Why I ❤️ DevContainers

Problems

> These problems are especially bad for **hobby** projects!

1. Development machines tend to collect too many tools over time.
2. Project dependencies can conflict with each other.
3. Opening *old* projects tends to go badly.
4. Dealing with inconsistencies in dependencies/tooling on multi-dev projects.

What is a *Container*?

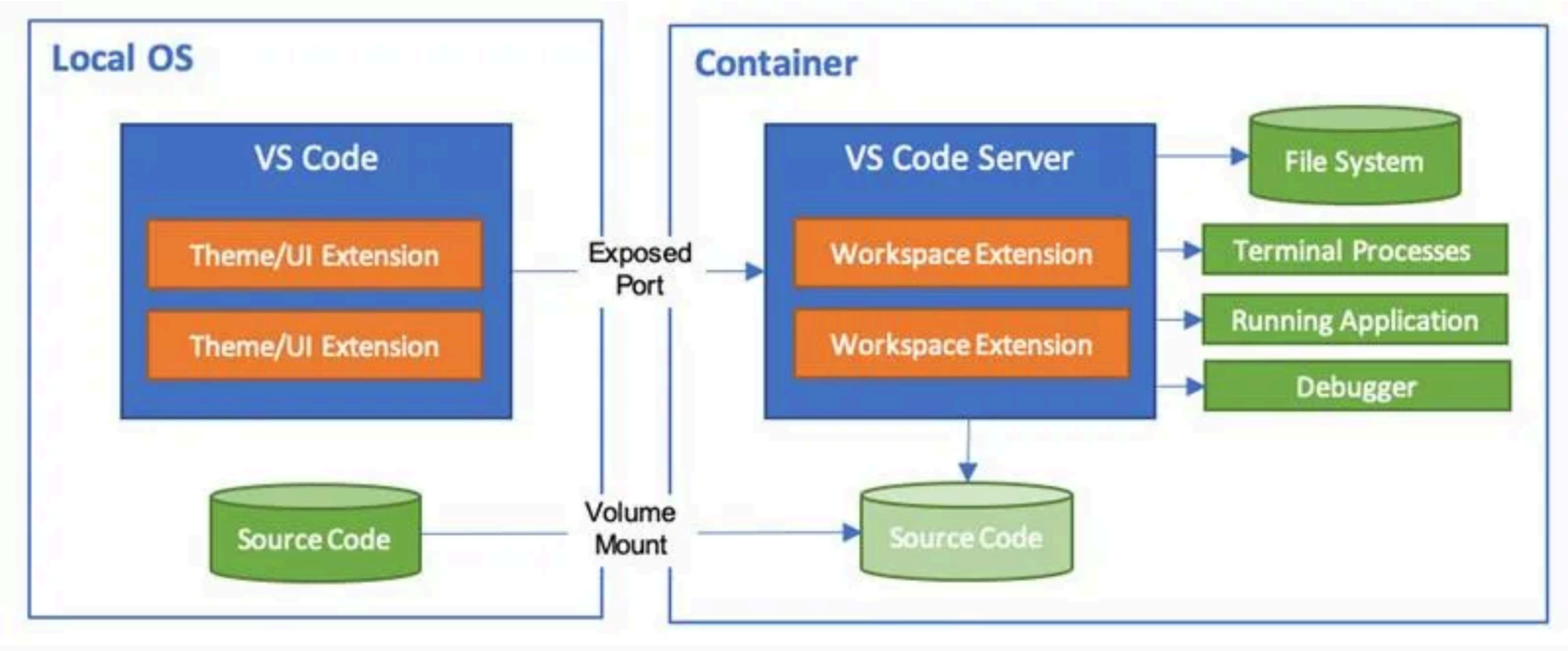
> A container (Docker) is a lightweight, standalone executable package that **includes everything needed to run a piece of software**: code, runtime, system tools, libraries, and settings. It allows you to install and run software in an **isolated environment**, ensuring consistent performance across different environments without the overhead of a full virtual machine.

