Containers, Continuous Integration, and Cyber-Physical Systems

A Modern Development Paradigm

Claudius Stern

FOM University of Applied Sciences
for Economics and Management

Prof. Dr. Claudius Stern

Short resume

Station (1999 – 2007):

Studies of Computer Science at the University of Paderborn

Station (2007 – 2014):

Doctorate at the University of Paderborn
in the field of embedded systems

Research projects:
Approaches to location-based mobile services
Approaches to the digitization of primary care

Station (2007 – today):

Own company for the development of embedded systems

Station (2013 – 2018):

Head of embedded systems development , biozoom services GmbH, Kassel, Germany

Station (2018 – today):

Professor at FOM University of Applied Sciences, Kassel, Germany

Research Area: Embedded Systems

Agenda

History
- Once upon a time

Nowadays
- Traditional development challenges
- Trends

Idea
- DevOps for Embedded Systems
- Consistent toolchains

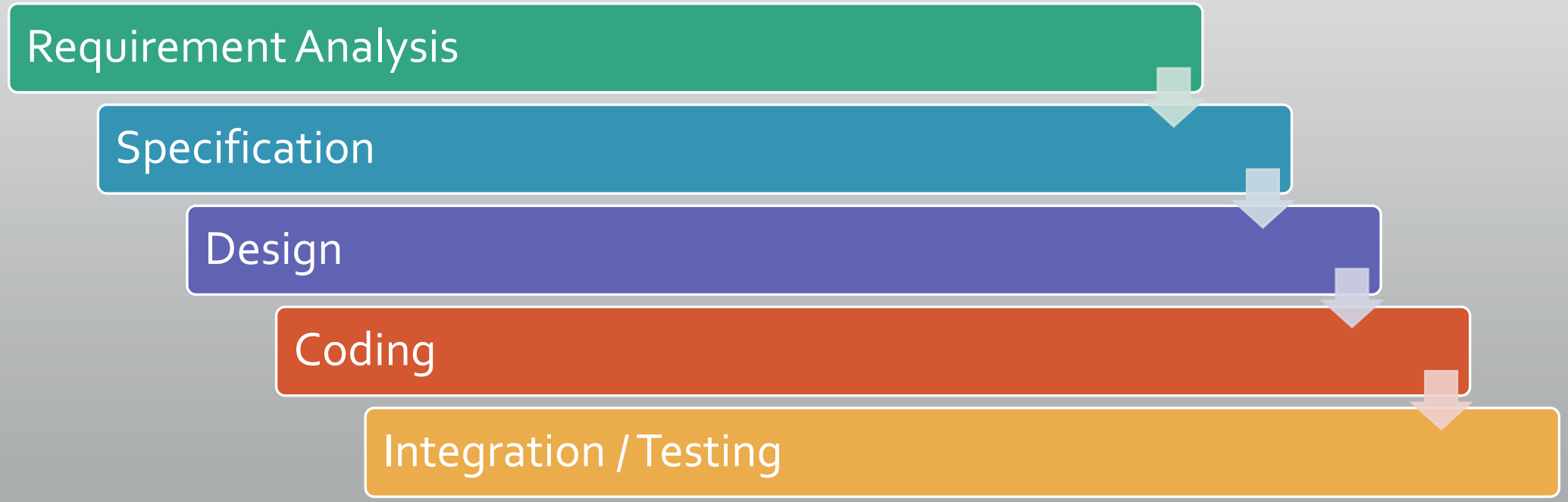
How
- Technical roadmap (what is needed?)
- Concepts (interaction of tools)

Implement
- Real world examples

Once upon a time…

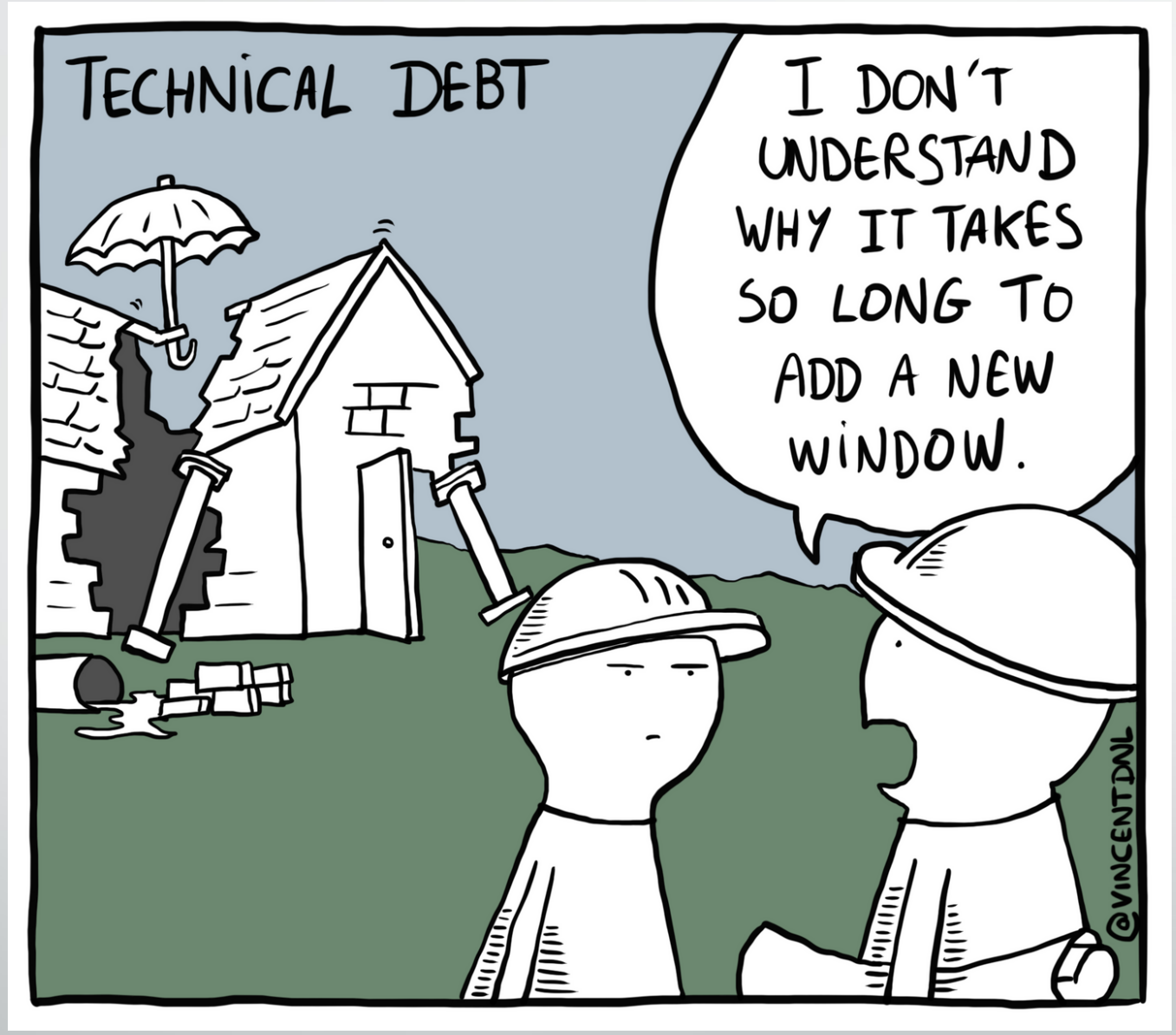
A short recap of last talk about DevOps.

