Devcontainers, the Why, What and How

Down the rabbit hole to stay

What's the problem?

- Machine setup...
  - Install this, that, ow and this, and hey don't forget that
  - Version hell and clashes with already installed tools
- "It works on my machine..."
- Linter/formatter rules not correctly applied
  - Frustrated developers
  - Frustrated maintainers
  - ...

Can we solve it?

- Stop complaining andd just install the lot
- Remote development (e.g. over ssh)
- Combine all tools, libraries and the lot in one package
  - VM
  - Snap/Flatpack/AppImage
  - (Docker) container
  - Devcontainer with IDE supporting them

So… devcontainers… what are they?

containers.dev: *A development container (or dev container for short) allows you to use a* ***container*** *as a full-featured development environment. It can be used to run an application, to separate tools, libraries, or runtimes needed for working with a codebase, and to aid in continuous integration and testing. …*

Down the rabbit hole: containers

*... you to use a **container** as a full-featured ...*

- Sandbox environment
- Like a VM: full OS
- Lightweight: uses kernel of the host
- Declarative: Infrastructure as Code
- Integration with lots of tools

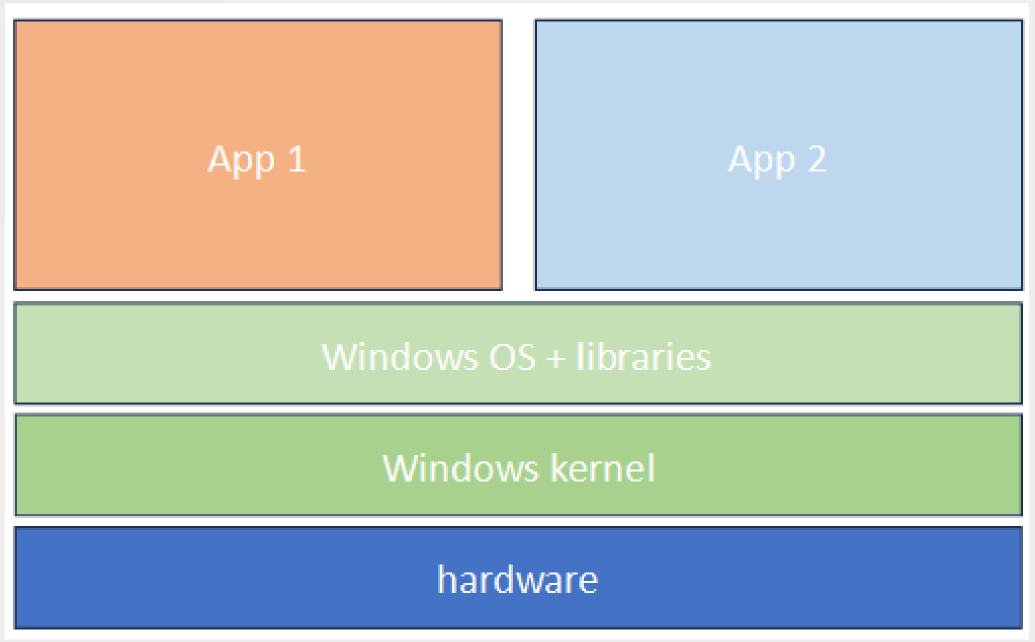
Down the rabbit hole: containers

They are used for:

- Easy deployment of (web) applications
- Micro-services on a (kubernetes) cluster
- Packaging of software and dependencies
  - running software locally
  - packaged software for use in CI/CD

