arm

Modern Embedded Software Development with CI/CD and DevOps

Driving Efficiency and Quality in Firmware Projects

Christopher Seidl, Arm
May 21, 2025

Agenda

- What is CI/CD and DevOps?
- Why is it important for embedded developers?
- Using GitHub for DevOps
- Enhancing workflows with Keil Studio
- Hands-on and Q&A

What is CI/CD?

Continuous integration (CI)

- Automates building and testing of code changes
- Ensures code merges are safe and regression-free
- Improves team velocity with early feedback

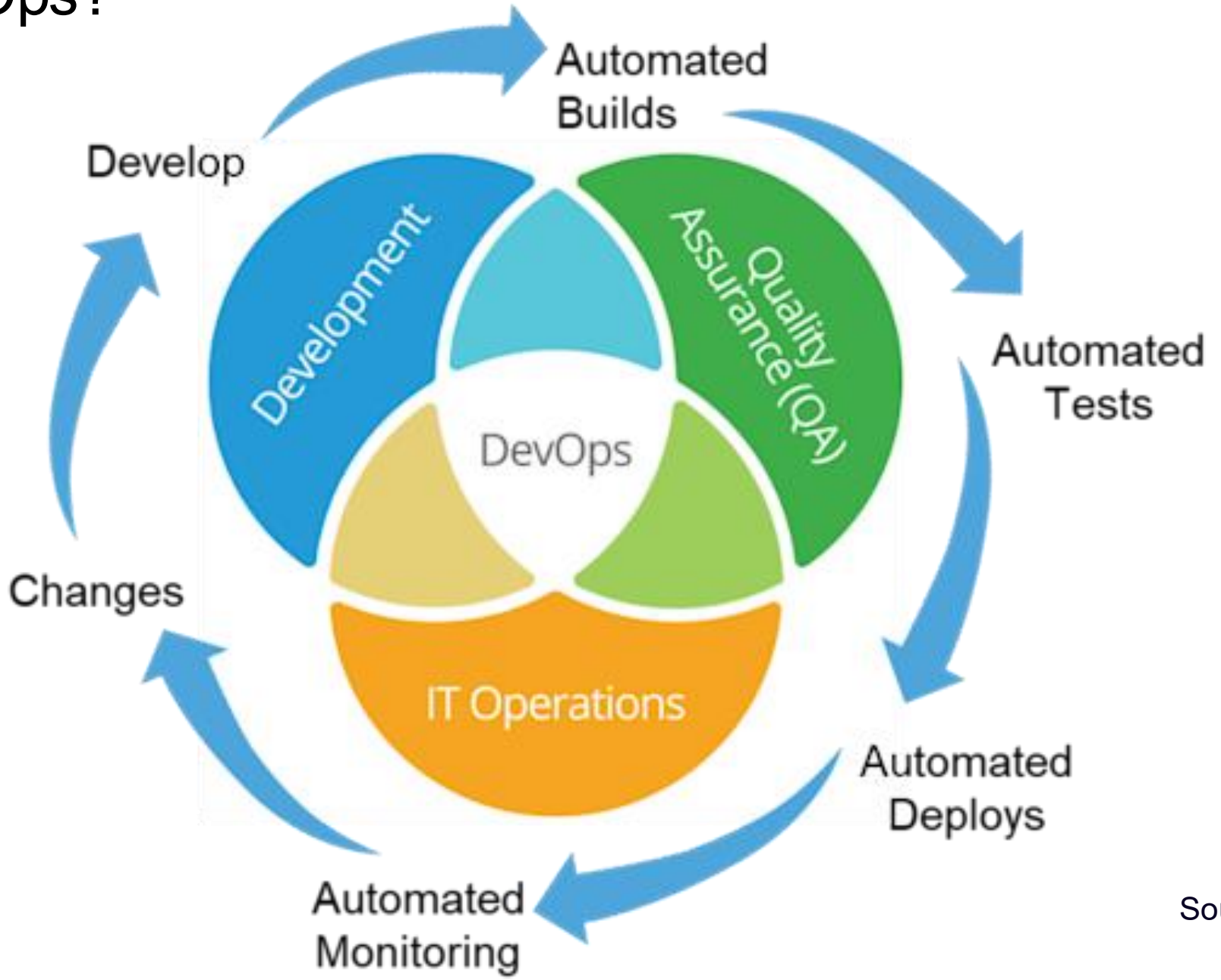
Key benefits

- Catch integration bugs early
- Enable collaborative workflows (e.g., PR validation)
- Build trust in the delivery pipeline

Continuous Delivery/Deployment (CD)

- Automates delivery of tested builds to release or staging
- Reduces manual handoffs and release risk
- Supports faster, more frequent firmware updates

What is DevOps?



