Devcontainers and Embedded software development

Down the rabbit hole to never come back

TODO Layout

TODO So... what's the problem?

Can we solve it?

- Just install and configure the lot
- Remote development (e.g. over ssh)
- Combine all tools, libraries and the lot
  - 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**. Dev containers can be run **locally or remotely, in a private or public cloud**, in a variety of supporting tools and editors.*

Down the rabbit hole: containers

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

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

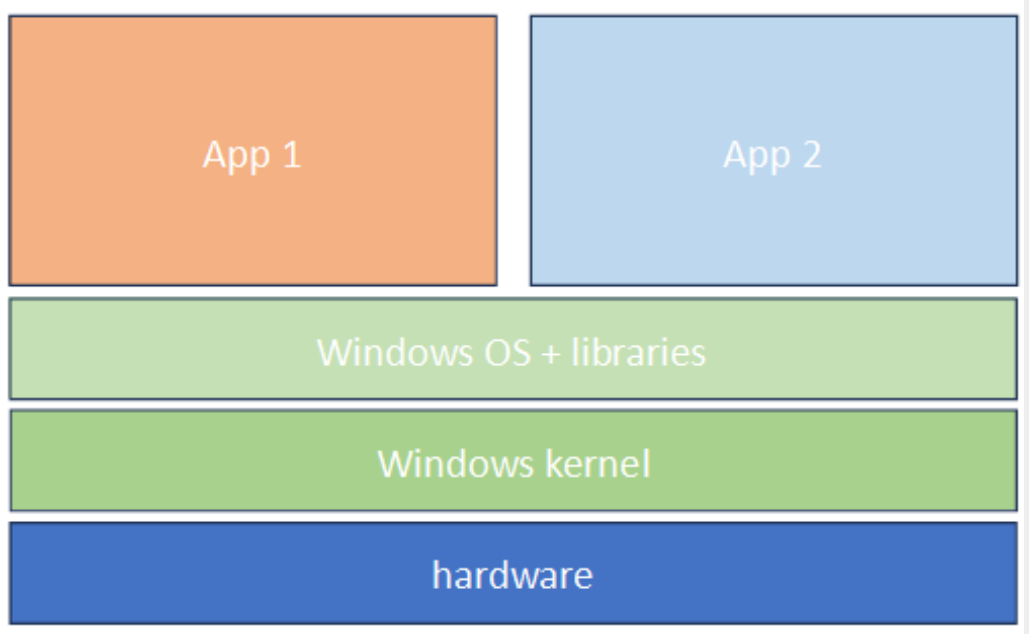
Down the rabbit hole: containers

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

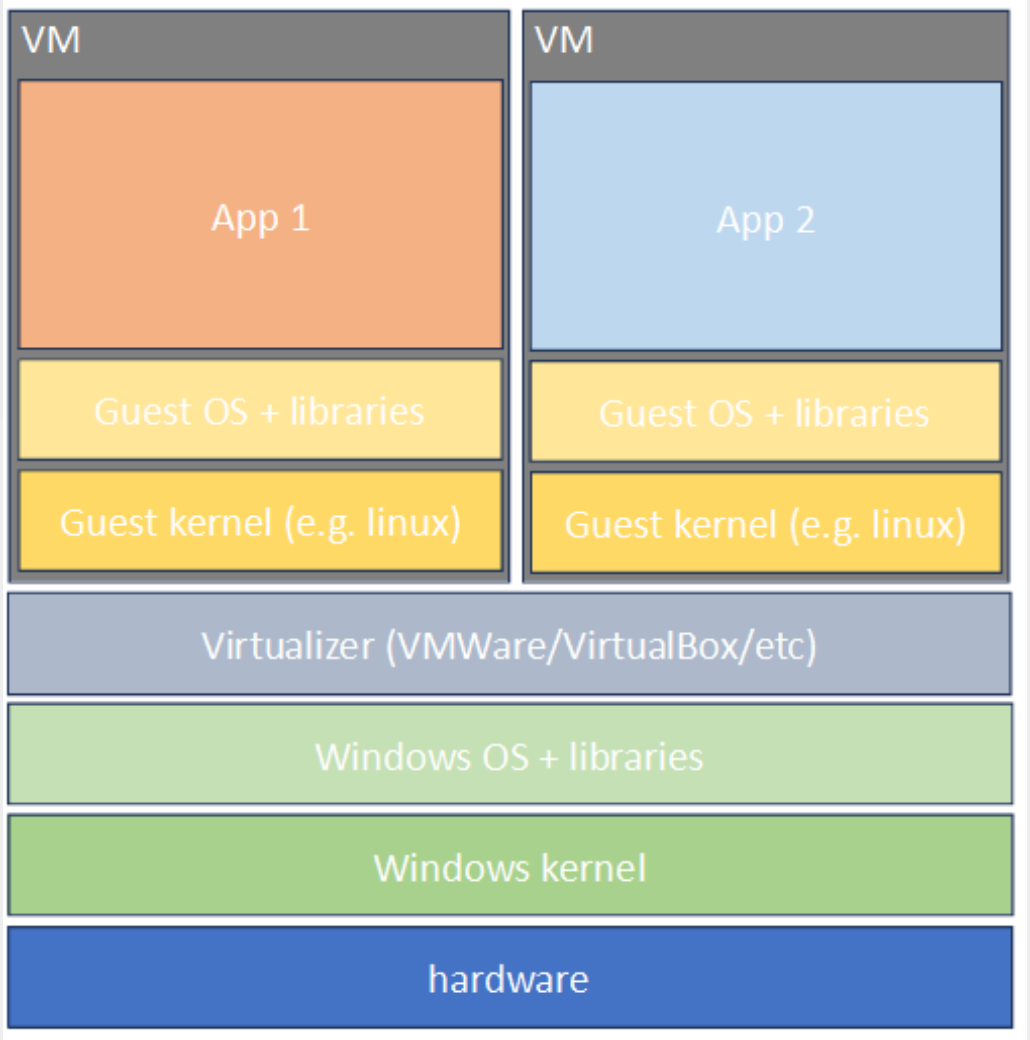
Like a VM and lightweight

Applications on top of OS



Like a VM and lightweight

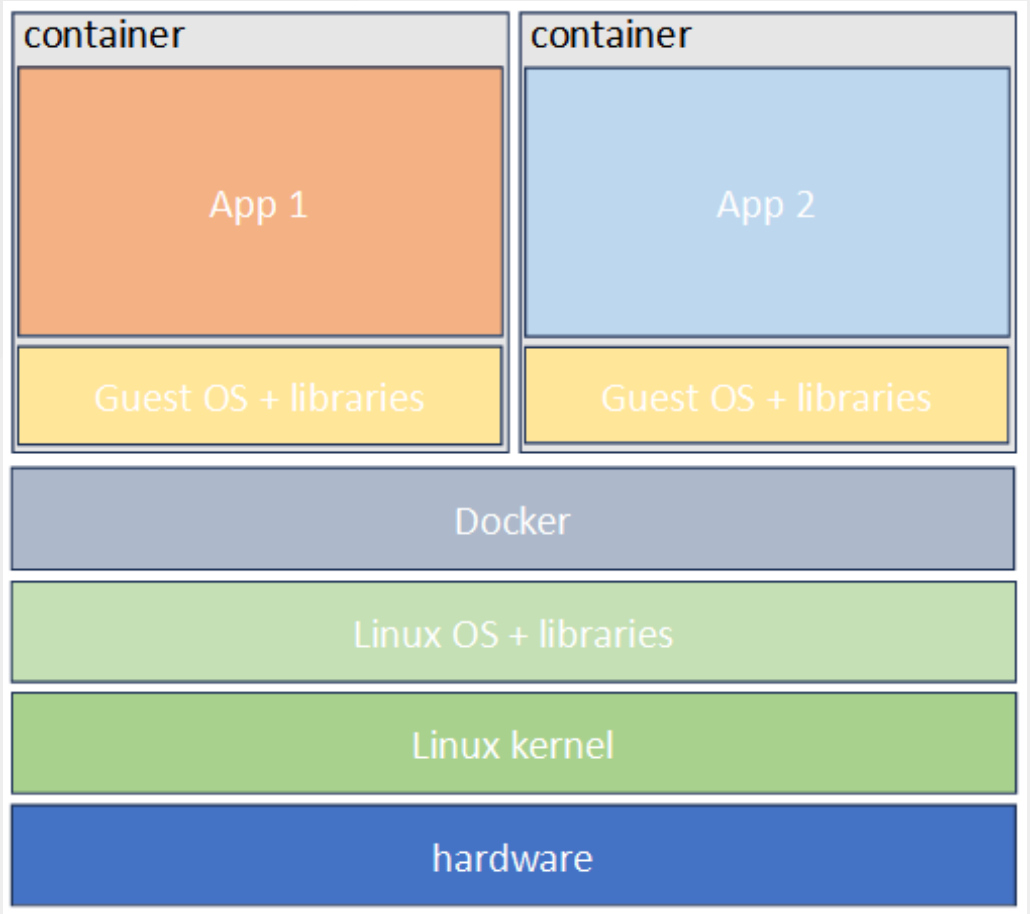
Applications in VM's



Like a VM and lightweight

Applications in Docker containers

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



