Zephyr® Project

Developer Summit 2022

June 8-9, 2022

Mountain View, CA + Virtual

Zephyr® Project
Developer Summit 2022

June 8-9, 2022
Mountain View, CA + Virtual

Leveraging Cloud Technologies for Development and Operation of Zephyr RTOS

Rob Woolley, Wind River

Why?

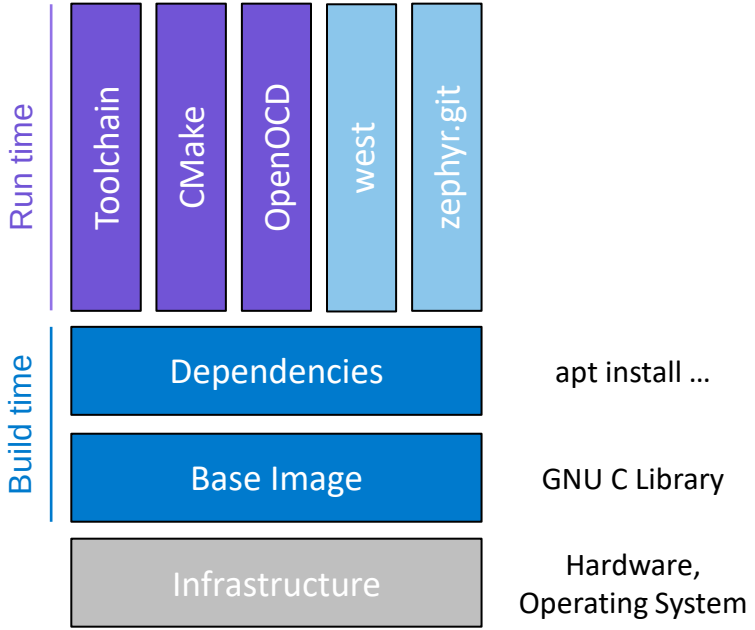
- OS development can be difficult for beginners
  - Requires new skills (compiling, flashing, booting, etc.)
  - Uses legacy or specialized tools like Bash, minicom, OpenOCD, git, west
  - Challenges with cross-platform tooling (e.g. Windows) and cross-architecture (e.g. ARM64)
- Development environments are hard
  - Setting up a development environment for the first time
  - Synchronizing the dev env across machines or between developers
  - Reproducing the same development environment years later to resolve a critical CVE
- Cloud methodologies help scale
  - Take advantage of building at scale in the cloud
  - Use a declarative approach for built-in automation and documentation
  - Stay independent of infrastructure
  - Reduce toil

Why Now?

- “Software is eating the world”, Marc Andreessen
- The demands on software development are only increasing:
  - Digital transformation is increasing the need for embedded and IoT development
  - Velocity and time to market shows no sign of slowing
  - 9 out of 10 companies are embracing free and open source software
  - Maintenance and security fixes for deployed devices (for 2 to 10 years)
- When you have global teams everyone is “working remotely”
- Bake-in best practices from subject matter experts for code quality including "shift left" and SBOMs (ie. CI/CD^2)
- Sharing methods for development, deployment, and operation between developers, testers, and operators (ie. DevSecOps)

What do I propose?

- Containers are the building block of cloud-native architecture
  - Size, Speed, Portability, Modularity, Self-sufficient
- Containers are not lightweight VMs, they are namespaces and resource limits applied to processes
- Separation of concerns
  - Build slow moving parts into the container image
  - Download fast moving parts at run-time (curl, git, etc)
  - Cache downloads for speed and long-term archiving
  - Store locally in reusable, read-only volumes