Source: loves.cloud

Traditional vs. Modern Embedded Development

Aspect	Traditional	Modern (CI/CD + DevOps)
Build Process	Manual, local build on developer machine	Automated builds on every commit using CI pipelines
Toolchain Setup	Manual installation, version drift	Containerized or scripted toolchain setup (Docker, vcpkg)
Testing	Manual testing on physical hardware	Automated unit and integration tests, possibly in simulation or HIL
Code Integration	Infrequent, large merges (merge hell)	Frequent, small pull requests with automated checks
Collaboration	Siloed development	Shared repositories, PR reviews, integrated task tracking
Debugging	Local JTAG debug only	Remote debugging, cloud logs, CI failure reproduction
Deployment	Manual flashing	Automated flashing/testing, artifact release via GitHub Actions
Traceability	Spreadsheet logs, manual notes	GitHub Issues, pull requests, CI logs, release artifacts
Environment Reproducibility	Difficult (e.g., “works on my machine”)	Dev Containers or Remote-SSH environments in VS Code
Feedback Loops	Delayed, often post-integration	Fast, continuous feedback on every code change

Why is CI/CD important for Embedded?

Challenges

- Cross-compilation and toolchain setup is complex
- Hardware access may be limited or shared across teams
- Long and manual validation/test cycles
- Risk of integration failures late in development

Key benefits

- Faster detection of regressions
- Higher confidence before flashing to hardware
- Easier collaboration and code reviews
- Accelerated time-to-market and better quality

Solutions

- Automate builds and testing on every commit
- Use containers or dev environments for reproducibility
- Use virtual hardware (AVH) or use Hardware-in-the-Loop (HIL)
- Run tests in parallel and on schedule (nightly, per-PR)

DevOps Benefits for Embedded Teams

Consistent Build Environments

- Use containers to eliminate "works on my machine"
- Standardized toolchains across teams

Scalable Test Automation

- Automate unit, integration, and HIL tests
- Run CI pipelines on every branch, PR, or nightly build
- Reduce dependence on manual QA

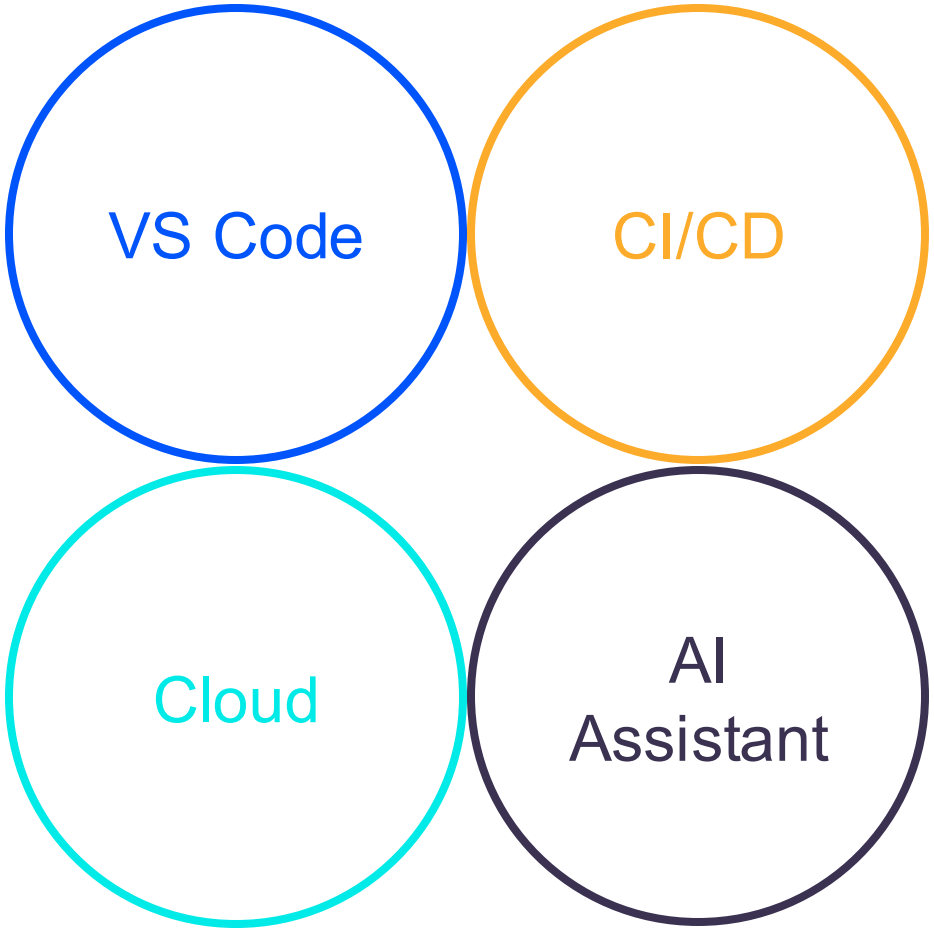
Artifact Management

- GitHub releases or internal package servers
- Traceable, versioned firmware builds
- Easy rollback and release automation

