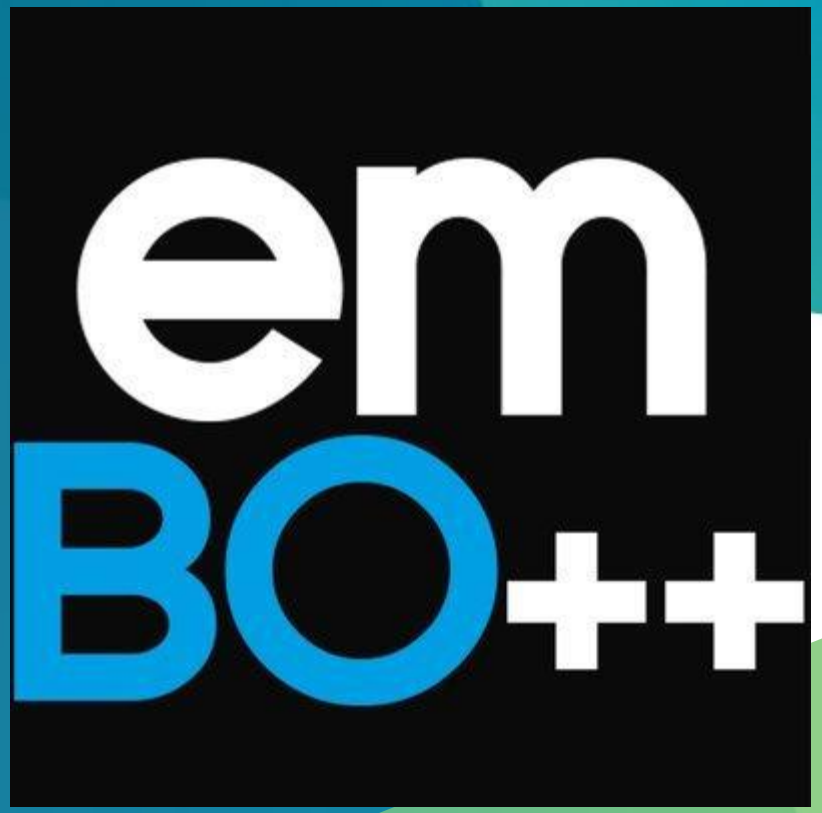


HEXAGON | Leica Geosystems

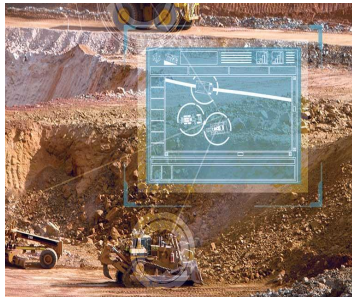
Hands-on: Molding VS Code into a care-free development environment to develop with Zephyr RTOS

Jonas Nussdorfer

Global leader in **sensor**, **software**,
and **autonomous** technologies committed to

empowering an autonomous future

Divisions and their products

Survey Solutions	Machine Control	Building	Geospatial	Mining
				
				
				

HEXAGON

Agenda

- Overview over Zephyr RTOS
  - Brief introduction
  - Typical project setup workflow
- Our development setup
  - Configuring VS Code
  - Evolution of the setup
- Demo of 1st time setup
- Summary & Outlook

Getting started with Zephyr

HEXAGON Leica Geosystems

Getting started with Zephyr

What is Zephyr?

From the Zephyr doc pages

- RTOS which is based on a small-footprint kernel
- Designed for resource-constrained and embedded systems

Some key features

- Highly configurable and modular setup (Kconfig)
- Hardware configuration described by devicetree
- Cross-architecture with wide range of supported boards
- Native development on Linux, macOS and Windows
- Native POSIX port, i.e. run your Zephyr application as a Linux application
- …

Getting started with Zephyr

Setting up a workspace

- Install dependencies
  - CMake
  - Python
  - Devicetree compiler
- Install Zephyr SDK
  - Download and install the SDK

