Nix ❤️ Python:

Maybe you don't need devcontainers after all

A lot of questions

1. What is Nix?
2. What is devenv?
3. Why is this relevant to me as a Python engineer?
4. Do I still need Docker?

What is Nix?

What is Nix?

Nix - the build system and package manager

- Declarative
  - Declare inputs and desired output state, Nix handles the rest
- Reproducible
  - build(inputs) → output
  - No accidentally installed system libs, user tools
  - Multiple version of same lib coexist in peace

What is Nix?

Nix - the purely function language

- Package definitions
- Configuration management

```
system.primaryUser = "tomislavmaricevic";

system.defaults = {
  dock.autohide = true;
  dock.show-recents = false;
  finder.AppleShowAllExtensions = true;
  finder.AppleShowAllFiles = true;
  screencapture.location = "~/Pictures/screenshots";
  screensaver.askForPasswordDelay = 10;
  controlcenter.Bluetooth = true;

system.keyboard = {
  enableKeyMapping = true;
  remapCapsLockToEscape = true;
};

homebrew = {
  enable = true;
  brews = [
    "ollama"
    "borders"
  ];
  taps = [
    "FelixKratz/formulae"
  ];
  casks = [
    "google-chrome"
    "raycast"
    "chatgpt"
    "whatsapp"
    "dropbox"
    "orcaslicer"
    "stremio"
    "claude"
    "todoist-app"
    "anki"
  ];
};

services.aerospace = {
  enable = true;
  settings = builtins.fromTOML (builtins.readFile ../config/aerospace/aerospace.toml);
};
```

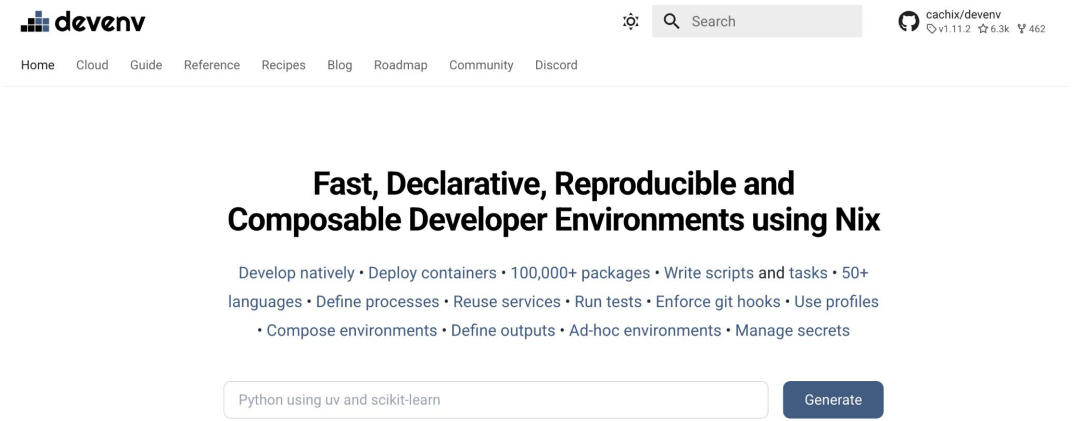
What is Nix?

NixOS - Linux distribution based on Nix

- Uses Nix lang for configuration and Nix package manager for packages

Switching atomically between system states as immutable artifacts >> mutating a live system

What is devenv?



Nix-based development environment manager

devenv = nix shell + nice abstractions + service management + direnv

Why is this relevant to me as a Python engineer?

Scenario:

- Monorepo with Django backend and React frontend
- Backend:
  - uv for package management
  - granian web server
  - rabbitmq message broker
  - Celery workers
  - ruff formatting
- Frontend:
  - Node v24
  - pnpm package management
  - Biome formatting

1. devenv.nix

```nix
{
  pkgs,
  lib,
  config,
  inputs,
  ...
}:

{
  # Helper packages, and optional lib dependencies
  packages = [
    pkgs.jujutsu
    pkgs.jq
    pkgs.zlib
  ];

  # Language runtimes and package managers
  languages = {
    python = {
      enable = true;
      version = "3.12";
      uv.enable = true;
    };
    javascript = {
      enable = true;
      package = pkgs.nodejs_24;
      pnpm.enable = true;
    };
  };

  # Predefined processes
  services = {
    postgres = {
      enable = true;
      package = pkgs.postgresql_18;
      initialDatabases = [
        {
          name = "app_db";
          user = "app_user";
          pass = "app_pass";
        }
      ];
      initialScript = ''
        CREATE USER app_user WITH PASSWORD 'app_pass';
      '';
    };
    redis.enable = true;
    rabbitmq = {
      enable = true;
      managementPlugin.enable = true;
    };
  };

  # Ad-hoc processes
  processes = {
    api = {
      exec = "uv run granian --interface rsgi main:app";
      cwd = "${config.git.root}/backend";
    };
    worker = {
      exec = "uv run celery -A tasks worker --loglevel=info";
      cwd = "${config.git.root}/backend";
    };
    frontend = {
      exec = "pnpm dev";
      cwd = "${config.git.root}/frontend";
    };
  };

  git-hooks.hooks = {
    ruff.enable = true;
    biome.enable = true;
  };
}
```

```nix
  git-hooks.hooks = {
    ruff.enable = true;
    biome.enable = true;
  };

  claude.code = {
    enable = true;
    agents = {
      code-reviewer = {
        description = "Expert code review specialist that checks for quality, security, and best practices";
        proactive = true; # Claude will use this automatically when appropriate
        tools = [
          "Read"
          "Grep"
          "TodoWrite"
        ];
        prompt = ''
          You are an expert code reviewer. When reviewing code, check for:
          - Code readability and maintainability
          - Proper error handling
          - Security vulnerabilities
          - Performance issues
          - Adherence to project conventions

          Provide constructive feedback with specific suggestions for improvement.
        '';
      };

      test-writer = {
        description = "Specialized in writing comprehensive test suites";
        proactive = false; # Only invoked explicitly
        tools = [
          "Read"
          "Write"
          "Edit"
          "Bash"
        ];
        prompt = ''
          You are a test writing specialist. Create comprehensive test suites that:
          - Cover edge cases and error conditions
          - Follow the project's testing conventions
          - Include unit, integration, and property-based tests where appropriate
          - Have clear test names that describe what is being tested
        '';
      };
    };
    hooks = {
      # Protect sensitive files (PreToolUse hook)
      protect-secrets = {
        enable = true;
        name = "Protect sensitive files";
        hookType = "PreToolUse";
        matcher = "^(Edit|MultiEdit|Write)$";
        command = ''
          # Read the JSON input from stdin
          json=$(cat)
          file_path=$(echo "$json" | jq -r '.file_path // empty')

          if [[ "$file_path" =~ \.(env|secret)$ ]]; then
            echo "Error: Cannot edit sensitive files"
            exit 1
          fi
        '';
      };
    };
  };
}
```

Do I still need Docker?

Yes.

Do I still need Docker?

Yes, when:

- You need complex networking (e.g. many local services) – service discovery, compose network
- Testing container-related functionality
- High production parity is necessary
- Toolchain native performance is not crucial
- You like to develop in the cloud (e.g. Github Codespaces)

Docker Was Too Slow, So We Replaced It: Nix in Production @ Anthropic

https://www.youtube.com/watch?v=iPoL03tFBtU

Thank you!

Q & A!

Open to new opportunities, reach out!

Tomislav Maričević | https://tmarice.dev | @tmrcv | github.com/tmarice