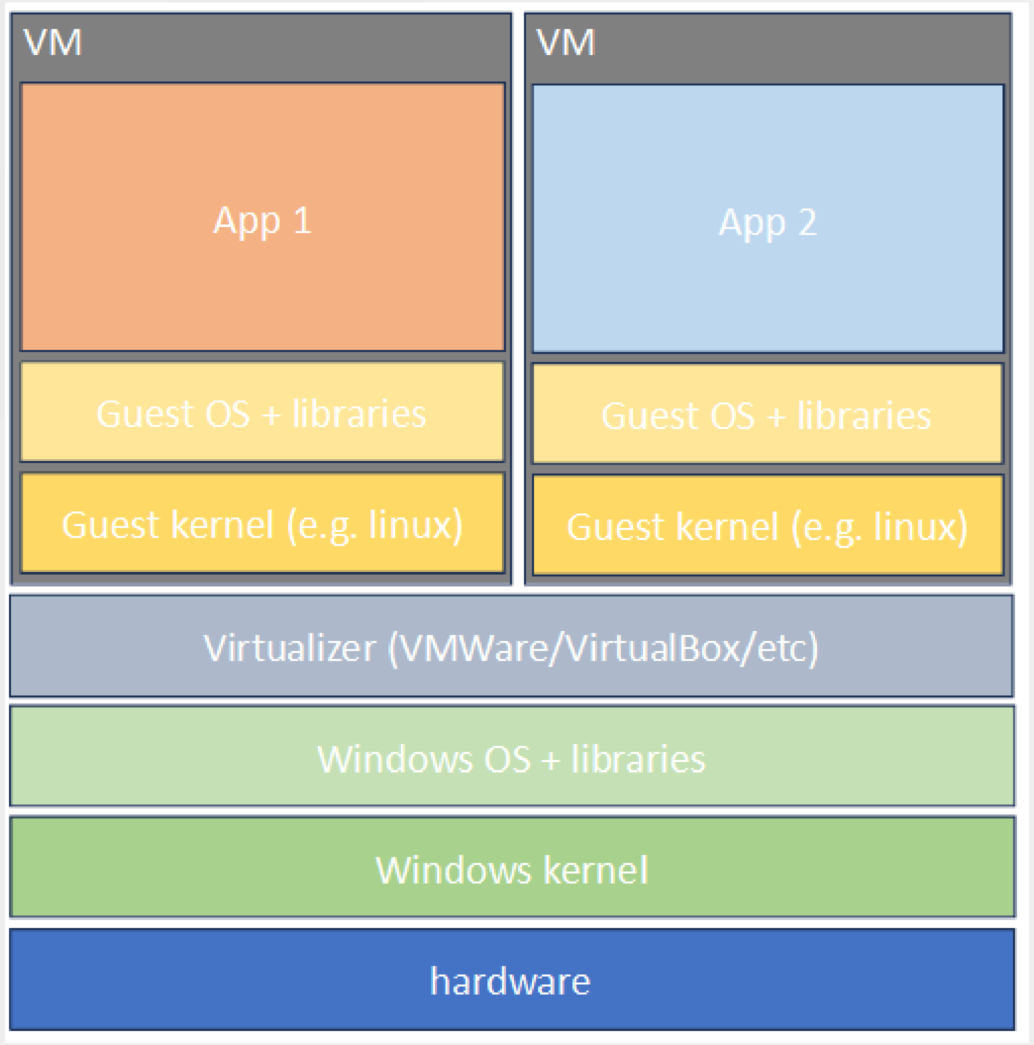
Like a VM and lightweight

Applications on top of OS



Like a VM and lightweight

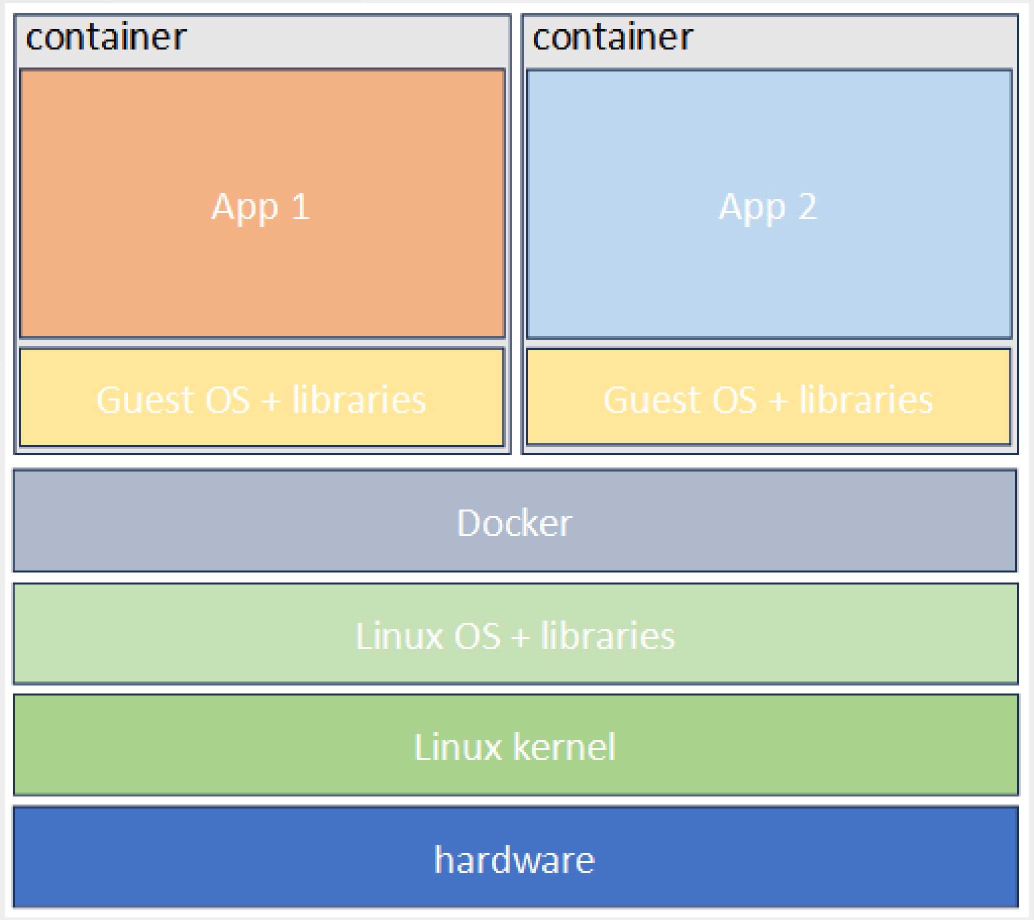
Applications in VM's



Like a VM and lightweight

Applications in Docker containers

Alpine: … *A container requires no more than* ***8 MB*** *…*



Declarative: Infrastructure as Code

```dockerfile
FROM node:10-alpine

RUN mkdir -p /app && chown -R node:node /app
WORKDIR /app
USER node

CMD /bin/sh
```

- To create a container image:

  `docker build --tag 'node_example' .`

Declarative: Infrastructure as Code

- To run a container image:
  `docker run -it 'node_example'`
- This will get you a Debian shell

```
$ docker run -it node_example
/app $ pwd
app
/app $ whoami
node
/app $
```