Declarative: Infrastructure as Code

```
FROM debian
ENV DEBIAN_FRONTEND=noninteractive

RUN apt update -y && apt install -y \
    build-essential \
    cmake

RUN useradd -ms /bin/bash someuser
WORKDIR /home/someuser
USER someuser

CMD /bin/bash
```

Declarative: Infrastructure as Code

- To create a container image:

```
docker build --tag 'debian_example' .
```

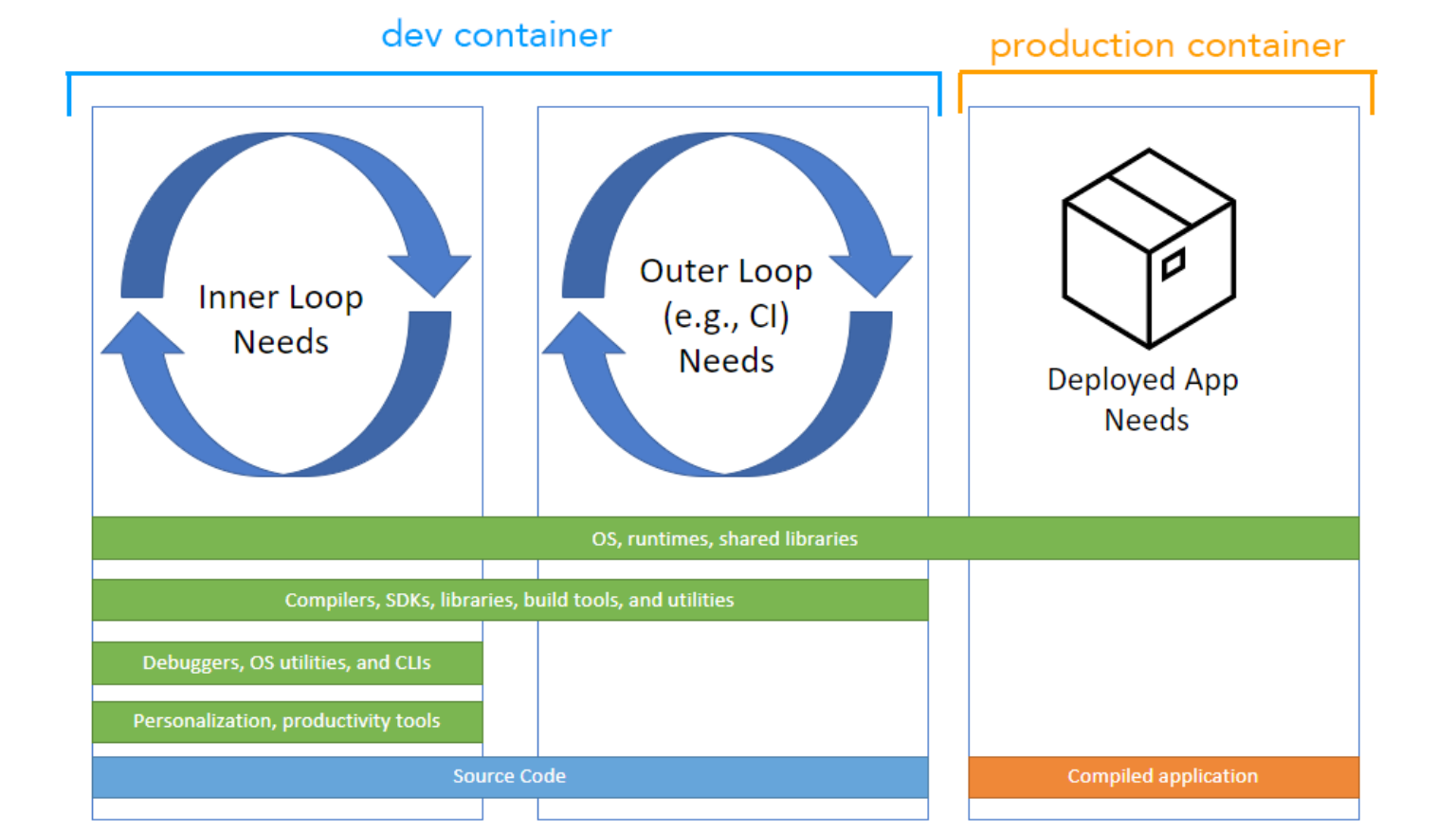
- To run a container image:

```
docker run -it 'debian_example'
```