Traditional Development



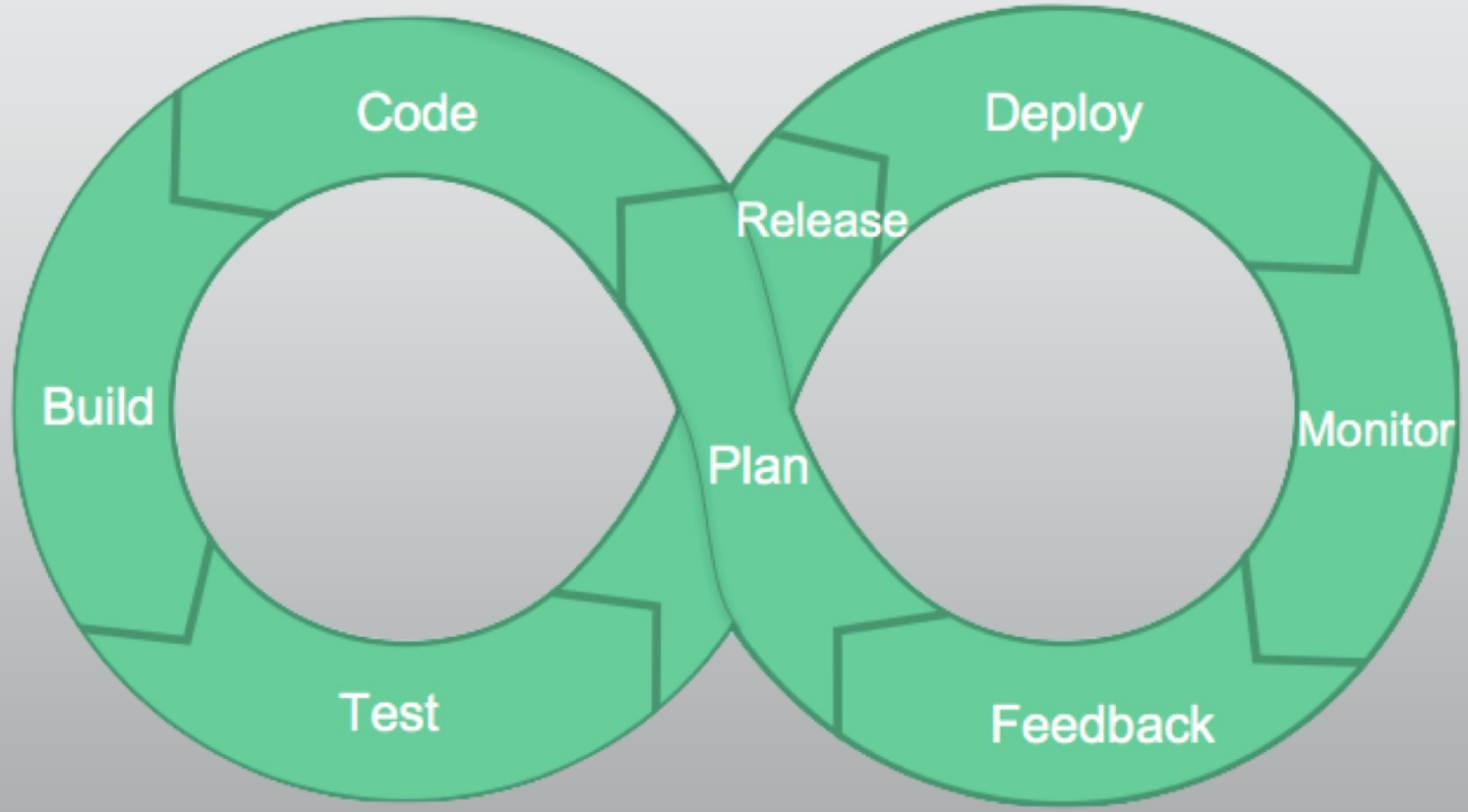
Challenges

- Splitted Teams
  - Communication problems
  - Manual and double testing
  - The teams expertise is not shared across department borders
- Every Team works on their own
  - No sense of unity



Source: https://vincentdnl.com/drawings/

DevOps Lifecycle



Source: https://ovh.github.io/tat/imgs/devops-lifecycle.png

DevOps for Embedded System

- DevOps can be used for embedded systems development
- First use case: automatic test of build-ability
  - Librarys availability, compatibility