Containers and Docker

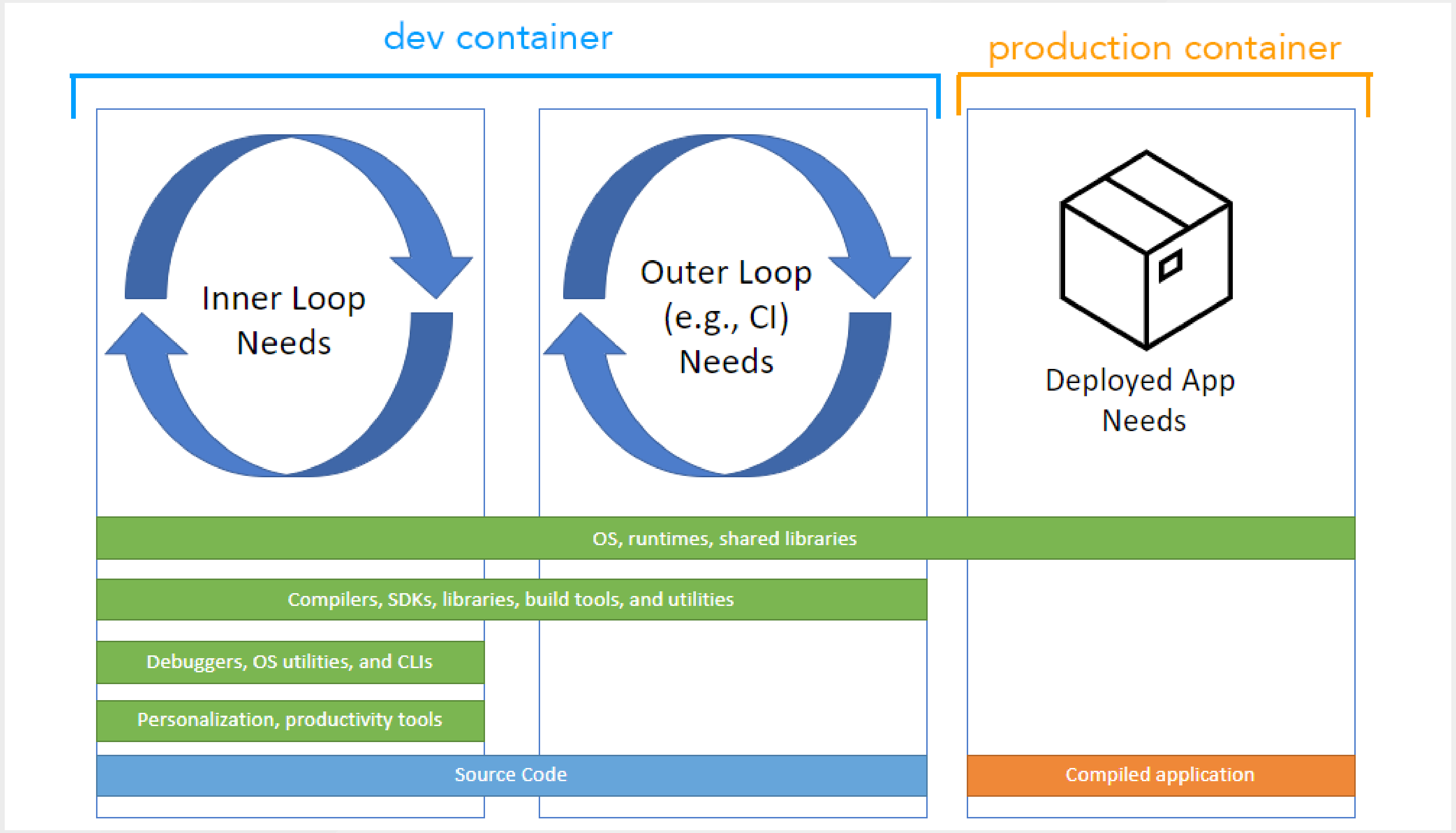
Example: Node app on Docker

- Note: **not** the app as Docker container, but development in Docker container
- `docker run -it -p 80:8080 -v .:/app node_example`
    - `-it`: interactive -> stay in terminal
    - `-p 80:8080`: map port 8080 from the container to port 80 on the host
    - `-v .:/app`: mount the current directory (`.`) to `/app` in the container

Example: Node app on Docker

- + Working on local machine
- + Whatever IDE you want
- - Building and running from terminal
- - No IDE-integrated debugging

Devcontainers: containers with spice


dev container
production container
Inner Loop Needs
Outer Loop (e.g., CI) Needs
Deployed App Needs
OS, runtimes, shared libraries
Compilers, SDKs, libraries, build tools, and utilities
Debuggers, OS utilities, and CLIs
Personalization, productivity tools
Source Code
Compiled application