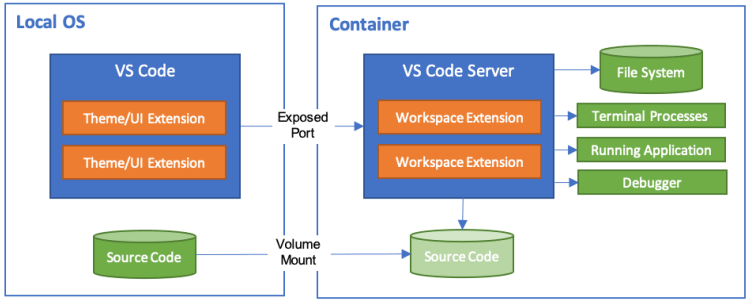


4 Types of Containers

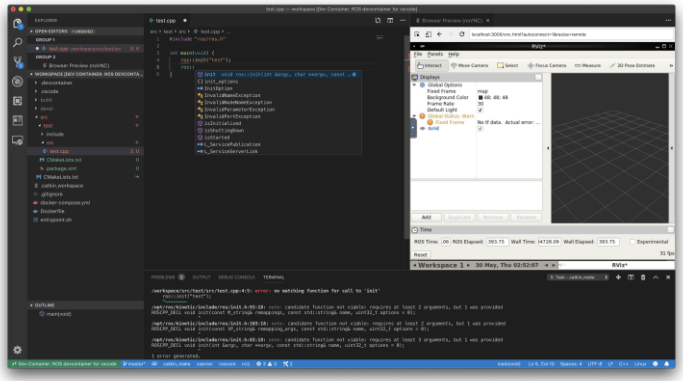
1. Development Container
2. Pipeline Container
3. Deployment Container
4. Operations Container

What is a Development Container?

- Full development environment that has all the dependencies and tools
- VS Code, CLion, Vim, and Emacs all have plugins and extensions for container-based development
- VS Code can even install extensions automatically on launch
- Can be built up beyond the RTOS to provide Zephyr samples or frameworks like the Robot Operating System (ROS) with extended tooling



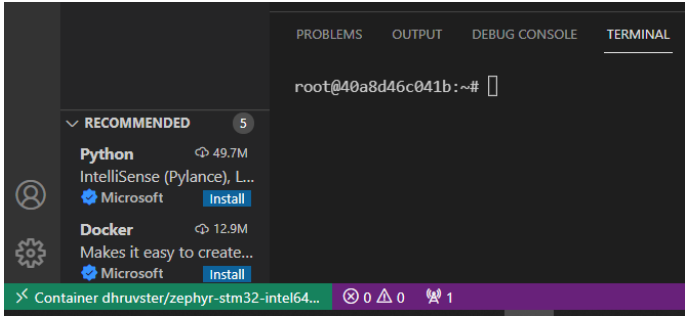
Microsoft VS Code "Developing Inside a Container"



