Nitro Setup: My Setup Settings v2.0 (Nix/NixOS edition)

Background

If this is the first time you see me, let me tell you to check out my GitHub. You will see some uncompleted projects I left because of problems with some dependencies, lack of time or just because I had no more interest on them.

The only project that is "well mantained" is My Setup Settings which is a script to setup my Kali Linux environment, which I really love. But the dependencies are all a problem. Sometimes an update can break all my preconfigured setup with no reason. The changes doesn't remain on the time and to change my setup settings it's all a topic. Moreover, I personally need the latest and greatest packages, which I cannot obtain without moving to unstable distros and always risking all my job.

This days, a "new" distro has appeared on the stage: NixOS. NixOS is a complete different Linux distro. Built on the top of the Nix package manager, this system isolates dependencies and uses a declarative and functional approach to make possible one thing: To make your config to work *everywhere*.

In order to learn this new distribution, I decided to move my hole setup to NixOS and I've found it a **really** difficult job. First, you have to be familiar with the Nix programming language, a strange mix between JSON, Bash Script and Haskell. The second thing that I have found quite difficult is the usage of flakes, modules and the hole system configuration.

Now that I belive that I've a better knowlage about what's going on, it's time to make things easier for other people too.

The Nix programming language

Let's begin with the Nix language. As you may guessed, there are datatypes, like in any other language. These are:

- Strings, like `"hello"` or `'I am a string too!'`. You can also use multi-line strings like this:

```
''
  I am
  a very
  long
  string!
''
```

- Numbers, like `1`, `2.0` and `-1`. You can also perform some math with them, like:

```
1 + 2 # 3
2 - 2 # 0
2 / 2 # 1
3 * 3 # 9
```

- Booleans, with the well-known `true` and `false`.
- Arrays, like `[ "Tokyo-3" "Athenas" "Hyrule" "Neverland" "Amestris" ]` or:

```
[
  1
  2
  3
  4
]
```

- And sets, like:

  ```
  {
    name: "Julius Caesar";
    age: 47;
    skils: [
      "Swordsmanship",
      "Appraisal",
      "Fusion"
    ];
  }
  ```

You can also reference a field inside a set like this:

```
# This is a comment line
# Let's say there's a set called plugin and has two fields:
# url, which is a string; and name; which is also a string.
plugin.name # Refers to the plugin name.
plugin.url # Refers to the plugin URL.
```

- There are also functions, which are defined like this:

```
param: result
```

Notice that a function can only accept *one* parameter and return just *one* output. This might not sound very useful, but let's make a little trick:

```
{ field1, field2 }: field1 + field2
```

We used a set to hold all the parameters of our function and returned their addition. The input and the output of a function can be as complex as the Nix data types allows you to define them.

Now that we've covered the most important stuffs from the Nix language, let's dive into configuring our system.