Docker Container

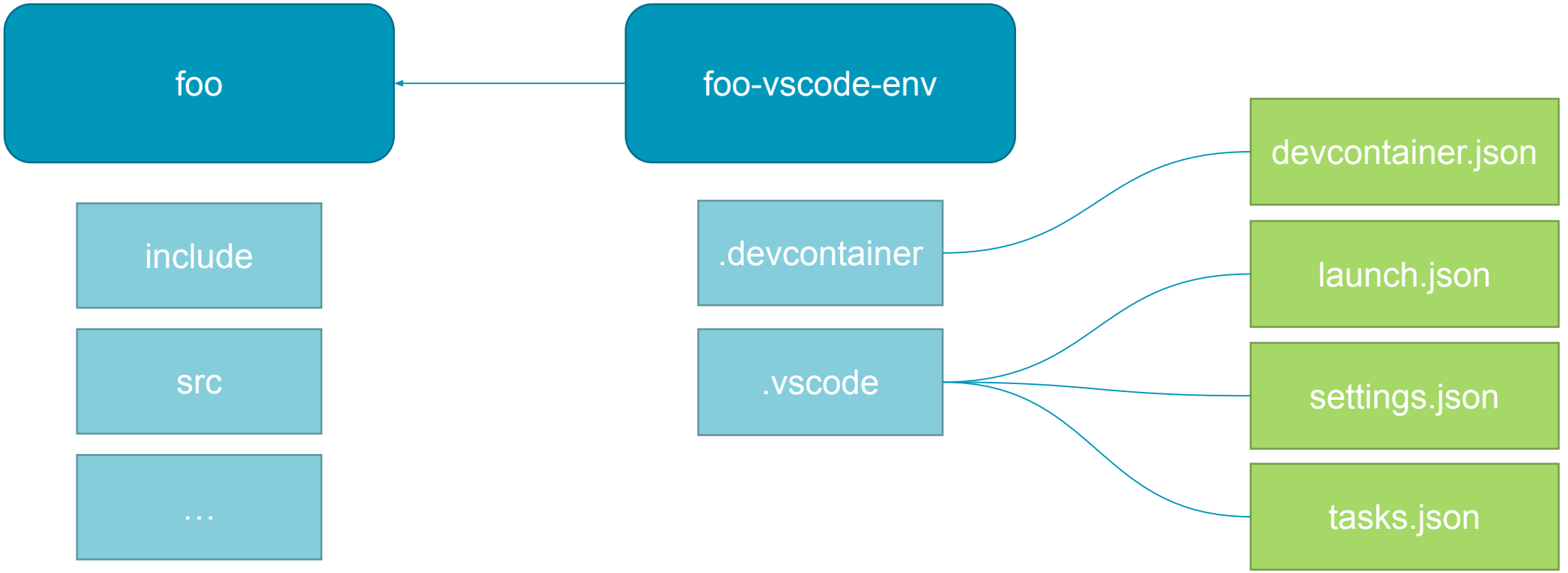
- Set up Zephyr
  - Install west (Zephyr build tool)
  - Initialize project directory via west and update
  - Export Zephyr CMake package
  - Install additional Python dependencies via pip

Custom Steps

Setting up the development environment

Setting up the development environment

First approach



HEXAGON

Development Containers

Docker container configuration with devcontainer.json

Configure what the container needs to run:

- Name
- Image specification or reference to Dockerfile from which to build
- Mounted folders
- Run arguments
- Container environment variables (alternatively via environment file)
- VS Code extensions

devcontainer.json

root > .devcontainer > devcontainer.json > ...

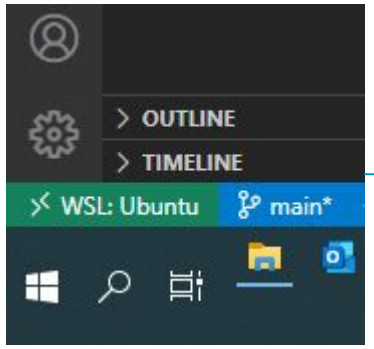
Jonas, 2 days ago | 2 authors (Jonas and others)

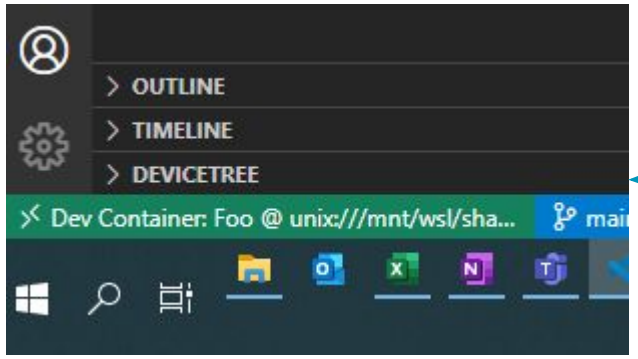
```
{
    "name": "Zephyr VS Code Template",
    "build": {
        "dockerfile": "Dockerfile"
    },
    "mounts": [
        "type=bind,source=${env:HOME}/.ssh,target=/home/zephyr/.ssh"
    ],
    "runArgs": [
        "--cap-add=SYS_PTRACE",
        "--security-opt",
        "seccomp=unconfined"
    ],
    "customizations": {
        "vscode": {
            "extensions": [
                "ms-vscode.cpptools",
                "ms-vscode.cmake-tools",
                "matepek.vscode-catch2-test-adapter",
                "actboy168.tasks",
                "eamodio.gitlens",
                "twxs.cmake",
                "asciidoctor.asciidoctor-vscode",
                "jebbs.plantuml"
            ]
        }
    }
}
```