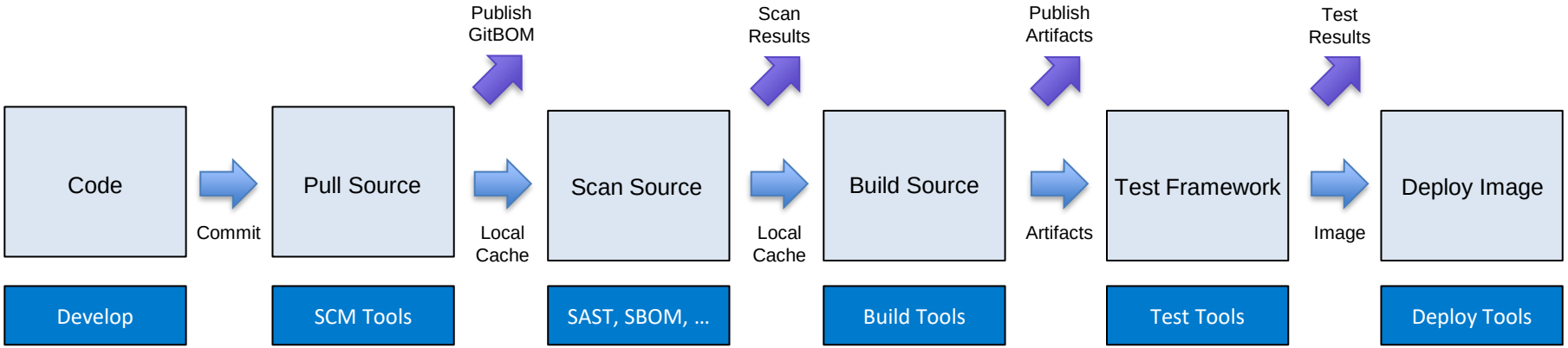
ROS dev container for VSCode on Mac

Zephyr DevContainer Example

- Dhruv Upadhyay prototyped a Zephyr DevContainer to build Zephyr for STM32 devices
- Approximately 3.99 GB and includes:
  - Ubuntu 20.04 LTS base image
  - Zephyr Host Dependencies
  - West and Python Dependencies
  - CMake
  - Zephyr SDK 0.13.2
  - Zephyr git repositories including modules
- In order to reduce the size of the devcontainer he supplied a custom west manifest file to limit the HAL to just stm32
- Successfully used it with VS Code and WSL2 to develop Zephyr application for the STM32L475 MCU
- Successfully ran on a Raspberry Pi 4, but the Zephyr build couldn't complete with only 2G of RAM



What is a Pipeline Container?



- Each container image may be specialized for its stage in the pipeline
- The pipelines may be customized with additional stage depending on the task (e.g. memory footprint reporting)
- Each stage may encapsulate best practices from subject matter experts (e.g. safety rules and security standards) and automatically produce a log of results
- Reports and artifacts may be saved for records, analytics, quality assurance, and even optimizing performance for iterative builds
- Declarative approach using configuration files (CaC)
- Commands executed in containers should be independent of the pipeline definition to be agnostic to a particular tool and support manual debugging

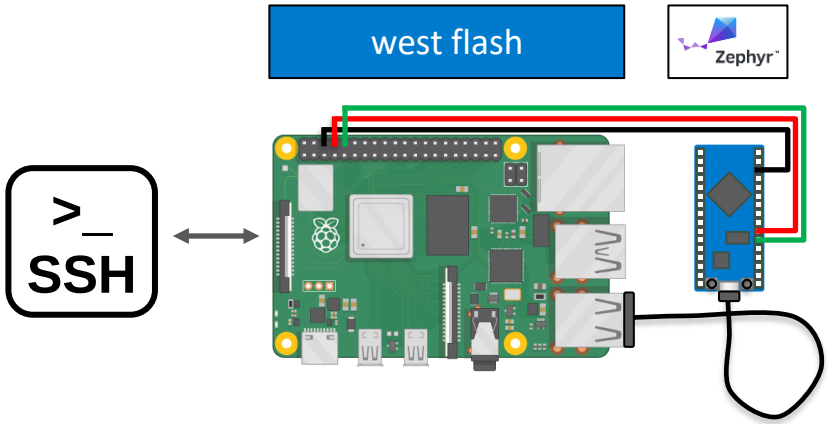
Practical Pipeline Example

- 3 levels of configuration:
  - **pipeline** – What stages should be run? What container images, volumes, envvars, and scripts should be used for each stage?
  - **project** – What configuration options are needed at a project level? Enabling and disabling features? Adding modules?
  - **device** – What configuration options are needed for a board? (Both set and validate)

- Examples of work done with the pipelines include:
  - Testing Reproducible Builds by using the exact same inputs to generate artifacts with identical checksums
  - Tracing the builds to monitor performance at both the process-level as well as inside the compiler itself
  - Keeping board configurations up-to-date across 12+ boards for a variety of projects
  - Building custom SDKs with additional open source headers and libraries for a variety of devices

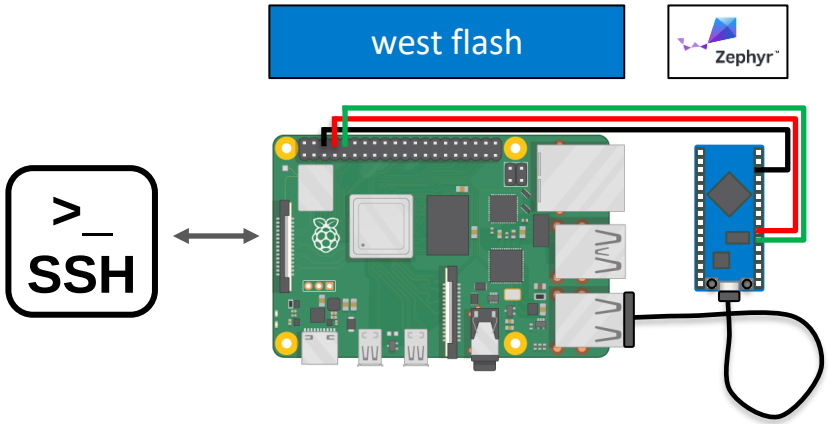
What is a Deployment Container?