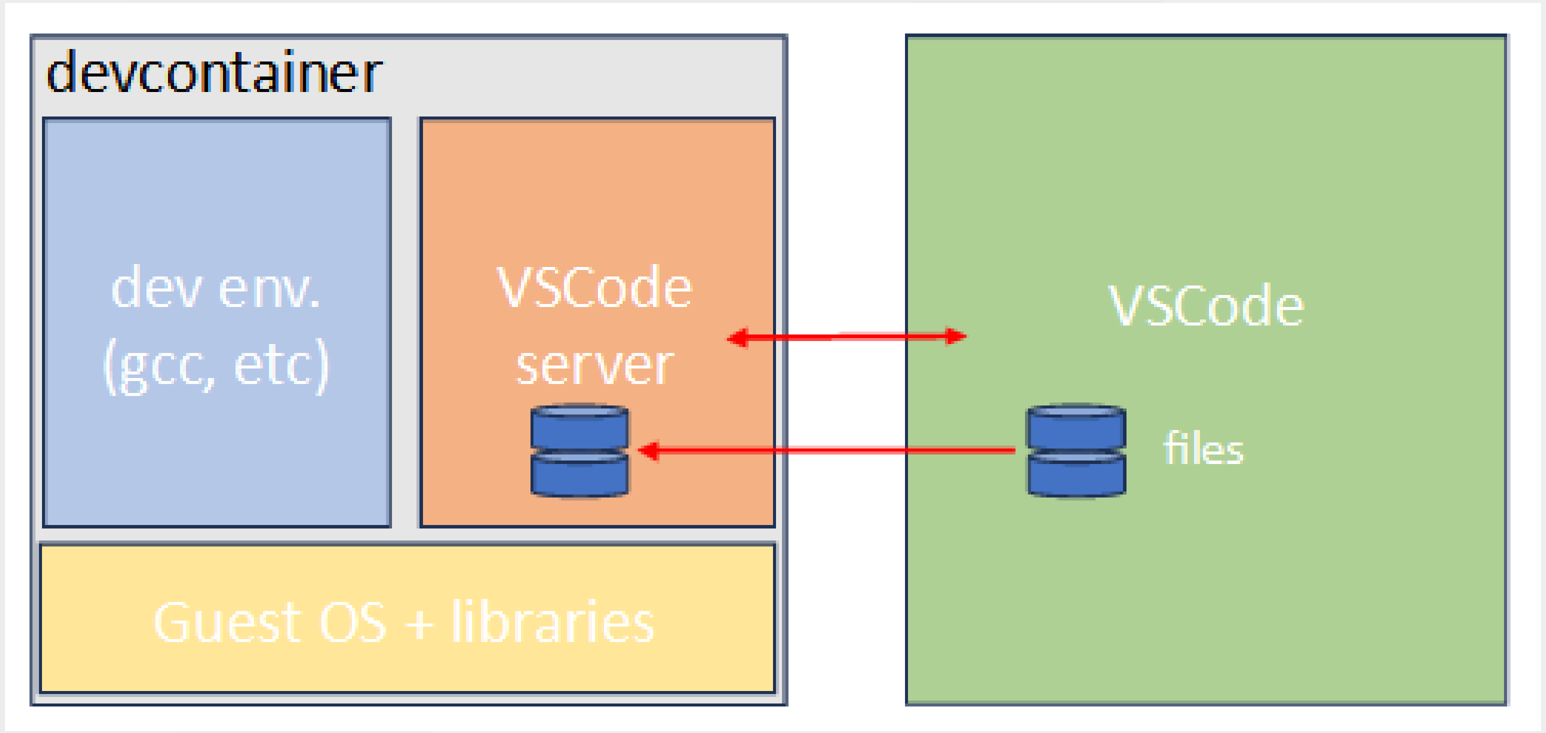
Devcontainers: containers with spice

- Full development environment
- Utilities and personalization
- Configuration of IDE
  - git with credentials
  - debugger

You should like VSCode...

- Supported tools
  - VSCode
  - Visual Studio >2022 for C++
  - IntelliJ IDEA (early support)
  - CLI, Cachix devenv, Jetpack.io Devbox
  - Github Codespaces, CodeSandbox, DevPod, Schema

Devcontainers and VSCode



Devcontainers: an example

```
{
    "build": { "dockerfile" : "Dockerfile" },
    "customizations": {
        "vscode": {
            "extensions" : [
                "ms-vscode.js-debug",
                "leizongmin.node-module-intellisense"
            ]
        }
    },
    "forwardPorts": [ 8080 ]
}
```