Start simple

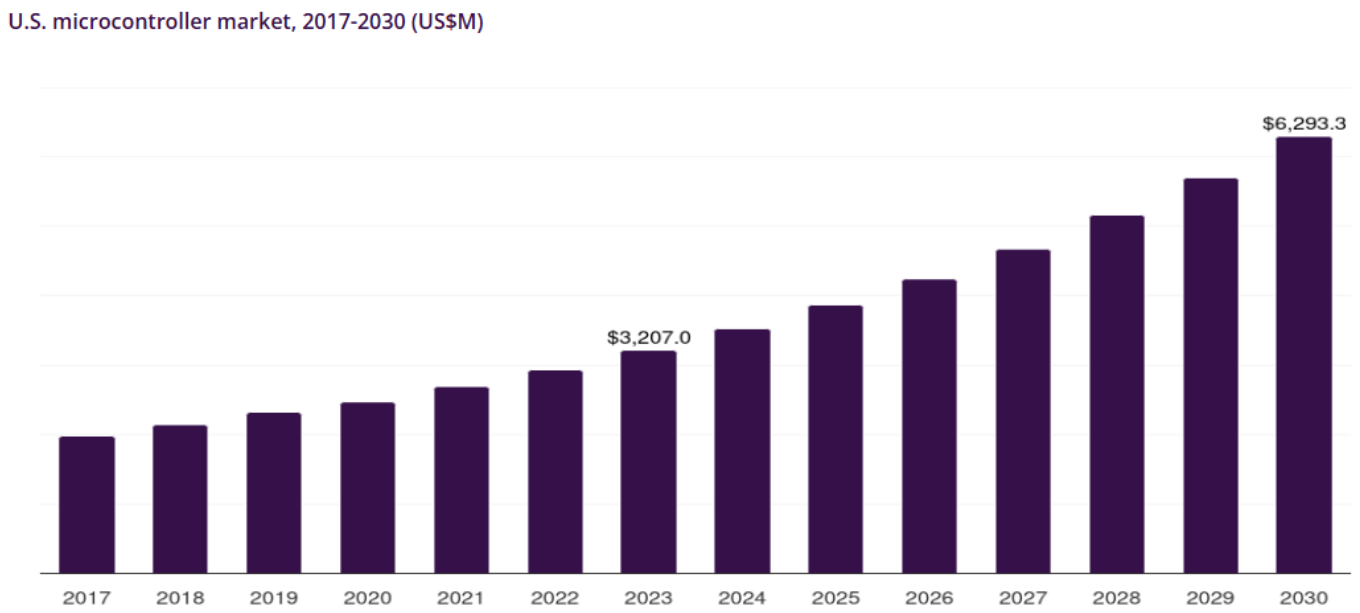

```yaml
image: python:3.7

stages:
    - test
cache:
    paths:
        - "~/.platformio"

before_script:
    - pip install -U configparser
    - pip install -U platformio
    - platformio update

job:
    stage: test
    script:
        - platformio run -e test
    tags:
        - python27
```

- Even for embedded systems development!
- Start simple, grow more complex over time
- Actually saved my day…
  - Caught error in a third-party library

Nowadays…

Microcontrollers are broadly used

- Large market
- Many different industries



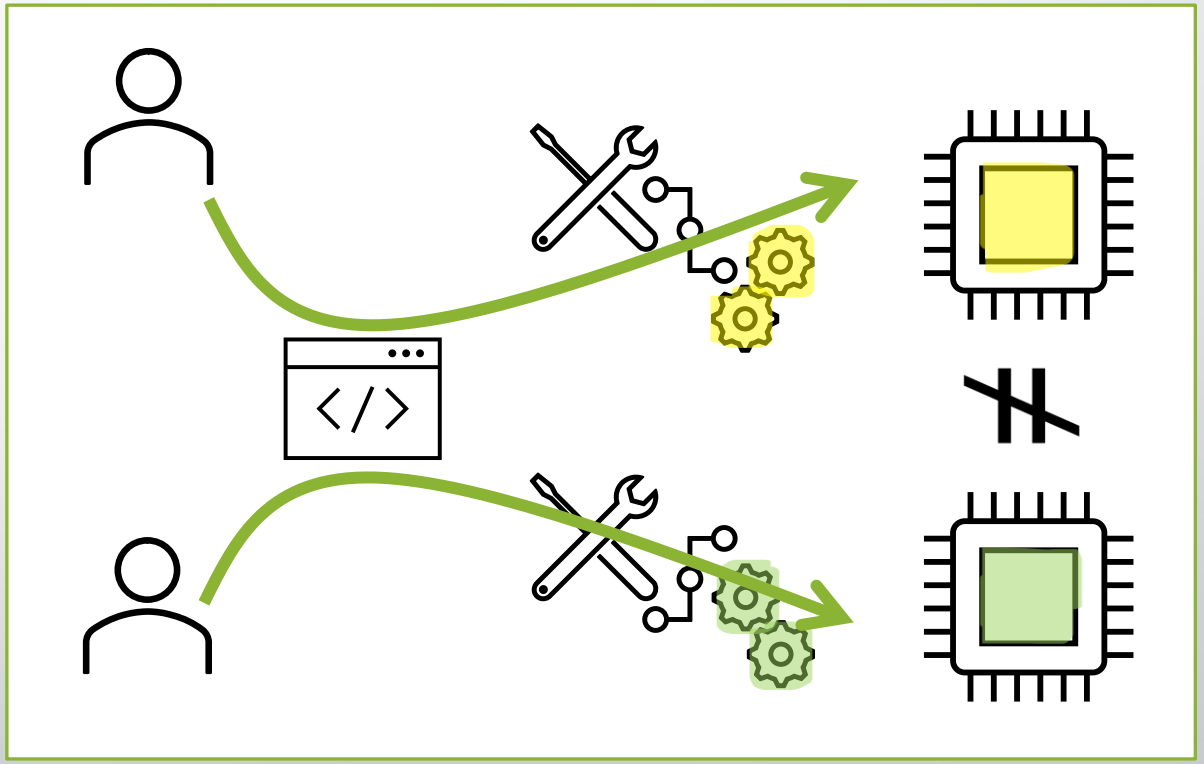
Source: https://www.grandviewresearch.com/horizon/outlook/microcontroller-market/united-states

Embedded Systems Development

- Resource constraints
  - Processing power, memory limitations
- Real-Time performance
  - Timing constraints
  - Deterministic behavior
- Hardware-Software-Co-Design
  - Debugging
- Reliability
  - Robust error handling

Toolchain inconsistencies → firmware inconsistencies

- Same source code
- Different toolchains
  - even only different settings



Idea…

What to improve?

- Ease-of-use
- Reproducibility
- Product quality
- Cycle time
- Repetitive tasks

Approaches

- Ease-of-use
- Reproducibility
- Product quality
- Cycle time
- Repetitive tasks