- A container with the deployment tools necessary to flash Zephyr
- Useful for situations when your development workstation is separate from your embedded board:
  - Doing headless, automated testing in the lab
  - Running an unsupported desktop OS
  - Using a development machine without the software or hardware necessary to connect to the board (like a Chromebook or iPad)
  - Development in a Cloud IDE with a remote devices



Zephyr Deployment Example

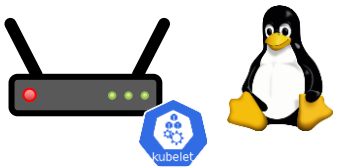
- Dhruv repurposed the devcontainers he created to deploy Zephyr to the MCU
- Successfully used it with VS Code, WSL2, and usbipd-win to flash the STM32L475 MCU
- Using a prebuilt Zephyr kernel image was also able to flash the MCU using the container on a Raspberry 4
- Trick involved adding ttyACM and usb_device to the cgroup-allowed devices list (--device-cgroup-rule)

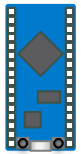


What is an Operations Container?

- A container with additional tools to facilitate operational tasks to fielded devices running Zephyr
- For example:
  - An organization may use Kubernetes (K8s) as its control plane for orchestration software
  - The Zephyr devices are sensors at the Far Edge that operate wirelessly over Bluetooth LE
  - Zephyr cannot join the Kubernetes cluster directly as it cannot talk directly to the other nodes over the overlay network

