NixOS Asia >

flake-parts

`flake-parts` brings the NixOS module system to flakes, thus providing a cleaner and simpler way to write otherwise complex flakes.

- Official site: https://flake.parts/
- Module documentation: https://community.flake.parts/

Links to this page

services-flake

> services-flake provides declarative, composable, and reproducible services for Nix development environment, and is based on flake-parts. Enabling users to have NixOS-like service on macOS and Linux.

process-compose-flake

> process-compose-flake is a flake-parts module for process-compose.

Replacing docker-compose with Nix for development

> It uses flake-parts for the module system (that's the simplicity aspect), and process-compose-flake for managing services, along with providing a TUI app to monitor them.

Nixifying a Haskell project using nixpkgs

> In the next tutorial part, we will modularize this `flake.nix` using flake-parts.

> [!note] `forAllSystems` The source code uses `forAllSystems`, which was not included in the tutorial above to maintain simplicity. Later, we will obviate `forAllSystems` and simplify the flake further using flake-parts.

Module System

> This module system is not natively supported in Flakes. However, flakes can define and use modules using flake-parts.

Modularize our flake using flake-parts

> flake-parts can be used as lightweight `forAllSystems` alternative

Introduction to module system

> We shall begin by understanding the low-levels: how to use `evalModules` from nixpkgs to define and use our own modules from scratch, using the aforementioned

> `lsd` use-case. The next tutorial in this series will go one high-level up and talk about how to work with modules across flakes, using flake-parts.

> You have just read a quick introduction to the module system, in particular how to define, use and share them in Flakes. To learn more about the module system, we recommend this video from Tweag as well the article "Module system deep dive" from nix.dev. Look out for the next tutorial in this series, where we will talk about flake-parts.

Auto formatting using `treefmt-nix`

> The `flake-root` flake-parts module is needed to find the root of your project based on the presence of a file, by default it is `flake.nix`.