Automatic checks

Common UI

Fixed build system

Everyone can build

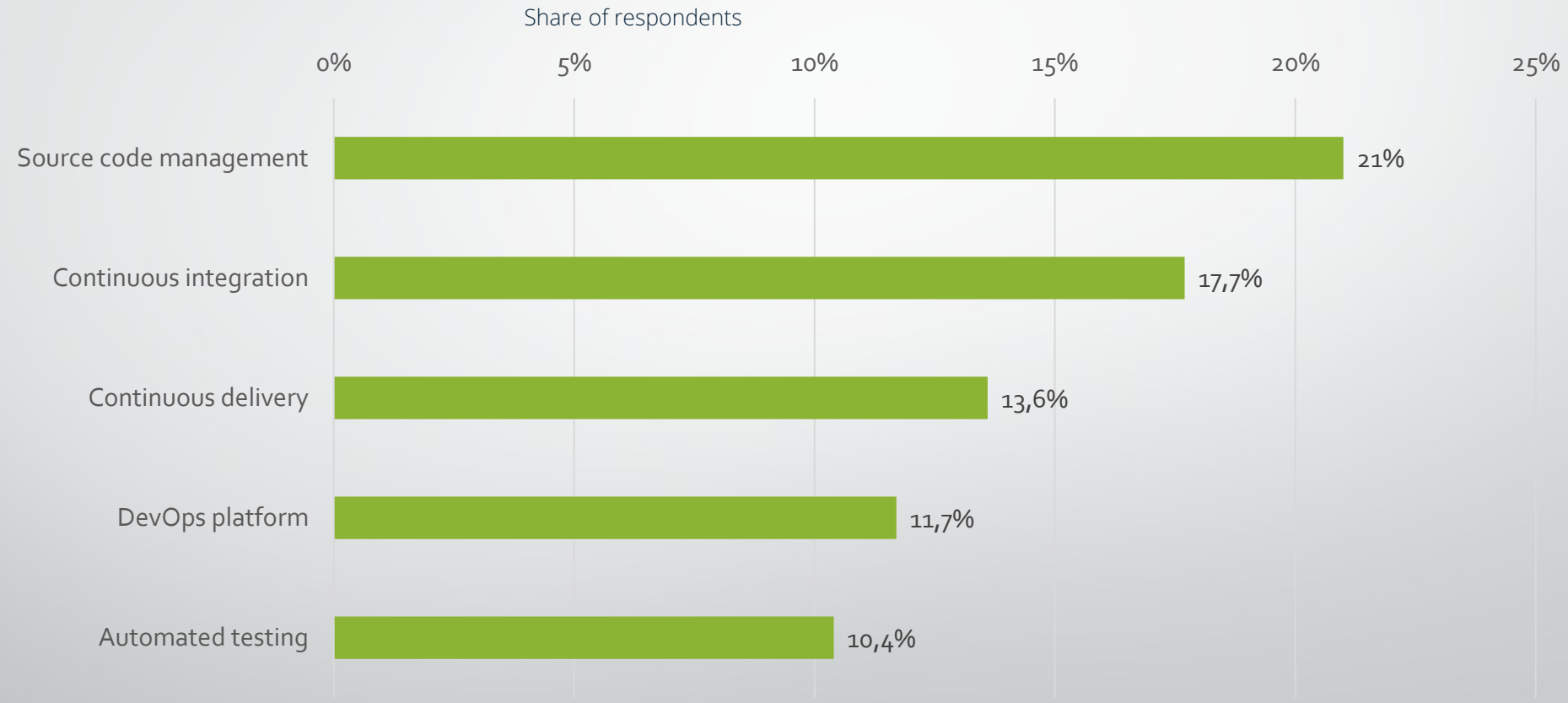
Fully automatic build

Build system for embedded systems

- Complex toolchain
  - Specific compiler for each system
  - Dozens of specific settings – even per project

Changes made to software development process in DevOps teams in organizations worldwide in 2021

DevOps teams software development process changes in organizations worldwide 2021



Note(s): Worldwide; February to March, 2021; 4,294 respondents; developers, operations, and security professionals*
Further information regarding this statistic can be found on page 8.
Source(s): GitLab; ID 1234098

Ease-of-use

Reproducibility

Product quality

Cycle time

Repetitive tasks

Benefits

- Same central instance to use
- Consistent toolchain usage
- Improved software quality due to automatic testing
- No need to install software on local environments
- Fully automated test and build process

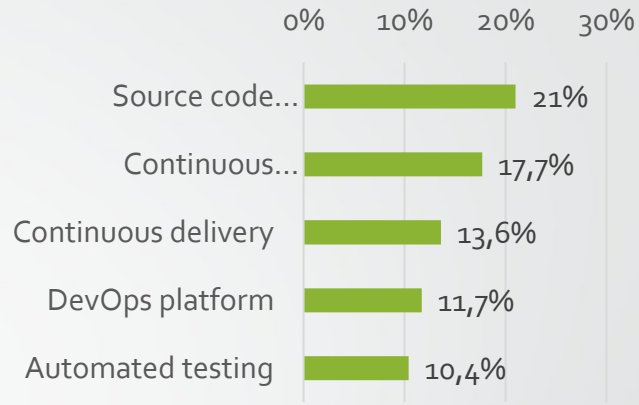
Everyone can build a modified firmware version!

How to…

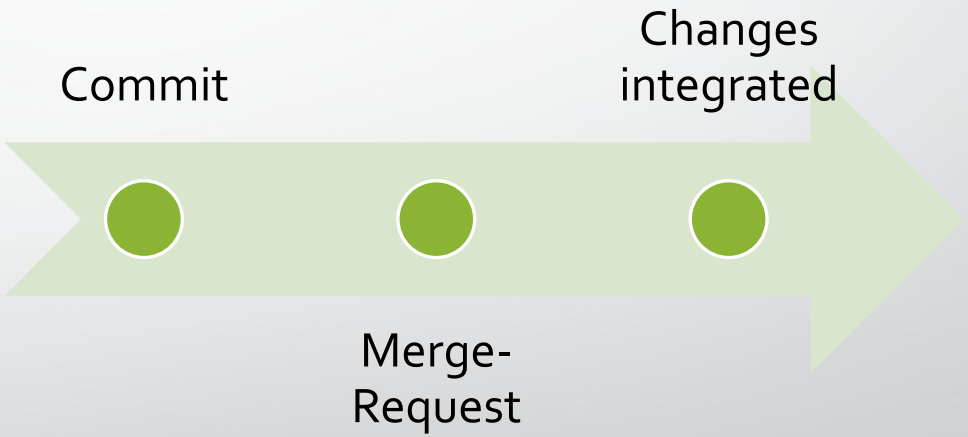
Technologies

- Exploit DevOps technologies
  - Source Code Management
  - Infrastructure-as-Code (IaC)
  - Software Container / Container Registry
  - CI/CD platform
  - Container Management

Versioning



- Essential to start
  - Creates changelog from commit comments
  - Grants version control
    - Source Code
    - Software Container
  - Enables distributed and decentralized working