kubernetes

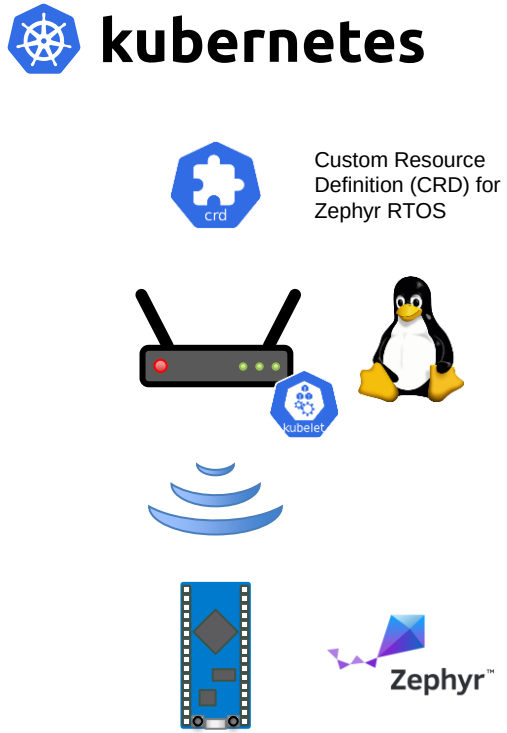




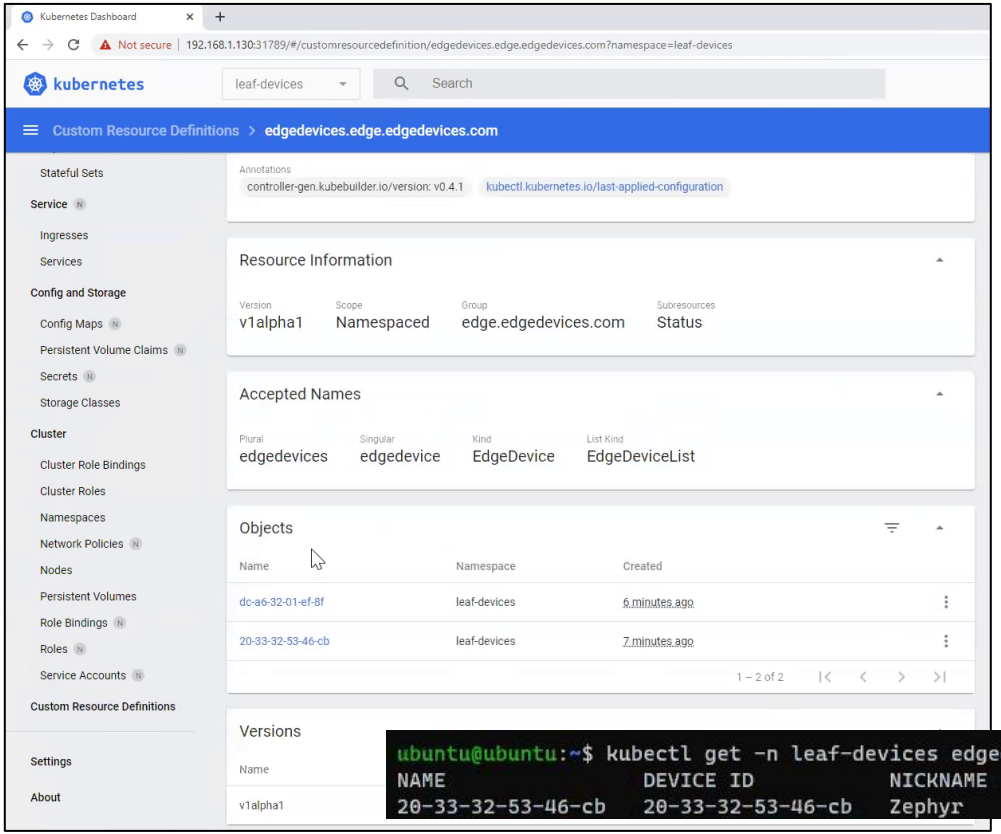
Zephyr™

Zephyr Operations Example

- A Linux gateway in between can speak to both networks and is running a kubelet to join the cluster as a node
- It is also running a Bluetooth LE server to detect BLE Clients
- Braydn Moore wrote a Custom Resource and Custom Controller for Zephyr
- Enabled the K8s control plane to see the Zephyr devices and proxy commands via the gateway
- Allows us to do OTA updates using MCUBoot to deploy new versions of our RTOS and application to the MCU



K8s Dashboard + Zephyr




```
ubuntu@ubuntu:~$ kubectl get -n leaf-devices edgedevices
NAME                DEVICE ID           NICKNAME   GATEWAY ID   IMAGE                                    VERSION
20-33-32-53-46-cb   20-33-32-53-46-cb   Zephyr     ubuntu       ghcr.io/bmoore2windriver/zephyrtest      alpha2
```

Summary

- Leveraging cloud technologies for development and operation shows some immediate and practical benefits
- Devcontainers make getting started as simple as pulling a container
- Provides a safe starting point for beginners using familiar technologies
- Cloud-Native principles encourage automation and sharing of knowledge between team members while reducing toil
- Getting acquainted with cloud helps us lean-in on applying CI/CD^2 and DevSecOps for the Far Edge

Next Steps

- Make some Zephyr devcontainers available from the community with new developers as the target audience
- Set a zero-to-60mph challenge to see if a new developer can run Zephyr on a board in less than 5 minutes
- Identify gotchas for new developers and reflect on what plugins, extensions, or scripts would help the workflow

References

- VS Code and Yocto Project, Yocto Project Summit 2020, Rob Woolley
- Developing Inside a Container, Microsoft VS Code Documentation
- ROS dev container for VSCode, Yosuke Matsusaka
- The Agile Embedded Podcast, Luca Ingianni and Jeff Gable
- Remote Zephyr development using Segger tunnel and a Raspberry Pi, Vojislav Milivojević, c/o Golioth
- Improving Zephyr Project Structure with Manifest Files, Asgeir Stavik Hustad, c/o Golioth
- GetPopper.io container-native task automation engine
- Nektos Act tool for running GitHub Actions locally including the Zephyr Project CI workflows


Questions?