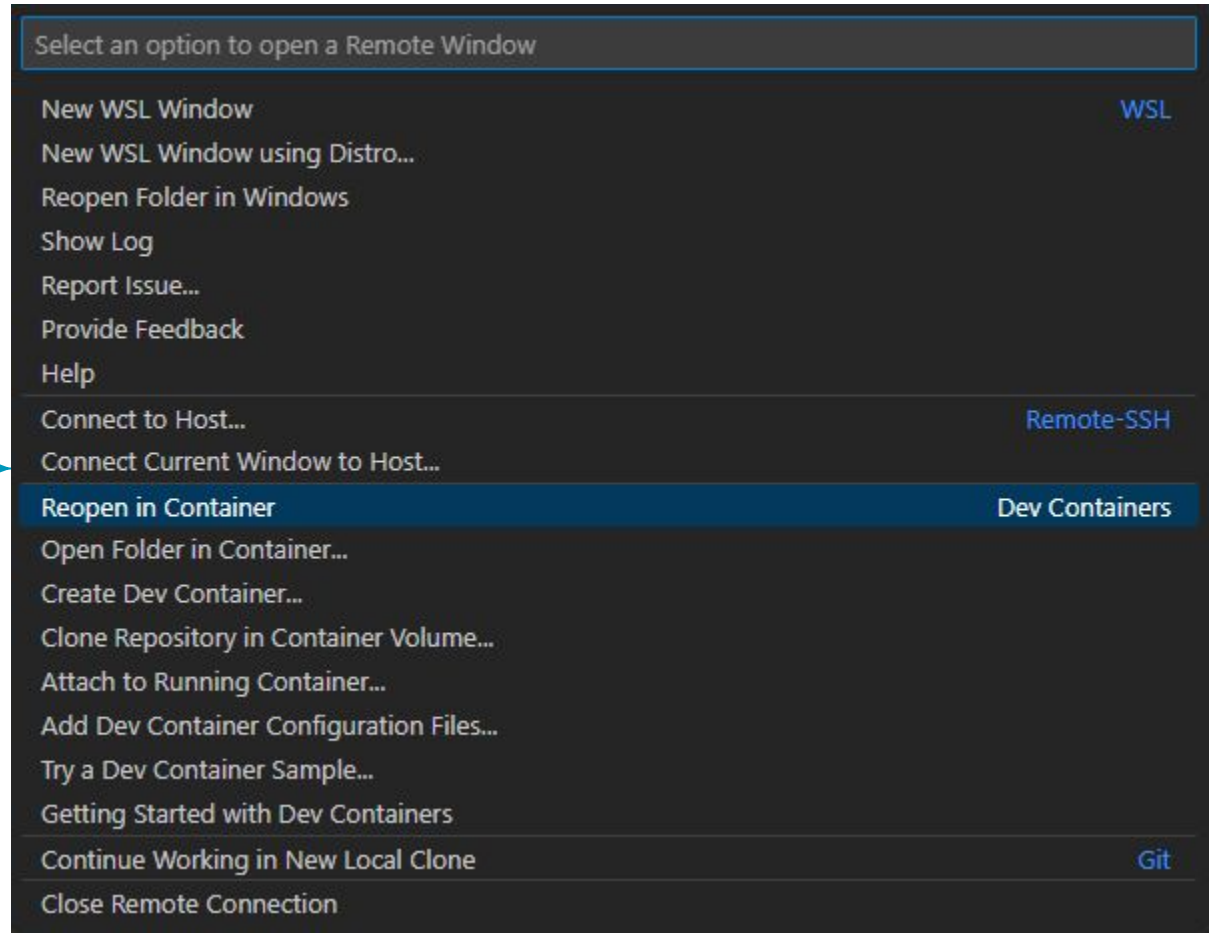

HEXAGON

Development Containers

Setting up VS Code with devcontainer.json (1st time)







HEXAGON

Development Containers

Dev containers in VS Code wrapped up

Required extensions

- Docker
- Dev Containers
- WSL (if on Windows)

No extensions specific to development required

Remarks for Windows

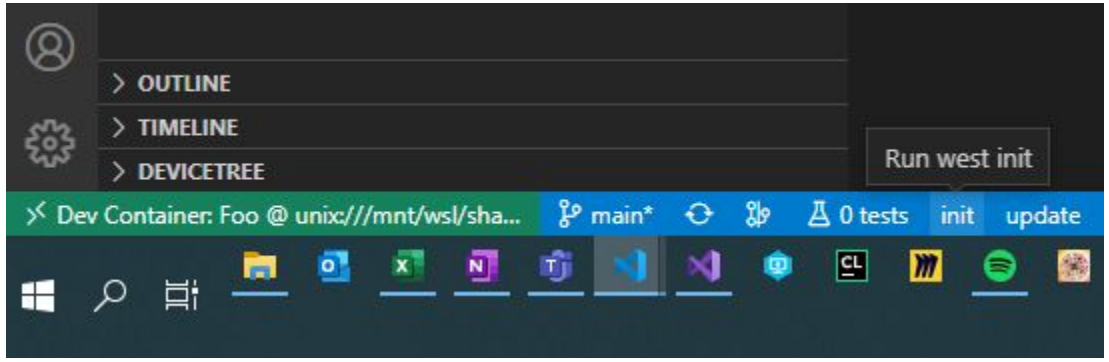
- Docker requires **WSL2** (Docker Desktop for Windows can be used, but WSL2 variant is free)
- Remember to pay attention to potential line ending conversions
- Code should be «hosted» in WSL for better performance (inside virtual hard disk/vhdx)

VS Code Configuration

Customizing Zephyr setup steps with tasks.json

Recall from Zephyr introduction:

- Initialize project with west (Zephyr meta-tool)
- Install some additional dependencies via pip
- Flash your device
- …



tasks.json U

tasks.json > ...

```
{
    "version": "2.0.0",
    "problemMatcher": [],
    "tasks": [
        {
            "label": "init",
            "detail": "Run west init",
            "type": "shell",
            "command": "west init --local firmware",
            "options": {
                "cwd": "${workspaceFolder:root}",
                "statusbar": {
                    "hide": false
                }
            }
        },
        {
            "label": "update",
            "detail": "Run west update",
            "type": "shell",
            "command": "west update",
            "options": {
                "cwd": "${workspaceFolder:root}",
                "statusbar": {
                    "hide": false
                }
            }
        }
    ]
}
```