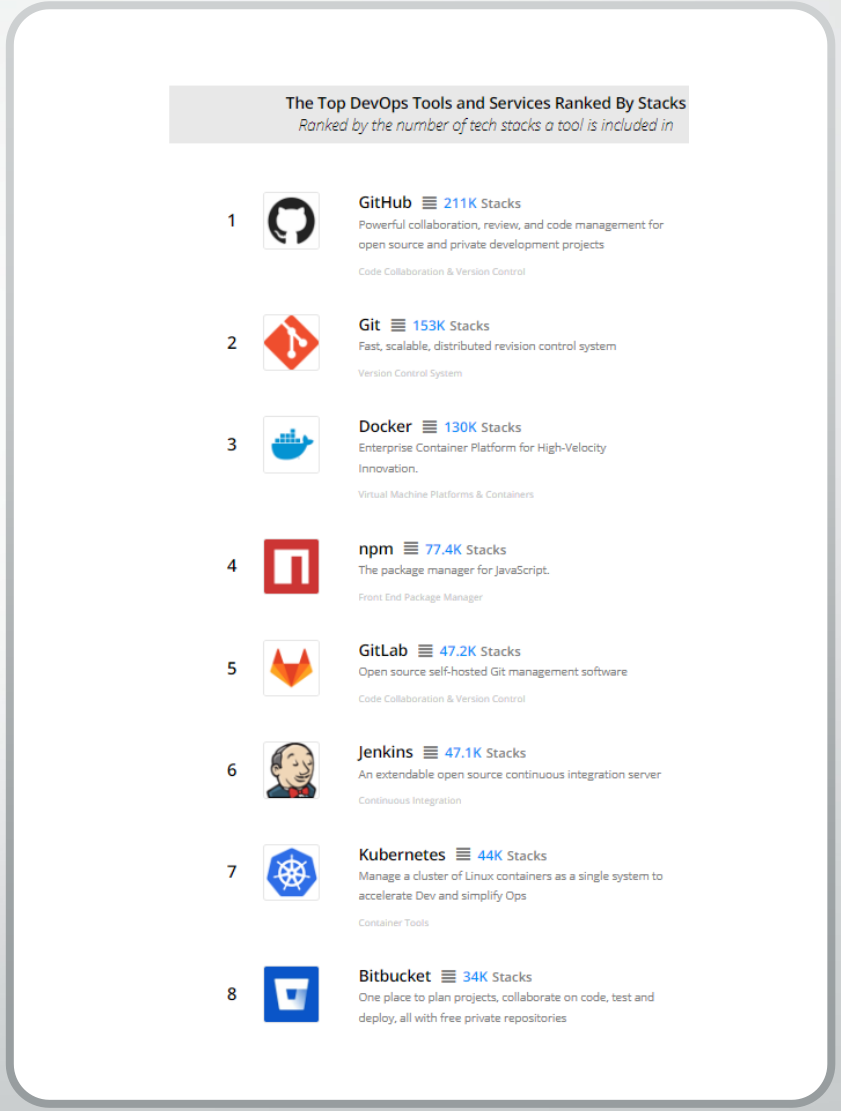
How to start implementing an DevOps environment?

- There are thousand ways and tools to build a DevOps infrastructure
  - A general blueprint does not exist
- Where to start?
  - For embedded systems: virtualization of the toolchain is a good starting point

Which tools are available? (2022)

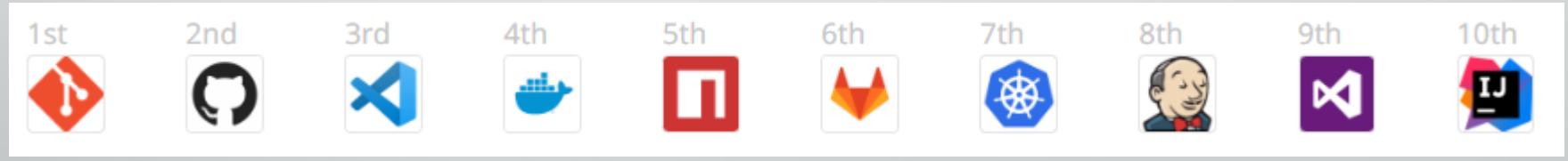
- Stackshare offers a good overview
  - e.g. DevOps Index
  - Shows how often Software is used in a stack
    - Indicates realistic popularity

Source: https://stackshare.io/index/devops - retrieved on 2022-05-06



Which tools are available? (2025)

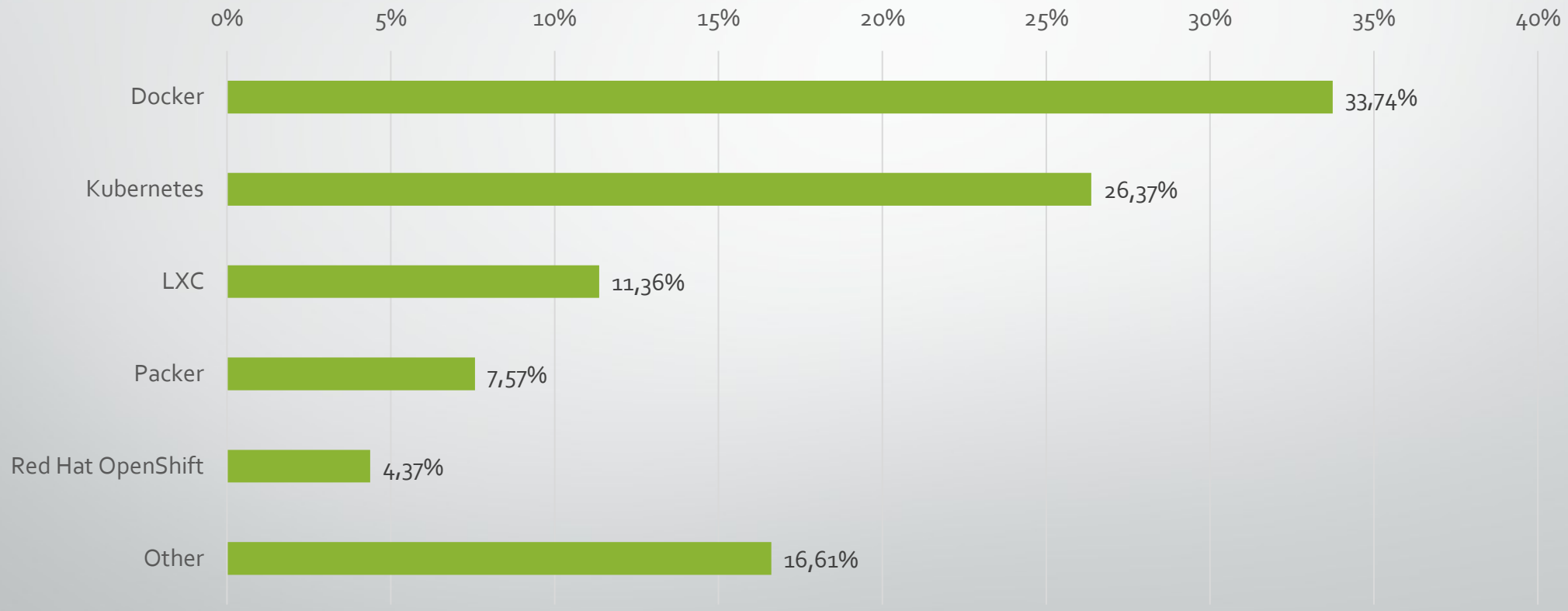
1. Git
2. GitHub
3. Visual Studio Code
4. Docker
5. npm
6. GitLab
7. Kubernetes
8. Jenkins
9. Visual Studio
10. IntelliJ IDEA



