robotizer
BY STOGL ROBOTICS

Deployment von ROS 2 Anwendungen für 24/7 Einsatz

DR. DENIS

@

ROSCON DE

2024

Warum?

- Weil wir Robotik machen wollen!
  - Effizient die Lösungen entwickeln
  - Fehlerfreie und Stabile Ergebnisse haben
  - Einfacher mit Kollegen zusammenarbeiten
  - Spaß haben

$whoami

- ROS user seit C-Turtle
- ROS 2 seit 2020
- ros2_control maintainer
- **Promotion am KIT**

Was tun wir?

Strategische Beratung und Konzeptentwicklung (“ConOps”)

- Automatisierungsmöglichkeiten in Sondermaschinenbau für langfristige Eigenständigkeit
- Spezifische ROS und ROS 2 Beratung und Entwicklung - Nutzung des neusten Stand-der-Technik
- Expertise in Echtzeitregelung und Steuerung von Maschinen und Roboter

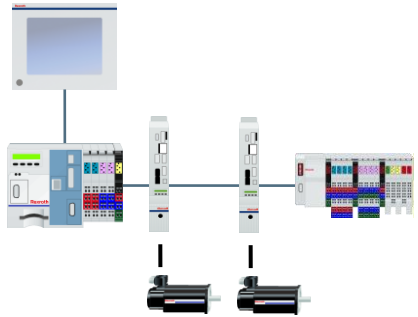
Entwicklungsdienstleistungen mit einer zukunftssicheren Basis

- Entwicklung von Lösungen mit open-source Kern für langfristige Unabhängigkeit
- Unterstützung von internen Entwicklungsteams mit Expertenwissen und Erfahrung
- Optimierung von existierenden Lösungen für mehr Flexibilität und Performance
- Begleitung bei der Produktisierung von Prototypen

ROS

b»robotized BY STOGL ROBOTICS

Industrial PLC System



State of the Art Algorithms (E.g. Perception, Motion Planning)