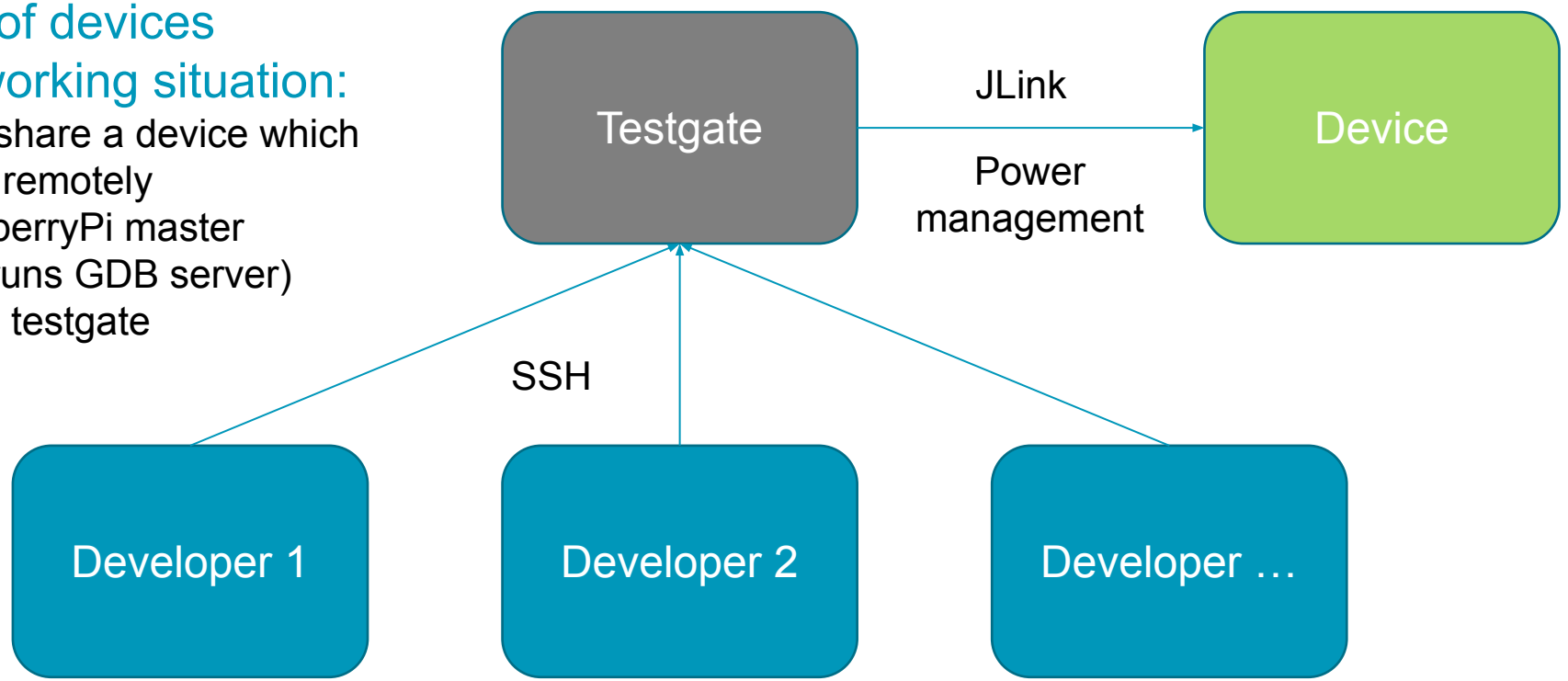

HEXAGON

VS Code Configuration

Remote debugging

Due to limited number of devices available and remote working situation:

- Multiple developers share a device which has to be debugged remotely
- Connection to RaspberryPi master (testgate) via SSH (runs GDB server)
- Device connected to testgate



VS Code Configuration

Houston, we have liftoff with launch.json

Define launch configurations:

- Name
- Executable
- Device
- GDB settings
- Tasks to execute before launch
- …

HEXAGON

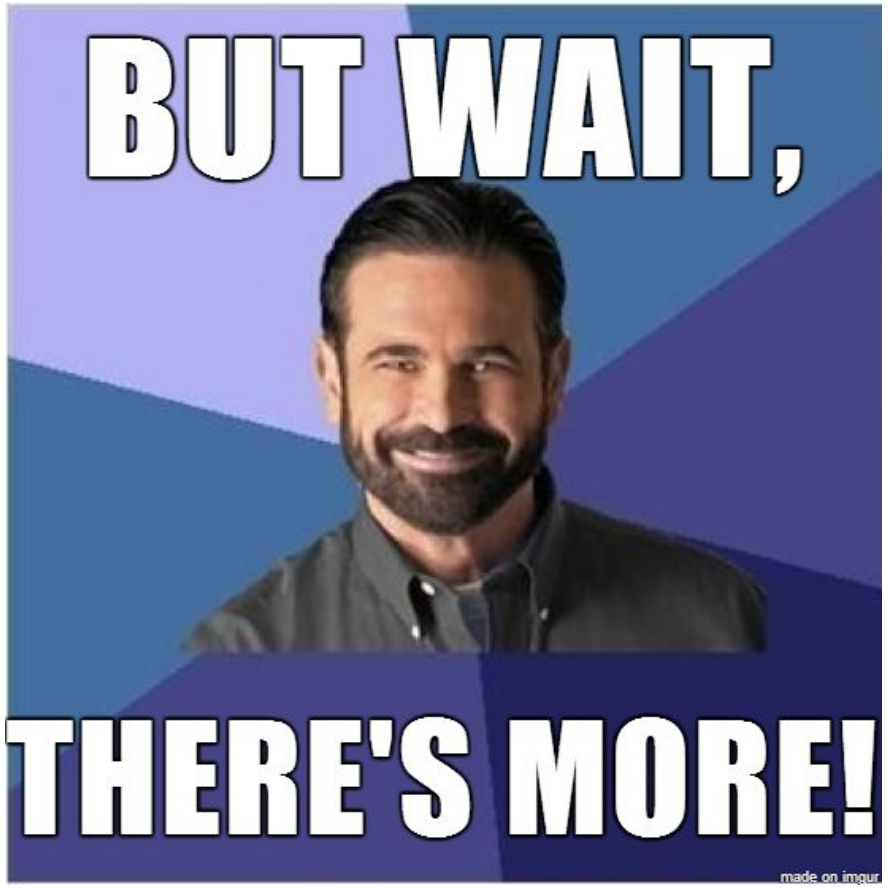
VS Code Dev Setup

Recap until now

How mature is the setup?

- ✔ Custom steps can be represented by a task, grouping tasks also possible
- ✔ Everything that is needed to develop (dependencies, tooling, etc…) is inside the docker container
- ✔ Remote debugging is set up and configured by a launch file
- ✔ Developer only needs a minimal setup (Docker, VS Code + some extensions, git)

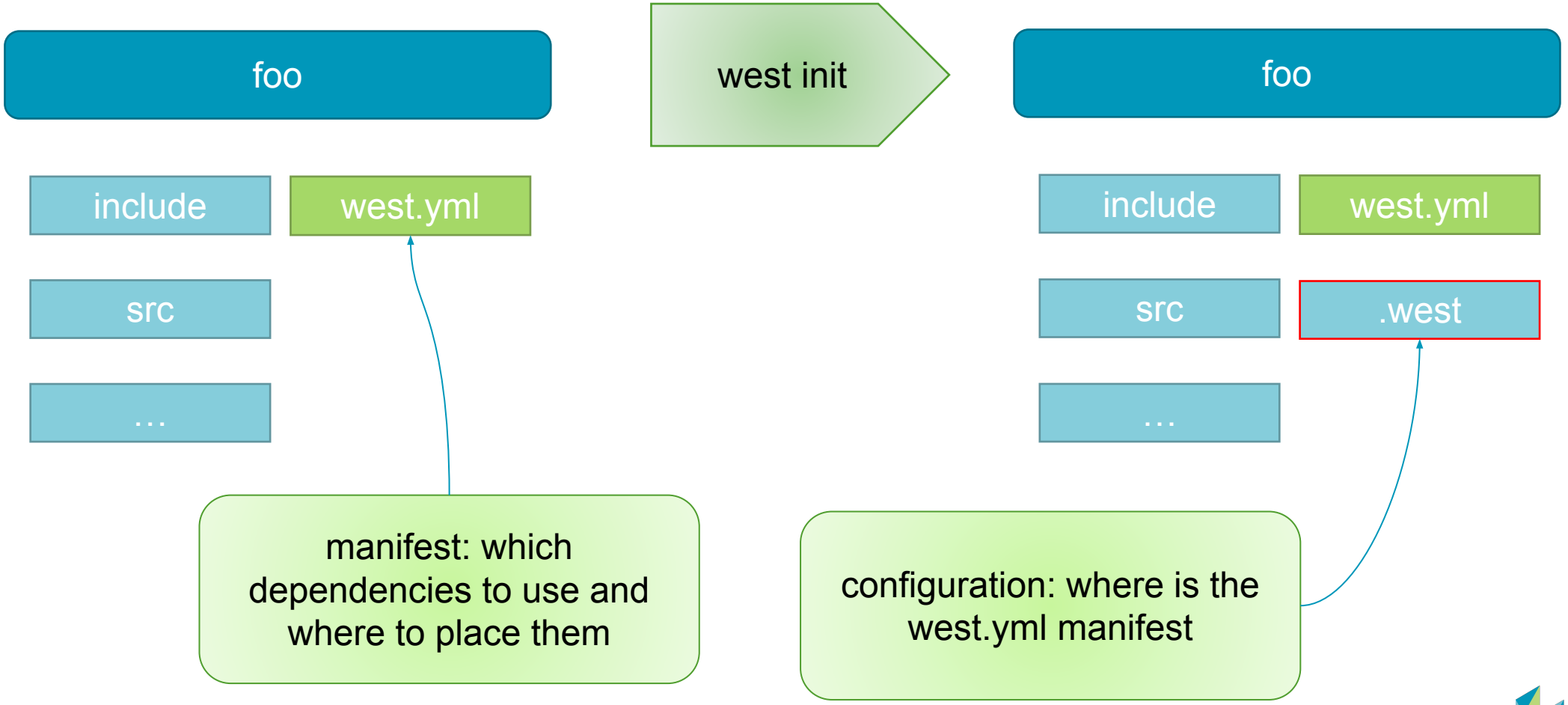
At this point anyone can clone the repo and collaborate right away without being plagued by a complex development environment setup