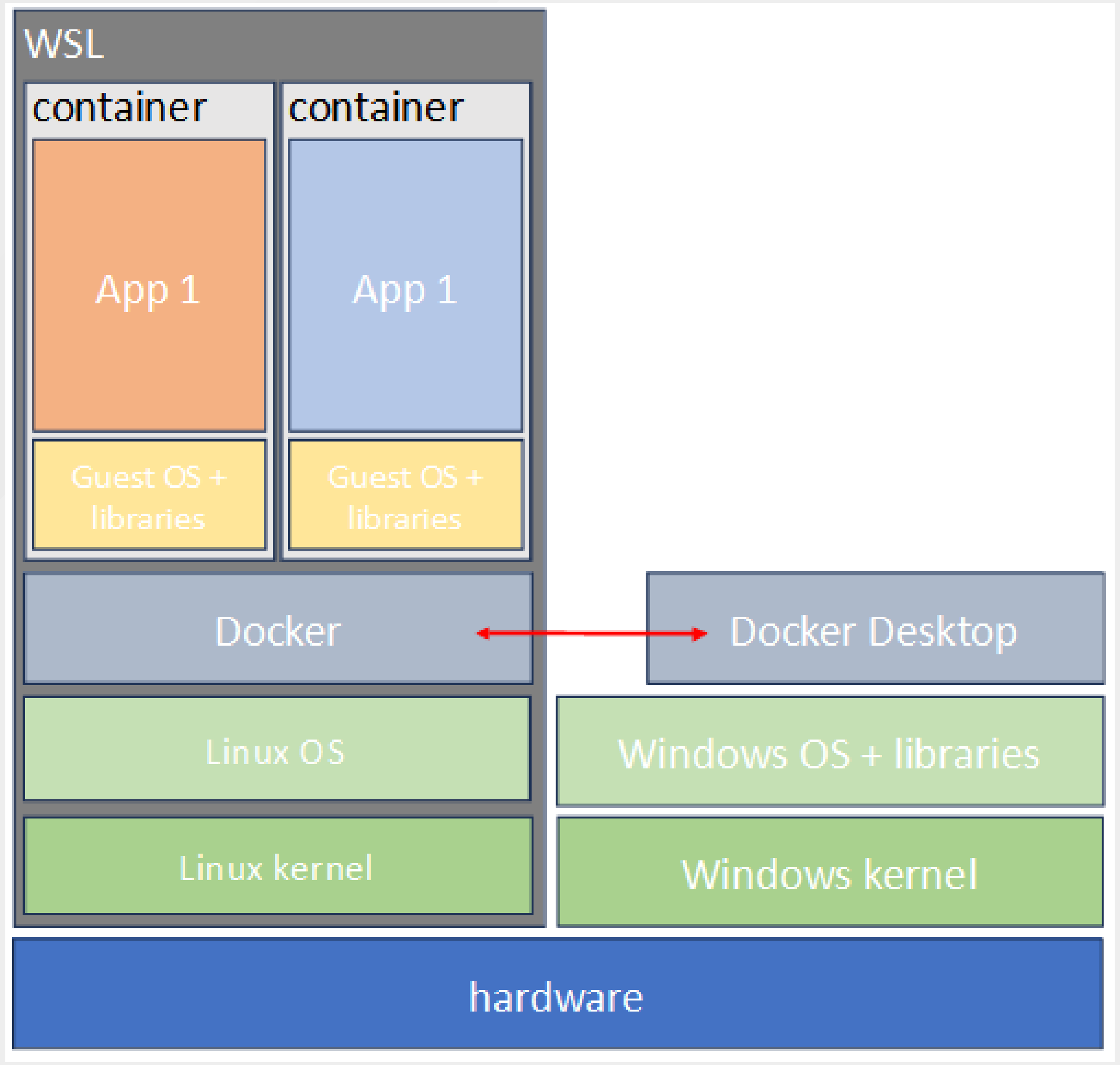
Devcontainers and VSCode

- Seamless: functions as local instance
- Executes build tools, debugger and other tools from container
- VSCode Configuration and plugins declared in the json file
- Make sure to install the Dev Containers plugin `ms-vscode-remote.remote-containers`
- Good to use `mcr.microsoft` docker images as BASE:
    - `mcr.microsoft.com/vscode/devcontainers/javascript-node`

Docker on Windows

Docker is in it's core a linux tool

- Containers themselves on WSL
- USB to WSL: USBIP
  - see readme


WSL
container
container
App 1
App 1
Guest OS + libraries
Guest OS + libraries
Docker
Docker Desktop
Linux OS
Windows OS + libraries
Linux kernel
Windows kernel
hardware