- This will get you a Debian shell

```
$ docker run -it debian_example
someuser@e9bf5806d588:~$ pwd
/home/someuser
someuser@e9bf5806d588:~$
```

Devcontainers: containers with spice



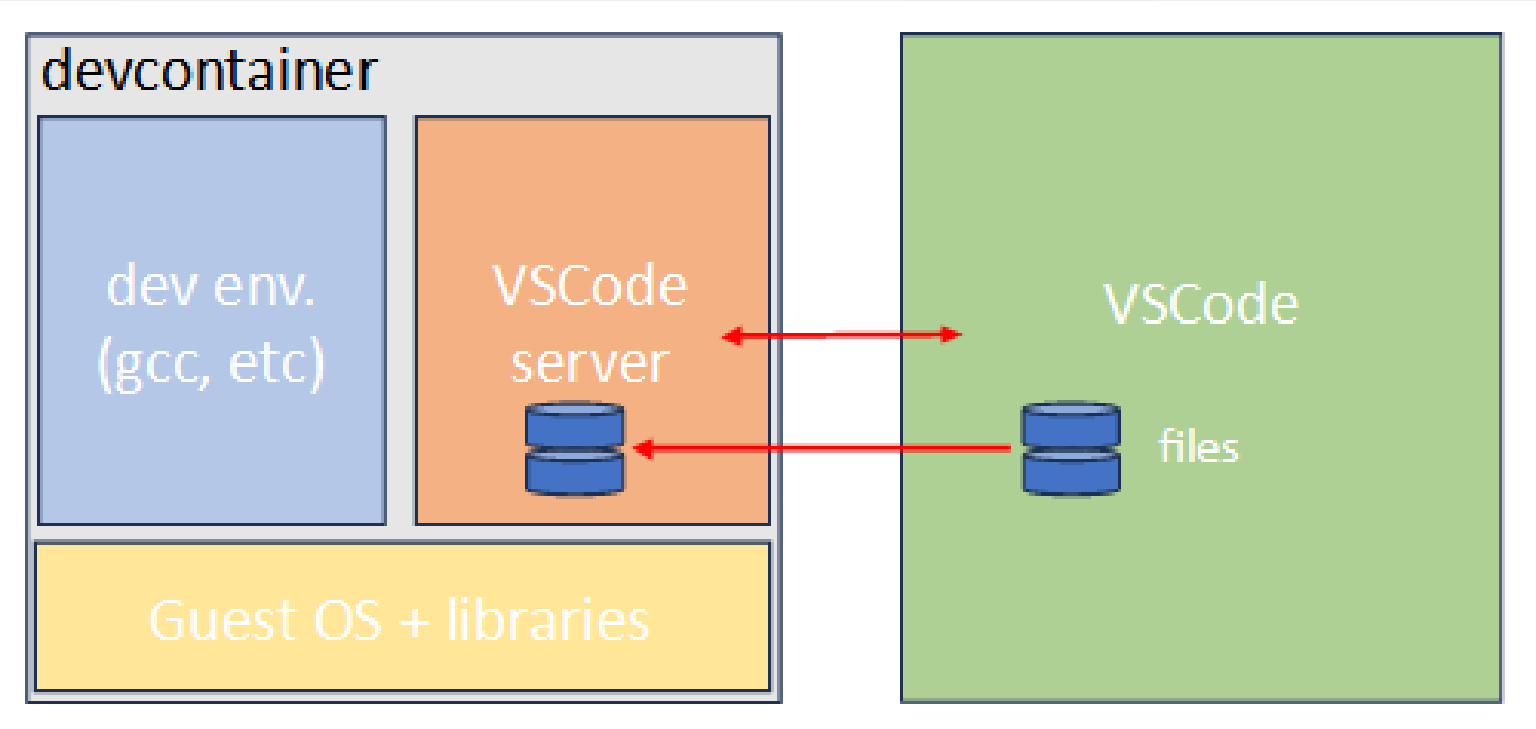
Devcontainers: containers with spice

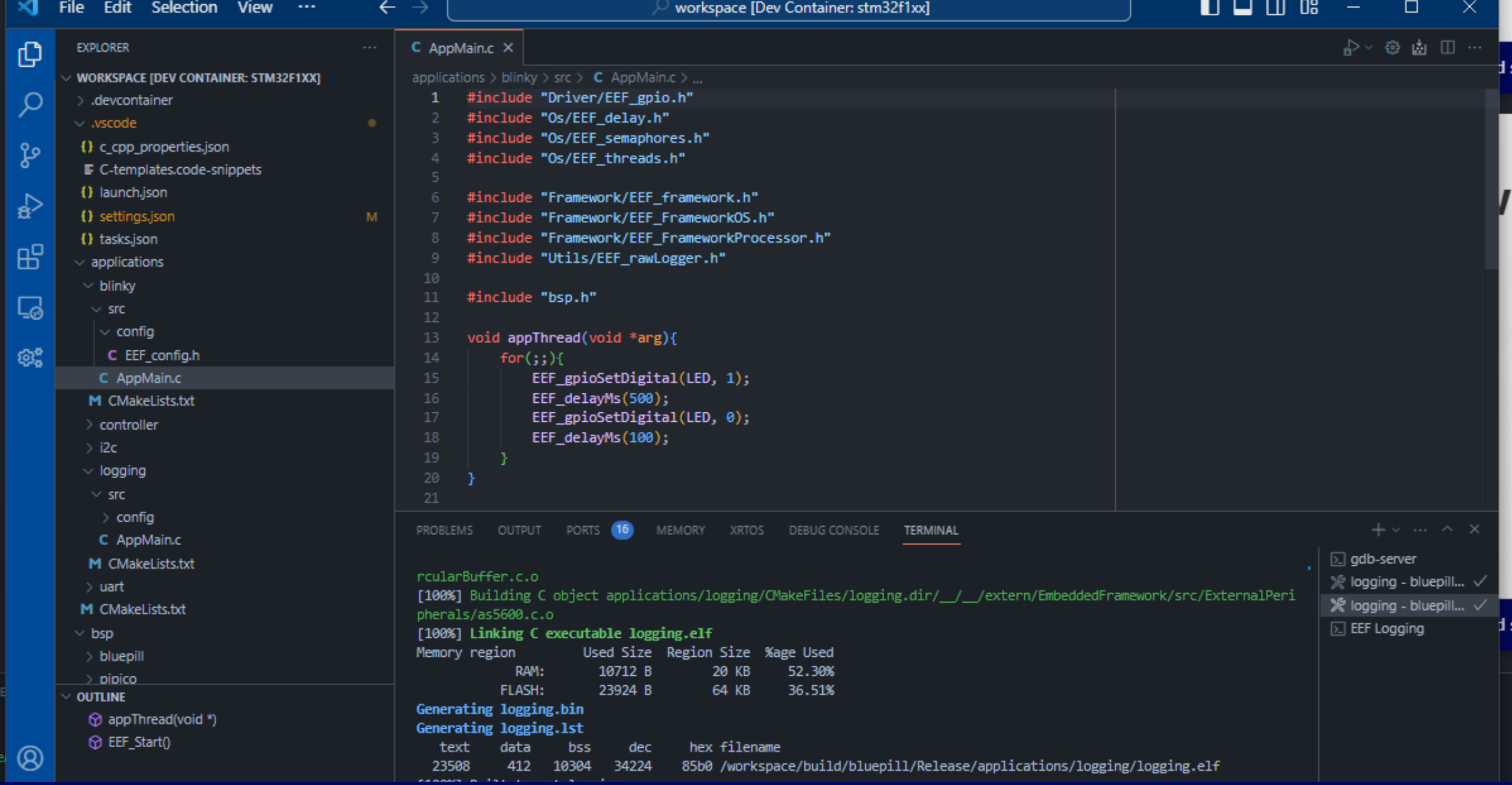
- Full development environment
- Utilities and personalization
- Configuration of IDE