HEXAGON

Zephyr Project Structure

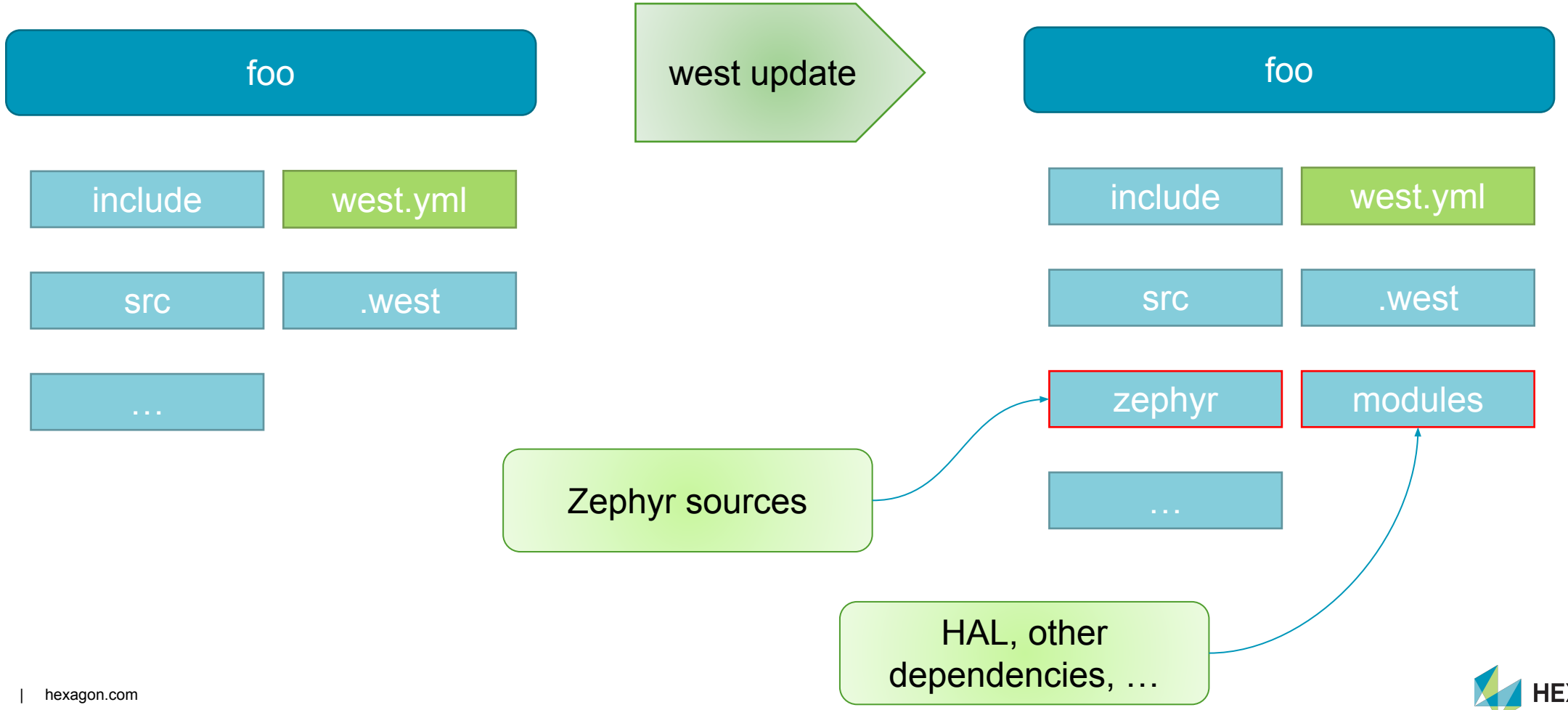
Initializing



HEXAGON

Zephyr Project Structure

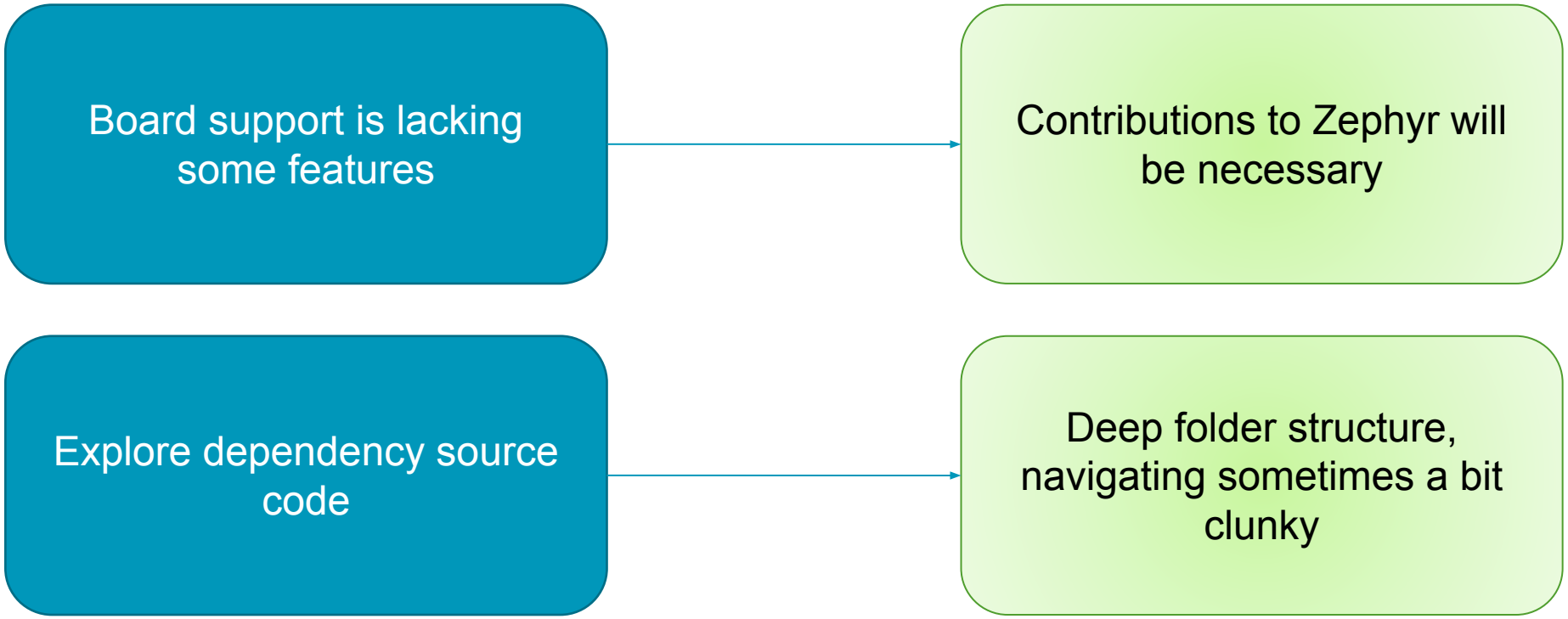
Retrieving Zephyr sources and dependencies



