Setting up a Python development environment

Contents

In this example you will build a Python web application using the Flask web framework as an exercise. To make best use of it you should be familiar with defining declarative shell environments.

Create a new file called `myapp.py` and add the following code:

```
#!/usr/bin/env python

from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello():
    return {
        "message": "Hello, Nix!"
    }

def run():
    app.run(host="0.0.0.0", port=5000)

if __name__ == "__main__":
    run()
```

This is a simple Flask application which serves a JSON document with the message `"Hello, Nix!"`.

Create a new file `shell.nix` to declare the development environment:

```
{ pkgs ? import (fetchTarball "https://github.com/NixOS/nixpkgs/tarball/nixos-23.11

pkgs.mkShellNoCC {
  packages = with pkgs; [
```

Skip to main content

```
    jq
  ];
}
```

This describes a shell environment with an instance of `python3` that includes the `flask` package using `python3.withPackages`. It also contains `curl`, a utility to perform web requests, and `jq`, a tool to parse and format JSON documents.

Both of them are not Python packages. If you went with Python's virtualenv, it would not be possible to add these utilities to the development environment without additional manual steps.

Run `nix-shell` to enter the environment you just declared:

```
$ nix-shell
these 2 derivations will be built:
  /nix/store/5yvz7zf8yzck6r9z4f1br9sh71vqkimk-builder.pl.drv
  /nix/store/aihgjkf856dbpjjqalgrdmxyyd8a5j2m-python3-3.9.13-env.drv
these 93 paths will be fetched (109.50 MiB download, 468.52 MiB unpacked):
  /nix/store/0xxjx37fcy2nl3yz6igmv4mag2a7giq6-glibc-2.33-123
  /nix/store/138azk9hs5a2yp3zzx6iy1vdwi9q26wv-hook
...

[nix-shell:~]$
```

Start the web application within this shell environment:

```
[nix-shell:~]$ python ./myapp.py
 * Serving Flask app 'myapp'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Us
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://192.168.1.100:5000
Press CTRL+C to quit
```

You now have a running Python web application. Try it out!

Open a new terminal to start another session of the shell environment and follow the commands below:

```
$ nix-shell

[nix-shell:~]$ curl 127.0.0.1:5000
{"message":"Hello, Nix!"}
```

Skip to main content

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    26  100    26    0     0  13785      0 --:--:-- --:--:-- --:--:-- 26000
"Hello, Nix!"
```

As demonstrated, you can use both `curl` and `jq` to test the running web application without any manual installation.

You can commit the files we created to version control and share them with other people. Others can now use the same shell environment as long as they have Nix installed.

Next steps

- Packaging existing software with Nix
- Working with local files
- Automatic environment activation with direnv
- Automatically managing remote sources with npins