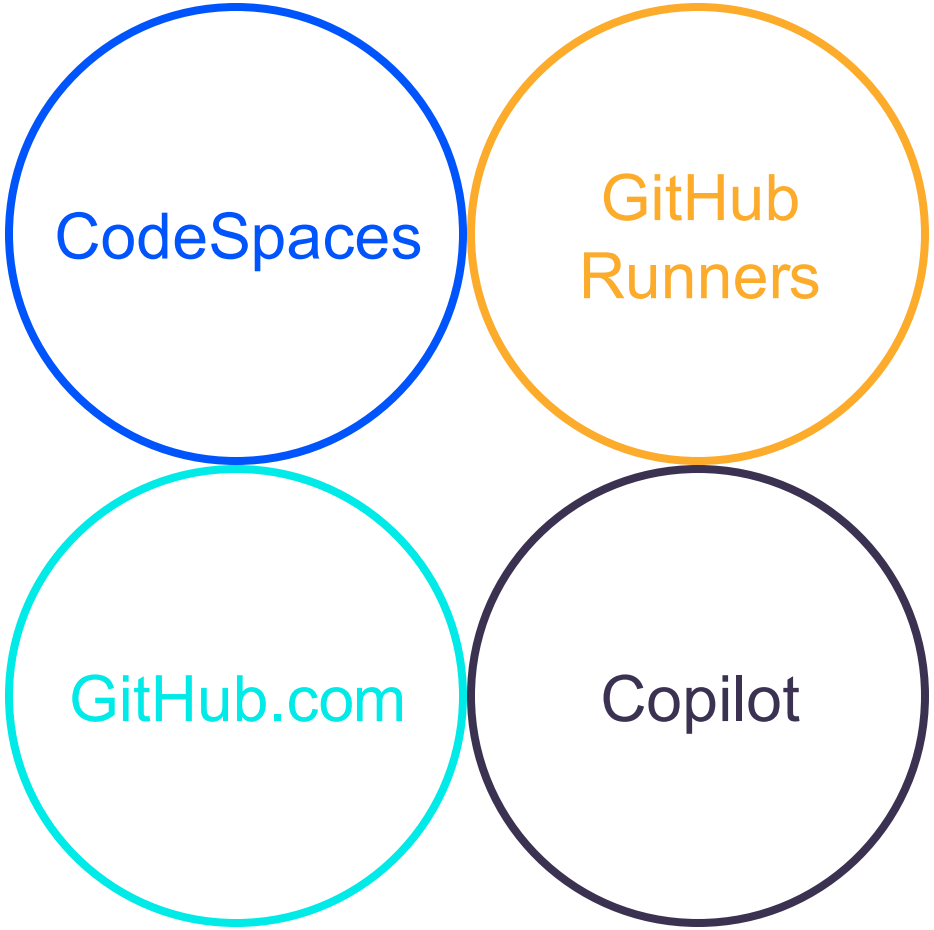
Improved Collaboration

- PR-based workflows encourage peer review
- GitHub issues + actions create clear audit trails
- Integrated status checks improve accountability

Modernized Workflows



GitHub Collaboration



GitHub as a DevOps Platform

GitHub Actions

- Build, test, and deploy on every commit or pull request
- Supports matrix builds, secrets, and reusable workflows
- Compatible with self-hosted runners for hardware access

Secrets & Environments

- Store API keys, certificates, credentials securely
- Define environments (e.g., staging, production) with deployment gates