HEXAGON

Development Pains

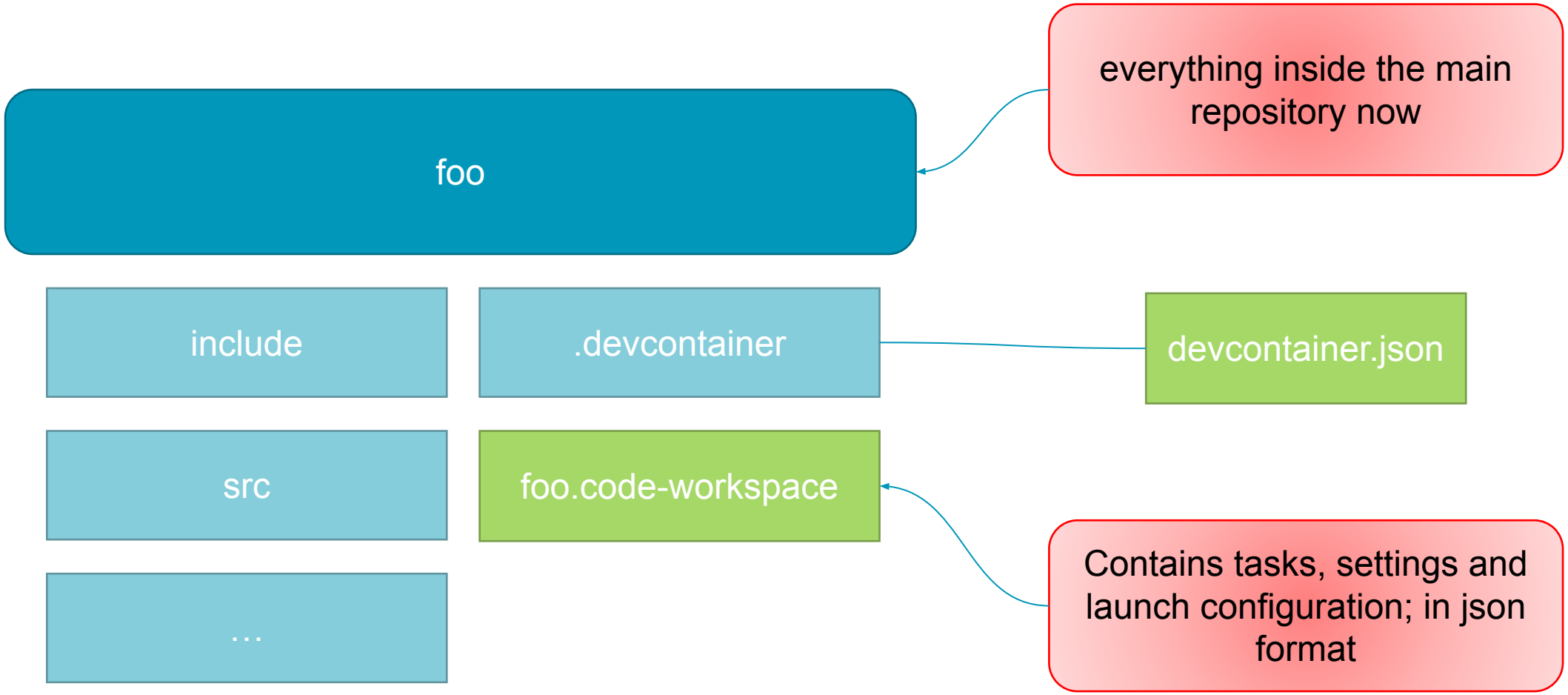
Interlude



HEXAGON

VS Code Workspaces

Redux of the development setup

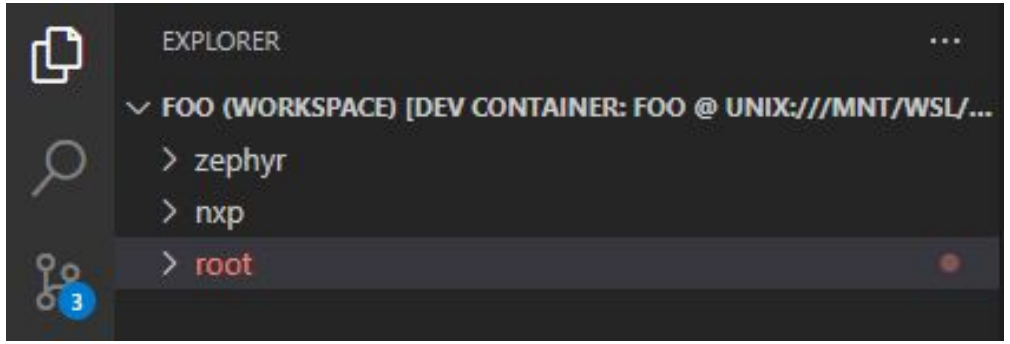


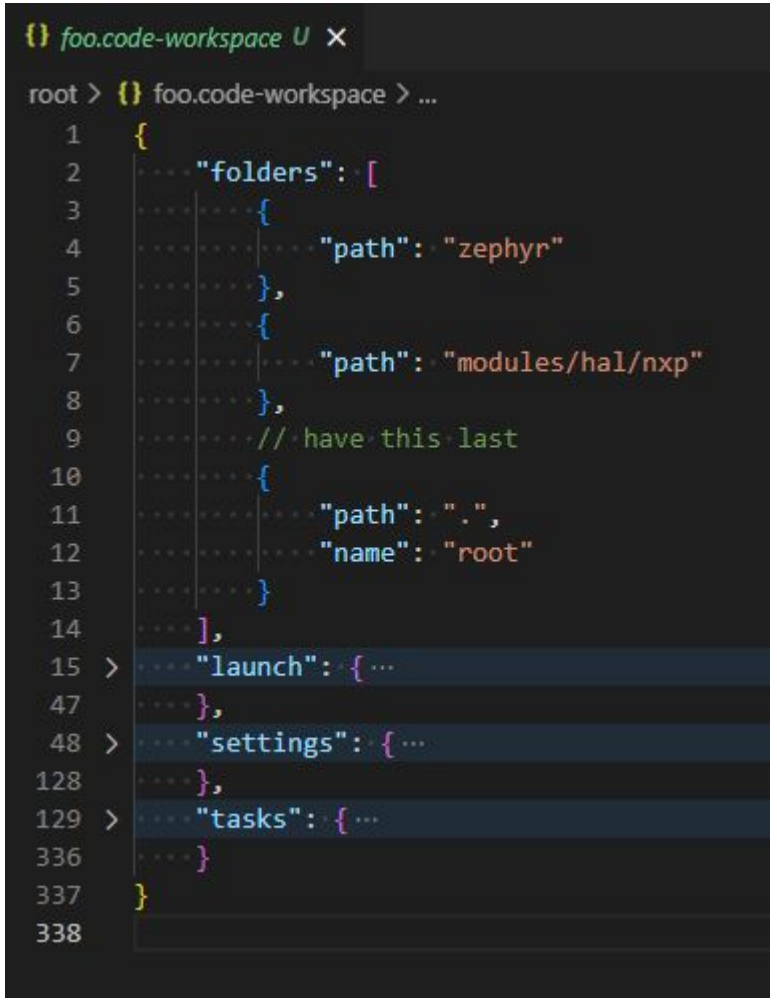
HEXAGON

VS Code Workspaces

Multiple repositories/folders

- Workspace files enable working on multiple projects in the same VS Code instance
- Folders that used to be buried deeper in the project are now made more accessible
- Note: Child nodes for launch, settings and tasks are now absorbed in the workspace file





HEXAGON

Demo

Select nuj@AHERDSKNUJ01: ~/Projects/embo

```
nuj@AHERDSKNUJ01:~/Projects/embo$
```

Outlook

Summary

Looking beyond

Some success stories after 1 year:

- Used by our team and a few other teams working on Zephyr projects
- New colleagues had a much easier time getting started
- Other teams working on Windows but developing inside a docker container have also started using this setup pattern (different container and extensions)
- Recently we ported a legacy project to cmake where we also introduced this setup (Freescale/NXP MQX based embedded project)

Link to GitHub repository:

We’re hiring!

Follow us

HEXAGON | Leica Geosystems

Thank you!

HEXAGON Leica Geosystems

Extra Slides

Additional Tools and Frameworks

Things we picked up along the way

- CMakePresets.json
  - Predefined configurations to build (e.g. build type, cache variables, toolchain, etc…)
- AsciiDoc
  - Extension for VS Code so we can edit the project documentation
- PlantUML
  - Installed in container, enhance documentation with UML
- Dotnet
  - System automation tests are C#-based
  - Installed in docker container + VS Code extension
  - Possible to execute in container AND debug simultaneously with embedded application
- Puncover
  - tool that allows to see which parts of the code base are expensive with respect to size