Source: https://stackshare.io/devops - retrieved on 2025-03-08

Leading containerization technologies

Market share of leading containerization technologies worldwide 2024



Note(s): Worldwide; 2024
Further information regarding this statistic can be found on page 8.
Source(s): Datanyze; ID 1256245

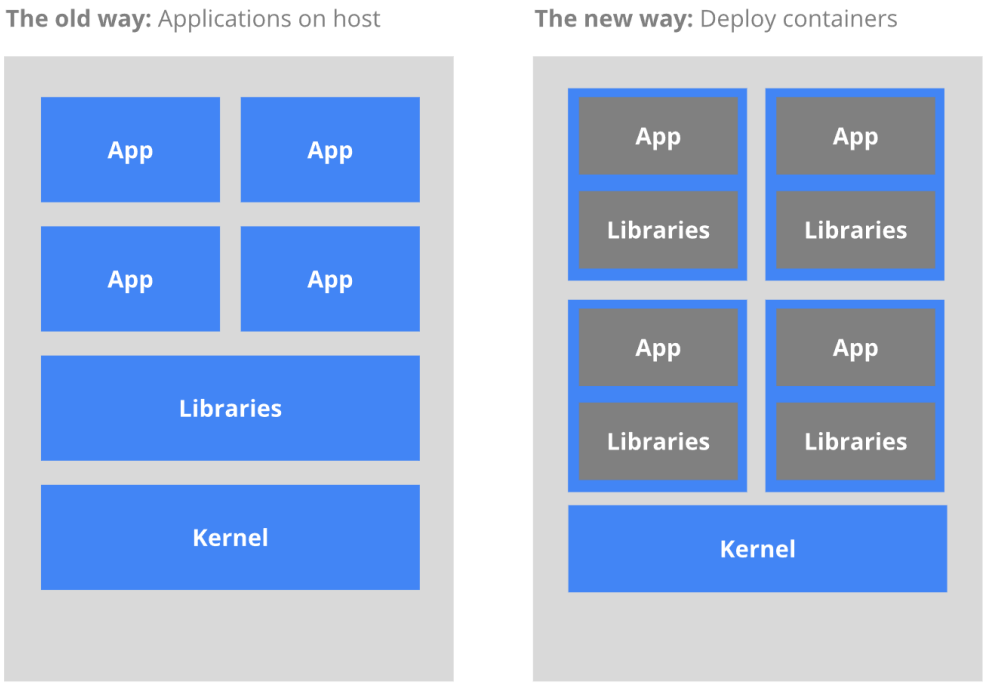
Proposed Architecture

Implement…

Software Container

- Portable
- Lightweight
- Fast to use

- Develop and build your software for software containers



Source: https://kubernetes.io/de/docs/concepts/overview/what-is-kubernetes/

Toolchain first!

- Create the toolchain container
  - Compiler
  - Settings
  - Necessary libraries
  - Use local Docker environment for testing

Build the toolchain container

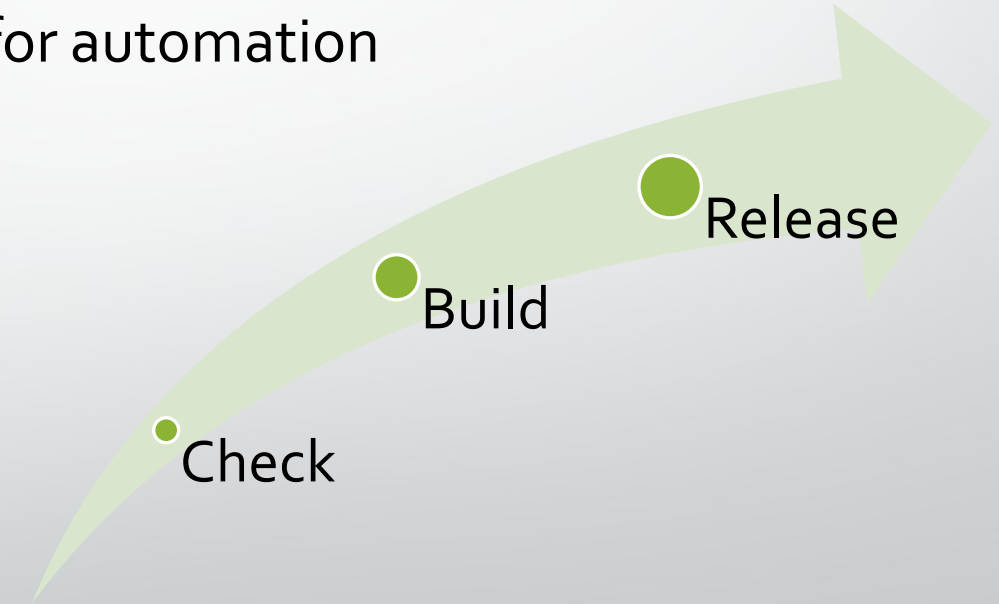
- Start with a base image you know, e.g., Ubuntu
- Install tools to get and prepare the actual toolchain
  - git, wget, python, build-essential, cmake
  - Good practice: clean up as far as possible to keep the container small
- Download and install the toolchain, e.g., arm-gcc
- Download and install necessary libraries, e.g., a board support package