Devcontainers: an example

```
{
    "build": { "dockerfile" : "Dockerfile" },
    "customizations": {
        "vscode": {
            "extensions" : [
                "ms-vscode.cpptools-extension-pack",
                "ms-vscode.cpptools",
                "ms-vscode.cmake-tools",
            ]
        }
    }
}
```

Devcontainers and VSCode




File Edit Selection View ···
workspace [Dev Container: stm32f1xx]
EXPLORER
WORKSPACE [DEV CONTAINER: STM32F1XX]
.devcontainer
.vscode
c_cpp_properties.json
C-templates.code-snippets
launch.json
settings.json M
tasks.json
applications
blinky
src
config
EEF_config.h
AppMain.c
CMakeLists.txt
controller
i2c
logging
src
config
AppMain.c
CMakeLists.txt
uart
CMakeLists.txt
bsp
bluepill
pipico
OUTLINE
appThread(void *)
EEF_Start()
AppMain.c
applications > blinky > src > AppMain.c > ...
1 #include "Driver/EEF_gpio.h"
2 #include "Os/EEF_delay.h"
3 #include "Os/EEF_semaphores.h"
4 #include "Os/EEF_threads.h"
5
6 #include "Framework/EEF_framework.h"
7 #include "Framework/EEF_FrameworkOS.h"
8 #include "Framework/EEF_FrameworkProcessor.h"
9 #include "Utils/EEF_rawLogger.h"
10
11 #include "bsp.h"
12
13 void appThread(void *arg){
14 for(;;){
15 EEF_gpioSetDigital(LED, 1);
16 EEF_delayMs(500);
17 EEF_gpioSetDigital(LED, 0);
18 EEF_delayMs(100);
19 }
20 }
21
PROBLEMS OUTPUT PORTS 16 MEMORY XRTOS DEBUG CONSOLE TERMINAL
rcularBuffer.c.o
[100%] Building C object applications/logging/CMakeFiles/logging.dir/__/__/extern/EmbeddedFramework/src/ExternalPeripherals/as5600.c.o
[100%] Linking C executable logging.elf
Memory region Used Size Region Size %age Used
RAM: 10712 B 20 KB 52.30%
FLASH: 23924 B 64 KB 36.51%
Generating logging.bin
Generating logging.lst
text data bss dec hex filename
23508 412 10304 34224 85b0 /workspace/build/bluepill/Release/applications/logging/logging.elf
gdb-server
logging - bluepill...
logging - bluepill...
EEF Logging

Devcontainers and VSCode

- Functions as local instance
- Executes build tools, debugger and other tools from container
- VSCode Configuration and plugins declared in the json file

The full story? Nope...

We develop firmware, not normal applications or web-apps

The full story? Nope...

We develop firmware, not normal applications or web-apps

... So I lied?

