Usage of the New CLI

Once you have enabled the `nix-command` and `flakes` features, you can start using the new generation Nix command-line tools provided by New Nix Commands. In this section, we will focus on two commands: `nix shell` and `nix run`. Other important commands like `nix build` will be discussed in detail in `nix develop` & `pkgs.mkShell`

`nix shell`

The `nix shell` command allows you to enter an environment with the specified Nix package and opens an interactive shell within that environment:

```
# hello is not available
› hello
hello: command not found

# Enter an environment with the 'hello' and `cowsay` package
› nix shell nixpkgs#hello nixpkgs#cowsay

# hello is now available
› hello
Hello, world!

# ponysay is also available
› cowsay "Hello, world!"
 _______
< hello >
 -------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

`nix run`

On the other hand, `nix run` creates an environment with the specified Nix package and directly runs that package within the environment (without installing it into the system environment):

```shell
# hello is not available
› hello
hello: command not found

# Create an environment with the 'hello' package and run it
› nix run nixpkgs#hello
Hello, world!
```

Since `nix run` directly executes the Nix package, the package specified as the argument must generate an executable program.

According to the `nix run --help` documentation, `nix run` executes the command `<out>/bin/<name>`, where `<out>` is the root directory of the derivation and `<name>` is selected in the following order:

- The `meta.mainProgram` attribute of the derivation
- The `pname` attribute of the derivation
- The content of the `name` attribute of the derivation with the version number removed

For example, in the case of the 'hello' package we tested earlier, `nix run` actually executes the program `$out/bin/hello`.

Here are two more examples with detailed explanations of the relevant parameters:

```bash
# Explanation of the command:
#   `nixpkgs#ponysay` means the 'ponysay' package in the 'nixpkgs' flake.
#   `nixpkgs` is a flake registry id, and Nix will find the corresponding GitHub
#   from <https://github.com/NixOS/flake-registry/blob/master/flake-registry.jso
# Therefore, this command creates a new environment, installs, and runs the 'pon
#   Note: It has been mentioned earlier that a Nix package is one of the outputs
echo "Hello Nix" | nix run "nixpkgs#ponysay"

# This command has the same effect as the previous one, but it uses the complete
echo "Hello Nix" | nix run "github:NixOS/nixpkgs/nixos-unstable#ponysay"
```

Common Use Cases for `nix run` and `nix shell`

These commands are commonly used for running programs temporarily. For example, if I want to clone my configuration repository using Git on a new NixOS host without Git installed, I can use the following command:

```bash
nix run nixpkgs#git clone git@github.com:ryan4yin/nix-config.git
```

Alternatively, I can use `nix shell` to enter an environment with Git and then run the `git clone` command:

```bash
nix shell nixpkgs#git
git clone git@github.com:ryan4yin/nix-config.git
```

0 reactions

0 comments