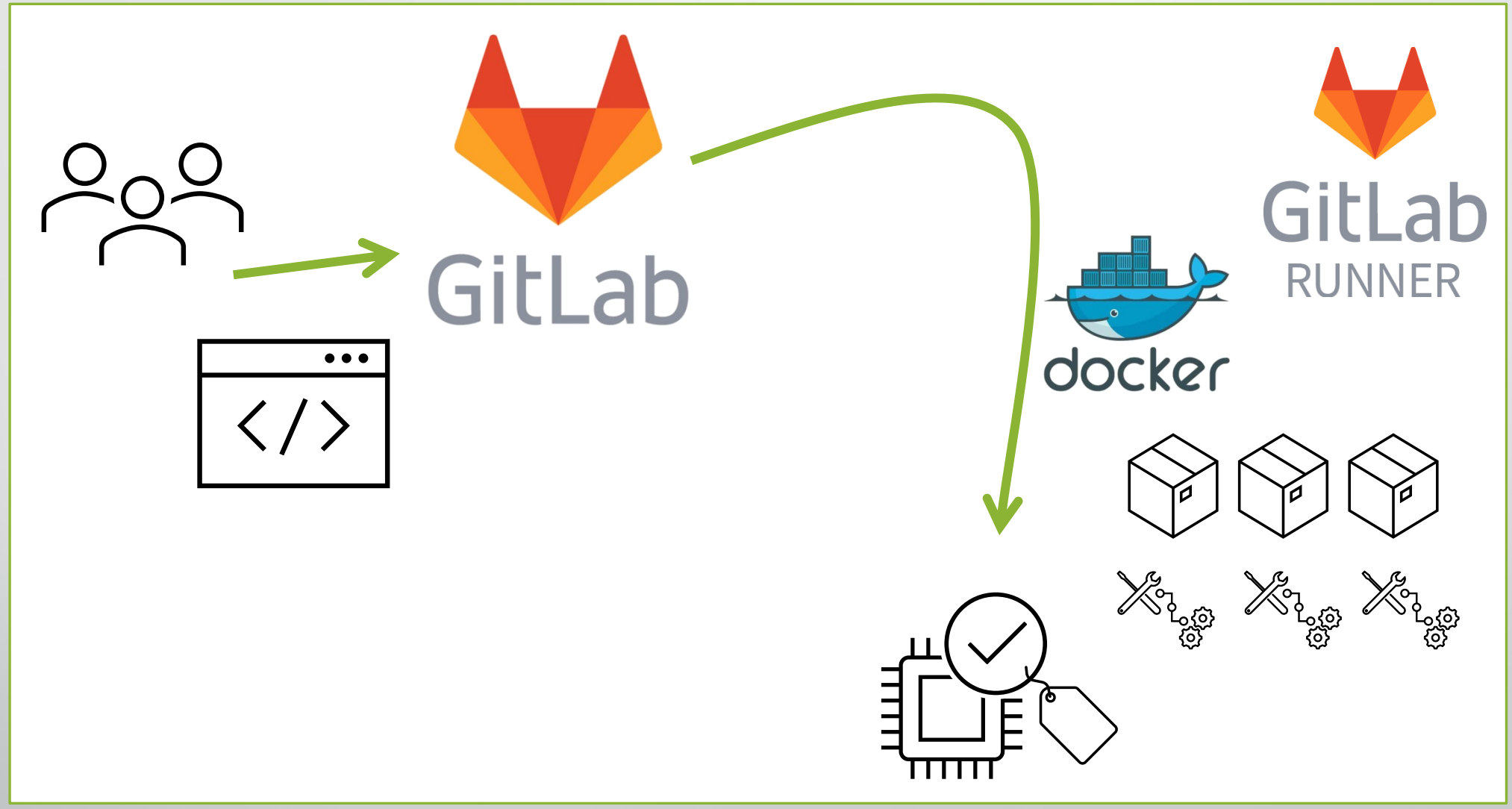
Toolchain Dockerfile

```
FROM ubuntu:22.04

# Set environment variables
ENV BSP_PATH=/opt/M031BSP-3.07.000
ENV BSP_PARENT_PATH=/opt
ENV ARM_GCC_PATH=/opt/arm-gnu-toolchain
ENV PATH="$ARM_GCC_PATH/bin:$PATH"

# Install necessary dependencies
RUN apt-get update && apt-get install -y wget unzip build-essential cmake git python3 python3-
pip && apt-get clean && rm -rf /var/lib/apt/lists/*

# Install ARM GCC toolchain
RUN wget https://developer.arm.com/-/media/Files/downloads/gnu/13.3.rel1/binrel/arm-gnu-
toolchain-13.3.rel1-x86_64-arm-none-eabi.tar.xz -O /tmp/arm-gcc.tar.xz \
    && mkdir -p $ARM_GCC_PATH \
    && tar -xvf /tmp/arm-gcc.tar.xz -C $ARM_GCC_PATH --strip-components=1 \
    && rm /tmp/arm-gcc.tar.xz

# Install Nuvoton Board Support Package
RUN wget https://github.com/OpenNuvoton/M031BSP/archive/refs/tags/V3.07.000.zip -O
/tmp/m031_bsp.zip \
    && mkdir -p $BSP_PATH \
    && unzip /tmp/m031_bsp.zip -d $BSP_PARENT_PATH \
    && rm /tmp/m031_bsp.zip

# Set working directory
WORKDIR /workspace
```

CI/CD platform

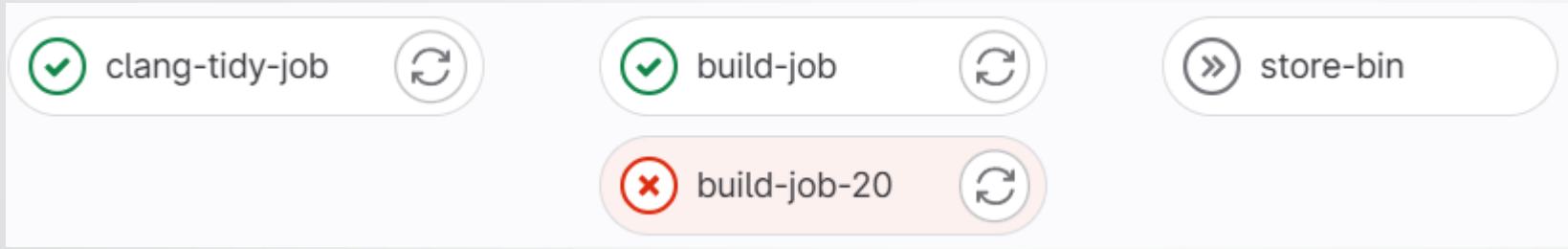
- Uses code/artifacts from versioning for automation
  - Check the firmware code
  - Build the firmware artifact
  - Release the artifact

