mise 치트시트

기본 개념

mise(이전 명칭 rtx)는 asdf-vm의 철학을 계승하여 개발된 도구로, 여러 프로그래밍 언어와 CLI 도구의 버전을 프로젝트별로 손쉽게 관리할 수 있게 해줍니다. .tool-versions 또는 mise.toml 파일을 사용하여 특정 프로젝트에 필요한 환경을 선언적으로 관리하는 것이 핵심입니다.

주요 명령어

- mise install (또는 mise i): 설정 파일에 명시된 모든 도구의 지정된 버전을 한꺼번에 설치합니다.
- mise use <tool>@<version>: 특정 도구의 버전을 전역 또는 로컬 프로젝트 설정에 추가하고 활성화합니다.
- mise current: 현재 디렉토리에서 활성화되어 사용 중인 모든 도구와 버전을 확인합니다.
- mise ls: 시스템에 설치된 모든 도구 및 버전 목록을 출력합니다.
- mise exec -- <command>: 설정된 특정 도구의 버전 환경에서 명령어를 실행합니다. (예: mise exec -- npm test)
- mise reshim: Shim(실행 대행 파일)을 재구성합니다. 플러그인을 새로 설치하거나 제거한 후에 필요할 수 있습니다.

플러그인 관리

- mise plugins ls: 현재 설치된 플러그인 목록을 확인합니다.
- mise plugins add <name>: 새로운 도구 플러그인을 추가합니다. (예: mise plugins add nodejs)
- mise plugins add <name> <git-url>: 특정 Git 리포지토리 주소를 통해 플러그인을 추가합니다.
- mise plugins rm <name>: 설치된 플러그인을 제거합니다.
- mise plugins update <name>: 특정 플러그인을 최신 상태로 업데이트합니다. (--all 옵션으로 전체 업데이트 가능)

버전 세부 제어

- mise ls-remote <tool>: 설치 가능한 전체 버전 목록을 온라인에서 조회합니다.
- mise install <tool>@<version>: 특정 버전의 도구를 직접 설치합니다.
- mise uninstall <tool>@<version>: 더 이상 필요 없는 특정 버전의 도구를 삭제합니다.
- mise global <tool>@<version>: 시스템 전체에서 기본으로 사용할 전역 버전을 설정합니다.
- mise local <tool>@<version>: 현재 디렉토리 전용 버전을 설정합니다. 실행 시 해당 디렉토리에 .tool-versions 파일이 생성되거나 수정됩니다.

설정 파일 예시 (.tool-versions)

프로젝트 루트 디렉토리에 이 파일을 두면, 해당 경로 진입 시 mise가 명시된 버전들을 자동으로 활성화합니다.

```
# .tool-versions 파일 예시
nodejs 20.11.0
python 3.11.7
rust 1.76.0
```

셸 통합 설정

mise가 정상적으로 동작하려면 사용하는 셸의 설정 파일(.bashrc, .zshrc 등)에 활성화 스크립트를 추가해야 합니다.

```
# .zshrc 설정 예시
eval "$(mise activate zsh)"
```

이 설정을 마치면 디렉토리를 이동할 때마다 mise가 설정을 자동으로 감지하여 해당 도구의 실행 경로를 PATH에 우선적으로 등록해 줍니다.

asdf-vm과의 차이점

mise는 asdf-vm과 호환되는 플러그인 생태계를 공유하며 사용법도 매우 유사합니다. 가장 큰 차별점은 Rust로 작성되어 asdf-vm보다 월등히 빠른 성능을 제공한다는 것이며, 환경 변수 관리 및 작업 실행(Task runner) 기능을 추가로 탑재하여 사용자 편의성을 높였습니다.