mise Cheatsheet

Basic Concepts

`mise` (formerly `rtx`) is a polyglot runtime manager, succeeding the philosophy of `asdf-vm`. It allows you to manage versions of different programming languages and CLI tools easily on a per-project basis using `.tool-versions` or `mise.toml` files.

Core Commands

- `mise install` (or `mise i`): Install all tool versions specified in the config file at once.
- `mise use <tool>@<version>`: Add and activate a specific tool version globally or in the local project config.
- `mise current`: Check all activated tools and versions in the current directory.
- `mise ls`: List all installed tools and their versions on the system.
- `mise exec -- <command>`: Run a command within the environment of specific tool versions (e.g., `mise exec -- npm test`).
- `mise reshim`: Reconstruct shims (proxy executables). Useful after installing/removing plugins.

Plugin Management

- `mise plugins ls`: List currently installed plugins.
- `mise plugins add <name>`: Add a new tool plugin (e.g., `mise plugins add nodejs`).
- `mise plugins add <name> <git-url>`: Add a plugin from a specific Git repository.
- `mise plugins rm <name>`: Remove an installed plugin.
- `mise plugins update <name>`: Update a specific plugin to the latest version (`--all` to update all).

Detailed Version Control

- `mise ls-remote <tool>`: View all installable versions online.
- `mise install <tool>@<version>`: Install a specific version of a tool.
- `mise uninstall <tool>@<version>`: Remove a specific version that is no longer needed.
- `mise global <tool>@<version>`: Set the default global version for the entire system.
- `mise local <tool>@<version>`: Set a version specific to the current directory (creates/modifies `.tool-versions`).

Config File Example (`.tool-versions`)

Place this in your project root, and `mise` will automatically activate these versions when you enter the directory.

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

Shell Integration

To make `mise` work, add the activation script to your shell config file (`.bashrc`, `.zshrc`, etc.).

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

Once configured, `mise` will automatically detect settings as you switch directories and prioritize the correct tool paths in your `PATH`.

Comparison with asdf-vm

`mise` shares the asdf-vm plugin ecosystem and has a very similar usage. The main differentiator is that `mise` is written in Rust, providing significantly faster performance, and includes additional features like environment variable management and a task runner.