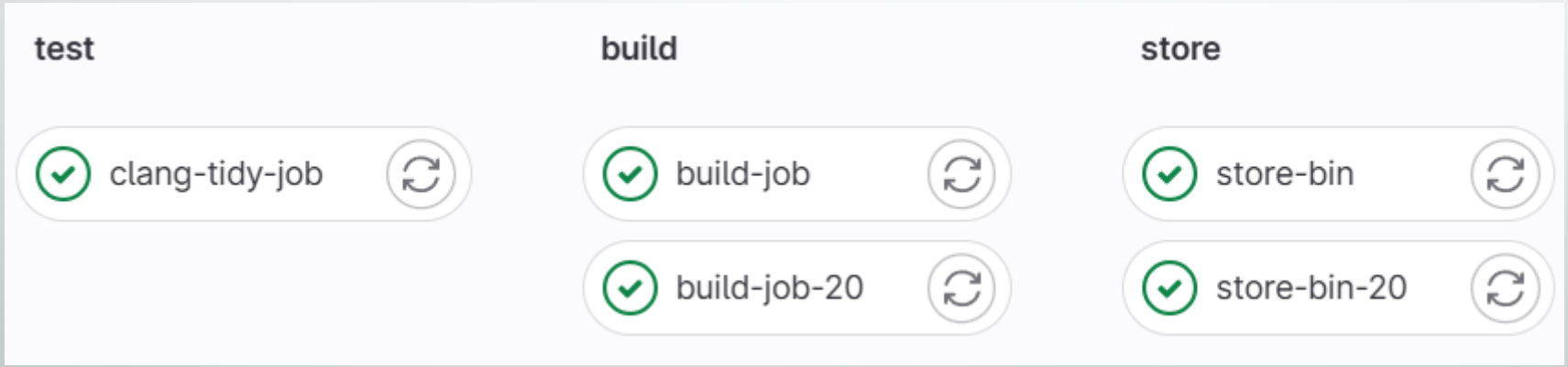



GitLab
GitLab
RUNNER
docker

Real-World Examples


```
65  src/main.cpp:263: [medium:warning] 200 is a magic number; consider replacing it with a named constant
    agic-numbers]
66  ========================= [FAILED] Took 5.37 seconds =========================
67  Component     HIGH    MEDIUM    LOW
68  -----------  ------  --------  -----
69  src            0        28       8
70  Total          0        28       8
71  Environment    Tool       Status    Duration
72  ------------  ---------  --------  ------------
73  scanner        cppcheck   PASSED    00:00:01.020
74  scanner        clangtidy  FAILED    00:00:05.368
75  ==================== 1 failed, 1 succeeded in 00:00:06.388 ====================
77  ERROR: Job failed: exit code 1
```

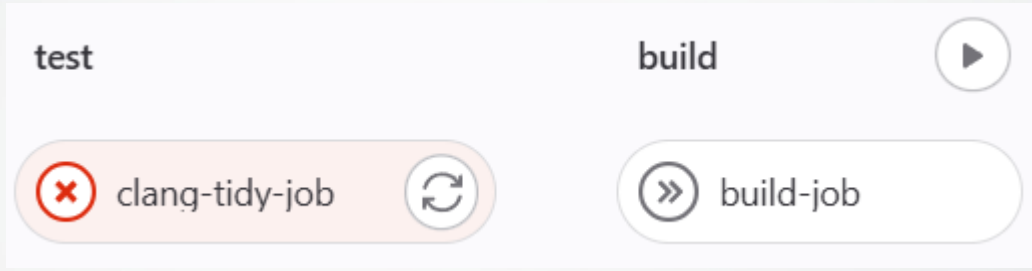

clang-tidy-job
build-job
store-bin
build-job-20

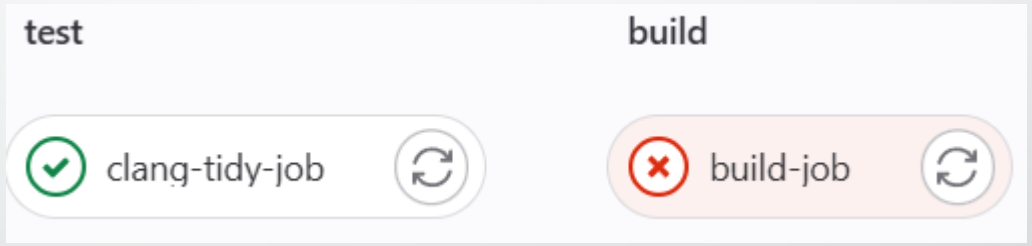

test
build
store
clang-tidy-job
build-job
store-bin
build-job-20
store-bin-20

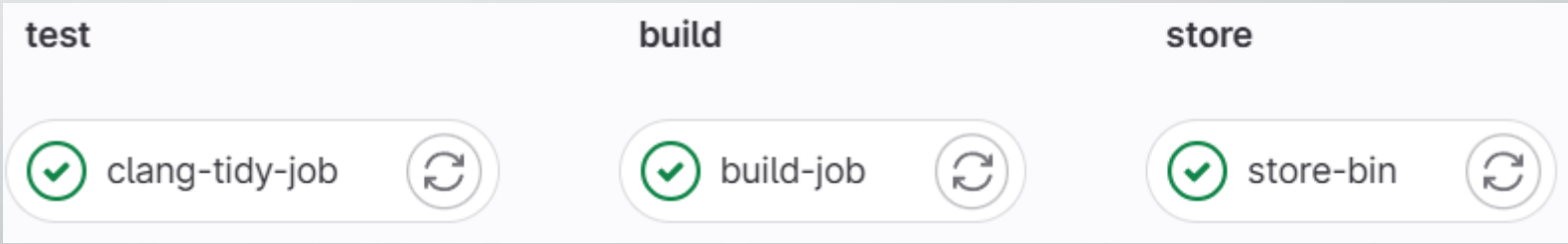
Real Firmware Example

```yaml
stages:
  - test
  - build
  - store

clang-tidy-job:
  stage: test
  image: …/platformio:latest
  script:
    - pio check --fail-on-defect=medium

build-job:
  stage: build
  variables:
    …
    WIFI_PASS: $WIFI_PASS
  image: …/platformio:latest
  script:
    - pio run
  after_script:
    - … # Copy artifact to root
  artifacts:
    paths:
      - $FILE_NAME-*.$FILE_TYPE

store-bin:
  stage: store
  variables:
    … # Set paths
  image: …/platformio:latest
  dependencies:
    - build-job
  script:
    - … # Copy artifact to server
  rules: # Run only on tagged versions
    - if: $CI_COMMIT_TAG =~ /^v\d+\.\d+\.\d+$/
```

GitLab







Technical Things to Remember

- Docker
  - Containerization technology
- Toolchain
  - Compiler, Tools, Libraries
- CI
  - „Continuous integration“
  - code, build, test
- Pipeline
  - Connects technologies for automation
  - Only successfully run jobs reach next stage

Lessons Learned

- An independently working toolchain is great!
- Using the DevOps pipeline to build artifacts.
- Changes can be made without the need to install and configure the toolchain.
- Always consistent state of artifact creation!