Packages & Releases

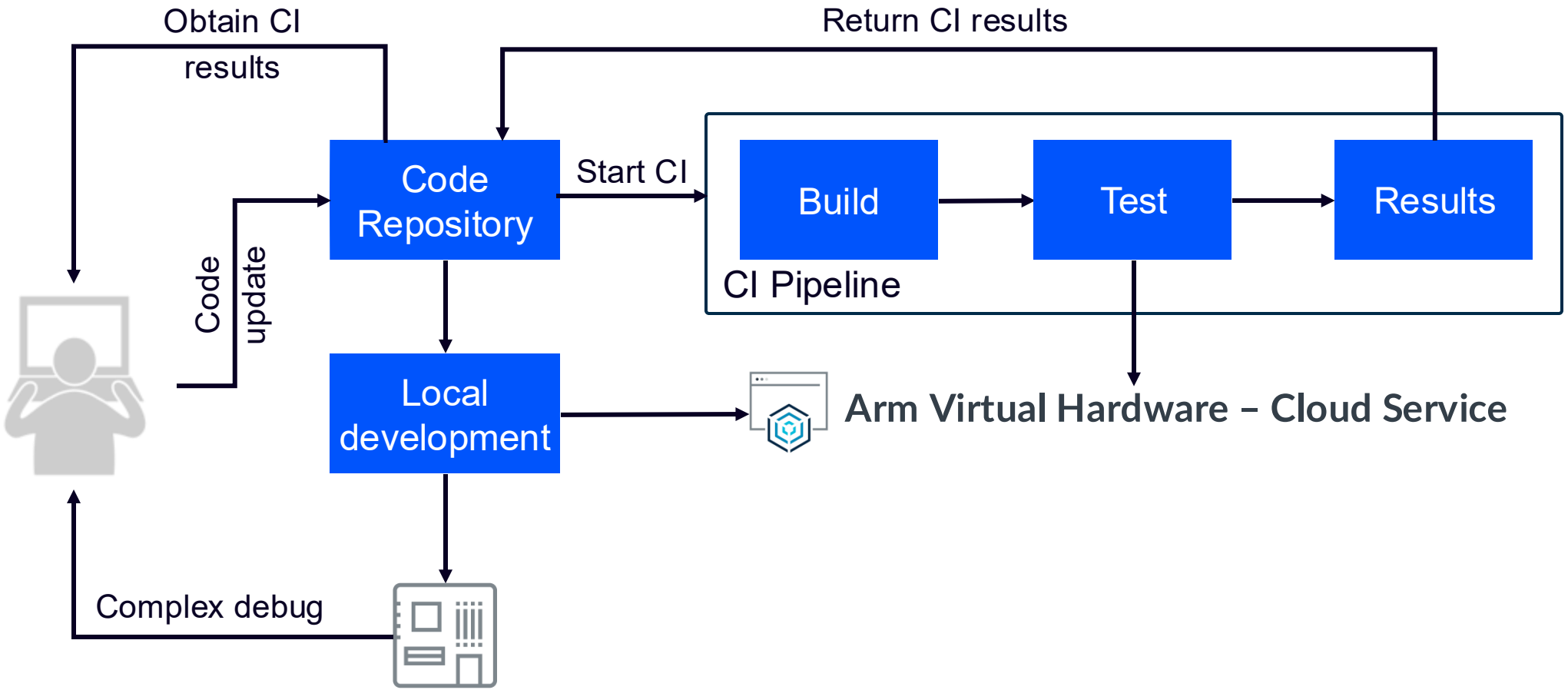
- Use GitHub Releases to publish versioned firmware binaries
- Integrate with GitHub Packages or third-party artifact repositories

Built-In CI/CD Observability

- Real-time logs, status badges, and actionable failure diagnostics

Modern DevOps development workflow

github.com/Open-CMSIS-Pack/vscode-get-started



GitHub Actions

A Platform called GitHub Actions

Source: dev.to

GitHub Actions Workflow Example

```yaml
name: Compile and Run
on:
  workflow_dispatch:
  pull_request:
    branches: [main]
  push:
    branches: [main]

jobs:
  CI_test_run:
    runs-on: ubuntu-22.04

    steps:
      - name: Checkout
        uses: actions/checkout@v4

      - name: Activate vcpkg
        uses: ARM-software/cmsis-actions/vcpkg@v1

      - name: Activate Arm tool license
        uses: ARM-software/cmsis-actions/armlm@v1

      - name: Build
        run: |
          echo "Building get started example ..."
          cbuild get_started.csolution.yml --packs --update-rte --context .debug+avh

      - name: Execute
        run: |
          echo "Running get started example ..."
          FVP_MPS2_Cortex-M3 --simlimit 10 -f Project/fvp_config.txt -a out/Project/avh/debug/Project.axf | tee Project.avh.log
          echo "Checking output..."
          test "$(grep "FAIL: " Project.avh.log | wc -l)" -eq 0
```


Hands-On

Key Takeaways

Agility	• Faster iterations, early Feedback, continuous improvement
Reduced Risk and better Quality	• Catch issues earlier with automated builds and tests
Readily available Tooling	• GitHub + VS Code provide an end-to-end workflow
Embedded ≠ Manual Anymore	• Automation, containers, and reproducible environments are now mainstream

Resources & Further Reading

- Webinar: CLI builds using CMSIS-Toolbox

- CMSIS-Toolbox documentation: https://github.com/Open-CMSIS-Pack/cmsis-toolbox/blob/main/docs/README.md

- Template repository: https://github.com/Arm-Examples/AVH_CI_Template
  - Use it as a starting point
  - Contains a simple test project and required GitHub workflows

- MLOps template: https://github.com/ARM-software/AVH-MLOps
  - A set of foundation tools and software components to enable MLOps systems and the overall development flow for machine learning applications

Q&A

- Thank you! Questions?

arm

Merci
Danke
Gracias
Grazie
谢谢
ありがとう
Asante
Thank You
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה
ధన్యవాదములు
Köszönöm