Docker is a very common container environment but there are others such as LXC, Kubernetes, Podman, ...

What is a *DevContainer*?

> A devcontainer is a pre-configured development environment that uses container technology to provide a **consistent, isolated, and reproducible workspace for developers**. It includes all the necessary tools, dependencies, and settings needed to work on a project, ensuring that the development setup remains the same across different systems.

How VSCode handles DevContainers...



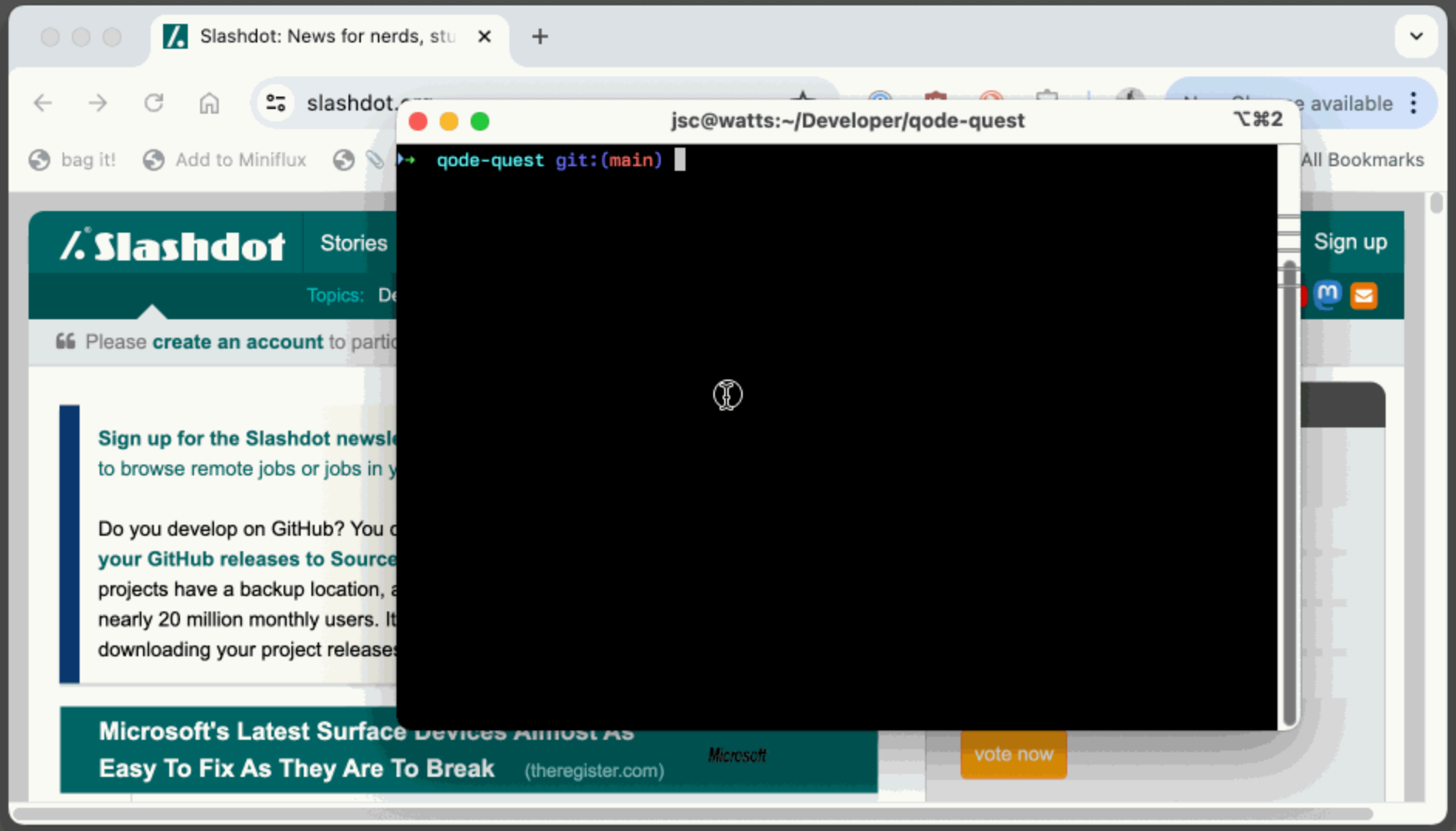
- Source code exists on your local machine
- All the code runs in the container
- Container specific Extensions!