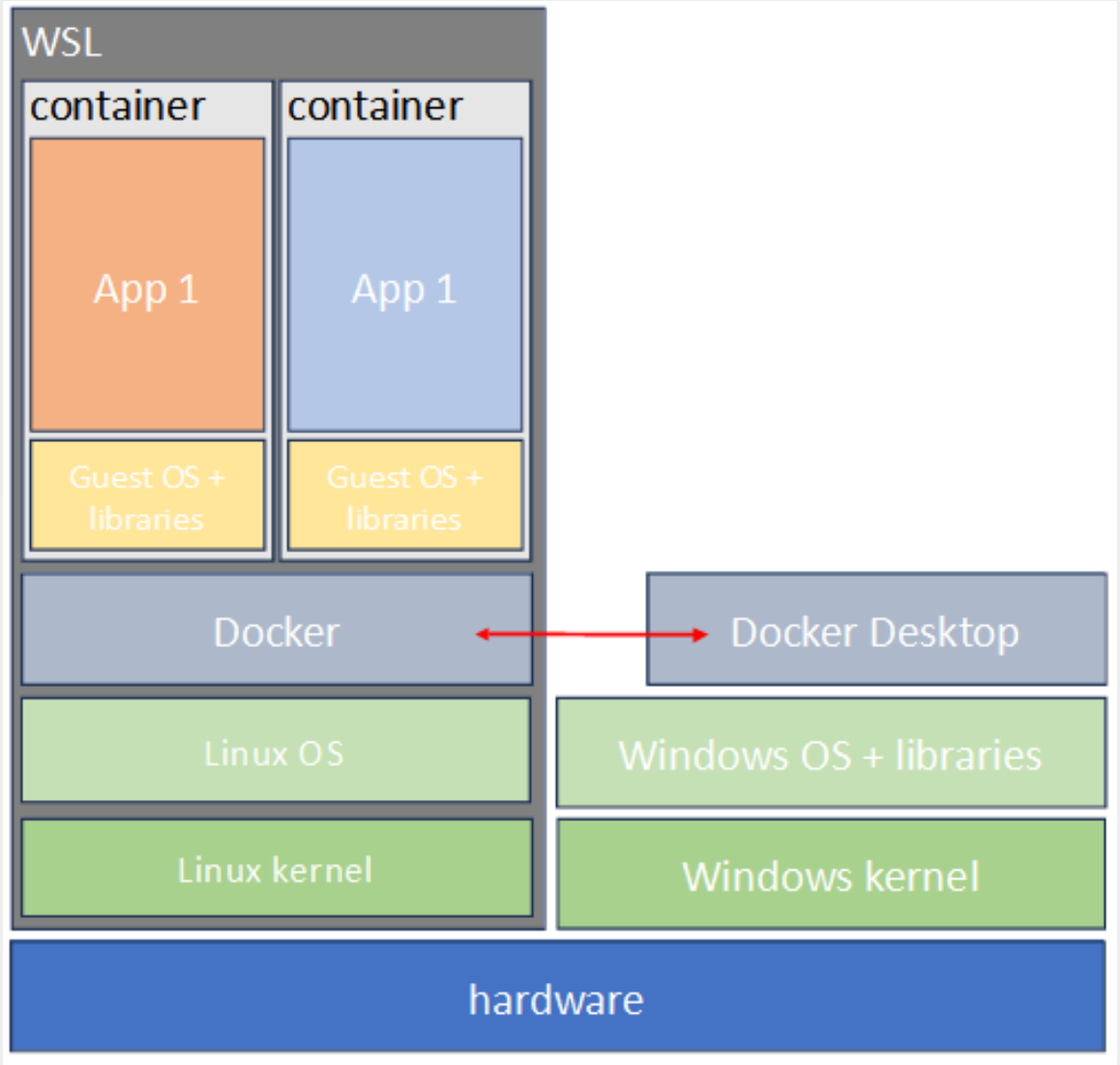
Embedded development

- Debugging of non-native applications
- Use of physical debugger (e.g. JLink/STLink/BMP)
- **we need USB**
  - so just pass the USB to the container?
  - .....

Docker on Windows

Docker is in it's core a linux tool

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



All together

- Toolchain, debugging tools and IDE configurations packaged together
- Declarative
- Lightweight
- Multi-purpose: development and CI/CD