`flox install` 명령을 사용합니다.

```
flox install cowsay
```

4. 환경 활성화

`flox activate`로 환경을 활성화해야 합니다.

```
flox activate
```

5. 설치된 패키지 목록

`flox list`를 사용하여 현재 환경에 설치된 모든 패키지 목록을 볼 수 있습니다.

```
flox list
```

6. 환경 비활성화

환경을 종료하고 일반 셸로 돌아가려면 `exit`를 입력하기만 하면 됩니다.

```
exit
```

선언적 환경 관리

Flox를 사용하면 `.flox` 디렉토리에 있는 `manifest.toml` 파일을 사용하여 선언적으로 환경을 관리할 수 있습니다. `flox edit` 명령으로 이 파일을 열 수 있습니다.

환경 공유

FloxHub를 사용하여 다른 사람과 개발 환경을 쉽게 공유할 수 있습니다. `flox push`를 사용하여 환경을 FloxHub에 푸시하고 다른 사람들은 `flox pull`을 사용하여 정확히 동일한 설정을 복제할 수 있습니다.

추가 학습

- **공식 문서:** https://flox.dev/docs/
- **GitHub 저장소:** https://github.com/flox/flox