Why I ❤️ DevContainers

- I dabble in many fun side projects, in many languages, with many toolkits
- My computers become a 🗑️🔥 fire of development tools over time
- Opening old projects is often frustrating (dependencies no longer there)

DevContainers make my life **much** better:

1. I can define (per project) exactly the environment that project needs (i.e. Elixir 1.17 + Erlang/OTP 27 + Postgres).
2. I can work on that project in a *container* keeping those dependencies off my bare machine.
3. I can open that project a year later and it'll still work, despite whatever upgrades / changes have happened on my machine


Slashdot: News for nerds, stu
slashdot.
bag it!
Add to Miniflux
All Bookmarks
Slashdot
Stories
Topics:
Please create an account to parti
Sign up for the Slashdot newsl
to browse remote jobs or jobs in y
Do you develop on GitHub? You c
your GitHub releases to Source
projects have a backup location, a
nearly 20 million monthly users. It
downloading your project releases
Microsoft's Latest Surface
Easy To Fix As They Are To Break
(theregister.com)
Microsoft
vote now
jsc@watts:~/Developer/qode-quest
⌥⌘2
qode-quest git:(main)

What's required to use a DevContainer?

1. Docker (*can technically be on a remote host*)
2. Editor with support for DevContainers:
    i. Visual Studio Code
    ii. NeoVim - I've never tried this
    iii. Github Codespaces - Work right from your browser (zero local footprint) - $$
    iv. Jetbrains.*
3. `.devcontainer` configuration in the project root

A basic `.devcontainer/devcontainer.json`

```
{
  "name": "Python 3",
  "image": "mcr.microsoft.com/devcontainers/python:1-3.12",

  // Features to add to the dev container. More info: https://containers.dev/features.
  "features": {
    "ghcr.io/devcontainers-contrib/features/postgres-asdf:1": "1.0.2",
  },

  // Install some Configure tool-specific properties.
  "customizations": {
    "vscode": {
      "settings": {},
      "extensions": [
        "streetsidesoftware.code-spell-checker",
        "ms-python.python"
      ]
    }
  },

  // Use 'forwardPorts' to make a list of ports inside the container available locally.
  "forwardPorts": [9000],

  // Use 'postCreateCommand' to run commands after the container is created.
  "postCreateCommand": "pip3 install -r requirements.txt"

}
```

But what if I need something special?

Customizing a container with `Docker` to get what we want

What is a `Dockerfile`?

> A Dockerfile is a text document that contains a series of instructions on how to build a Docker image. Each instruction in the Dockerfile creates a layer in the image, starting from a base image, and adds software, copies files, sets environment variables, and defines commands to run. The Dockerfile is used by the Docker engine to automate the process of creating a containerized environment, ensuring consistency and reproducibility across different systems.

Updated `devcontainer.json` (using a Dockerfile)

```
{
  "name": "My DevContainer",
  "build": {"dockerfile": "Dockerfile"},

  // Command to run after dev container *created*.
  "postCreateCommand": ". .devcontainer/post_create.sh",

  "forwardPorts": [4000, 5432],

  "customizations": {
    "vscode": {
      "extensions": [
        "phoenixframework.phoenix",
        "JakeBecker.elixir-ls",
      ]
    }
  }
}
```

The special `.devcontainer/Dockerfile`

```
FROM mcr.microsoft.com/devcontainers/base:jammy

# Get this thing up-to-date
RUN apt-get update
RUN apt-get upgrade -y

# Install some core tools
RUN apt-get install -y git curl inotify-tools wget imagemagick

# Erlang Deps
RUN apt-get install -y build-essential autoconf m4 libncurses5-dev
RUN apt-get install -y libwxgtk3.0-gtk3-dev libwxgtk-webview3.0-gtk3-dev
RUN apt-get install -y libgl1-mesa-dev libglu1-mesa-dev libpng-dev libssh-dev
RUN apt-get install -y unixodbc-dev xsltproc fop libxml2-utils libncurses-dev openjdk-11-jdk

# Install ASDF
RUN git clone https://github.com/asdf-vm/asdf.git /opt/asdf --branch v0.13.1

# install Elixir and Erlang
ARG ELIXIR_VERSION=1.17.1
ARG ERLANG_VERSION=27.0
RUN sh -c 'echo "source /opt/asdf/asdf.sh" >> /home/vscode/.bashrc'
RUN sudo -u vscode bash -c 'source /opt/asdf/asdf.sh && asdf plugin-add erlang https://github.com/asdf-vm/asdf-erlang.git'
RUN sudo -u vscode bash -c "source /opt/asdf/asdf.sh && asdf install erlang $ERLANG_VERSION && asdf global erlang $ERLANG_VERSION"
RUN sudo -u vscode bash -c 'source /opt/asdf/asdf.sh && asdf plugin-add elixir https://github.com/asdf-vm/asdf-elixir.git'
RUN sudo -u vscode bash -c "source /opt/asdf/asdf.sh && asdf install elixir $ELIXIR_VERSION && asdf global elixir $ELIXIR_VERSION"

# locale
ENV LANG en_US.UTF-8
ENV LC_ALL en_US.UTF-8
ENV LANGUAGE en_US:en

CMD ["/bin/bash"]
```

Doing some setup steps on container creation

```
{
  "name": "My DevContainer",
  ...

  // Command to run after dev container *created*.
  "postCreateCommand": ". .devcontainer/post_create.sh",

  ...
}
```

My `.devcontainer/post_create.sh` script:

```
cd assets && yarn install && cd ..
mix setup
```

But what if I need other services?

<cough>A database</cough>

Adding more services

1. Use `docker-compose`

   Not really supported by Github Codespaces.

2. Add services to your `Dockerfile` 👈👈

   Slightly annoying to start when reattaching to the container.

3. Host those services outside of your container

   Defeats the self-containedness benefits of the container.

Adding postgres to my `Dockerfile`

```
...

ENV PGUSER=postgres
ENV PGHOST=127.0.0.1
RUN sudo sh -c 'echo "deb https://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" \
  > /etc/apt/sources.list.d/pgdg.list'
RUN curl -fsSL https://www.postgresql.org/media/keys/ACCC4CF8.asc \
  |sudo gpg --dearmor -o /etc/apt/trusted.gpg.d/postgresql.gpg
RUN apt update
RUN apt install -y postgresql-16
# Postgres from anywhere!
RUN sh -c 'echo "host all all all trust" > /etc/postgresql/16/main/pg_hba.conf'

...
```

Docker containers don't really support starting additional services on "boot", so we need to start it manually each time our container starts.

```
$ sudo service postgresql start
```

TL;DR; Why I ❤️ DevContainers

- I can experiment with new tools without trashing my development machine.
- I don't end up with dependency conflicts between different projects.
- I spend less time trying to fix old projects that no longer run on my computer.
- I *somewhat* reduce the risk of a malicious development tool (or code base) doing bad things to my entire computer

Resources

- My Full Elixir Sample Project
  https://github.com/jclement/elixir_devcontainer_example
- VSCode DevContainer Docs
  https://code.visualstudio.com/docs/devcontainers/containers
- Available DevContainer Features (addons)
  https://containers.dev/features