Physics Simulation

Visualization of Complex Data

Rich Hardware Driver Ecosystem

Machine Learning Integration

Complex Data Logging

Simple Inter-Process Communication on Complex Data

EtherCAT Fieldbus Connectivity

Scalable Hardware Portfolio

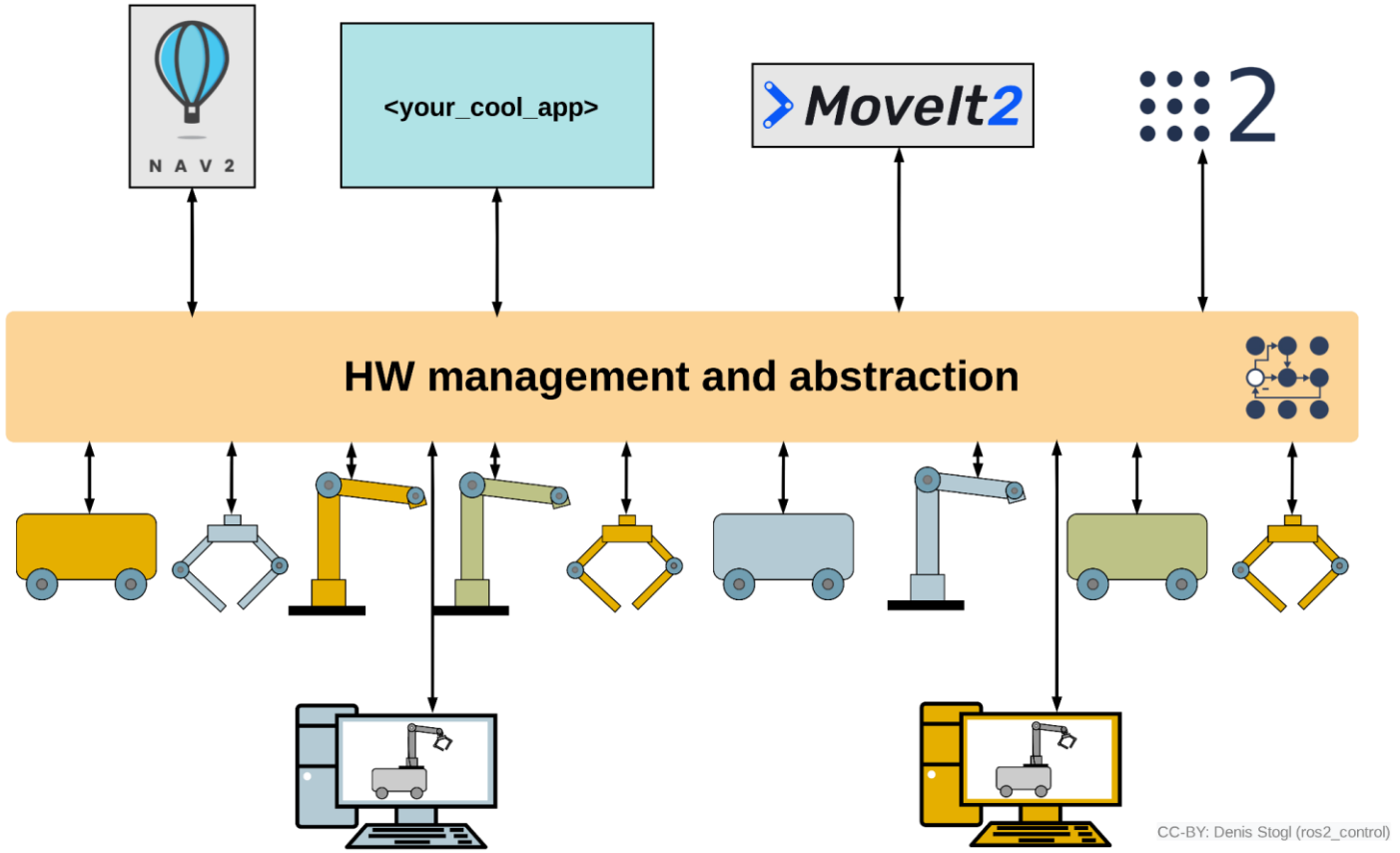
Industrial Grade Long Term Availability

PLC Programming

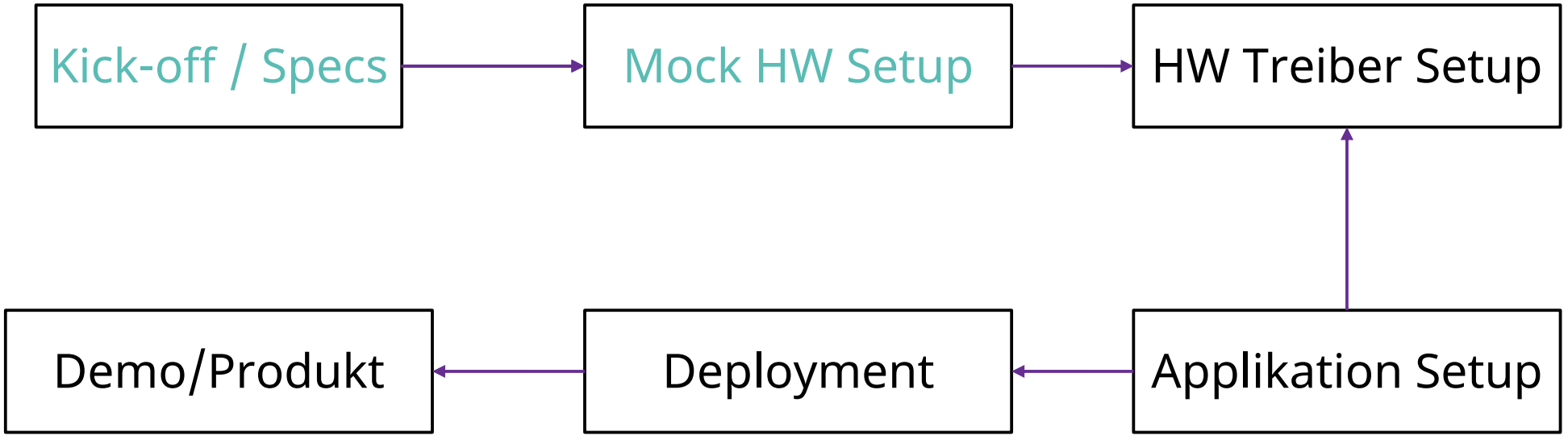
Certified Safety (IEC 61508)

Sensors, Actors, Input/Output Devices

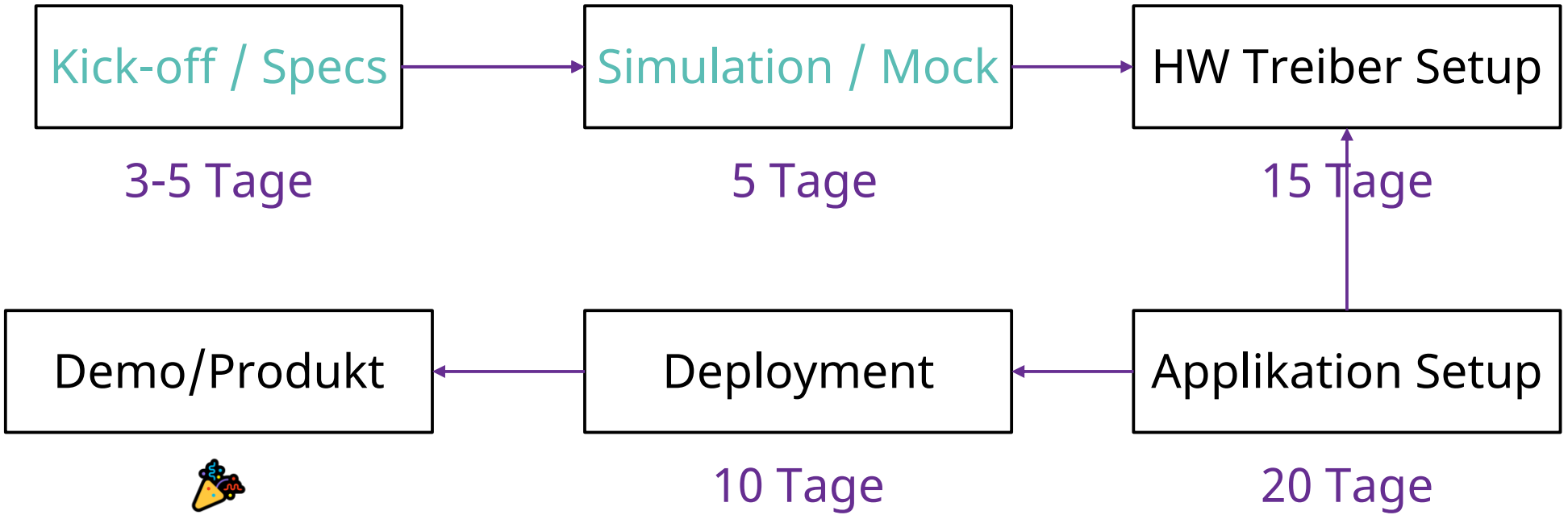
ros2_control - Kernel für ROS 2



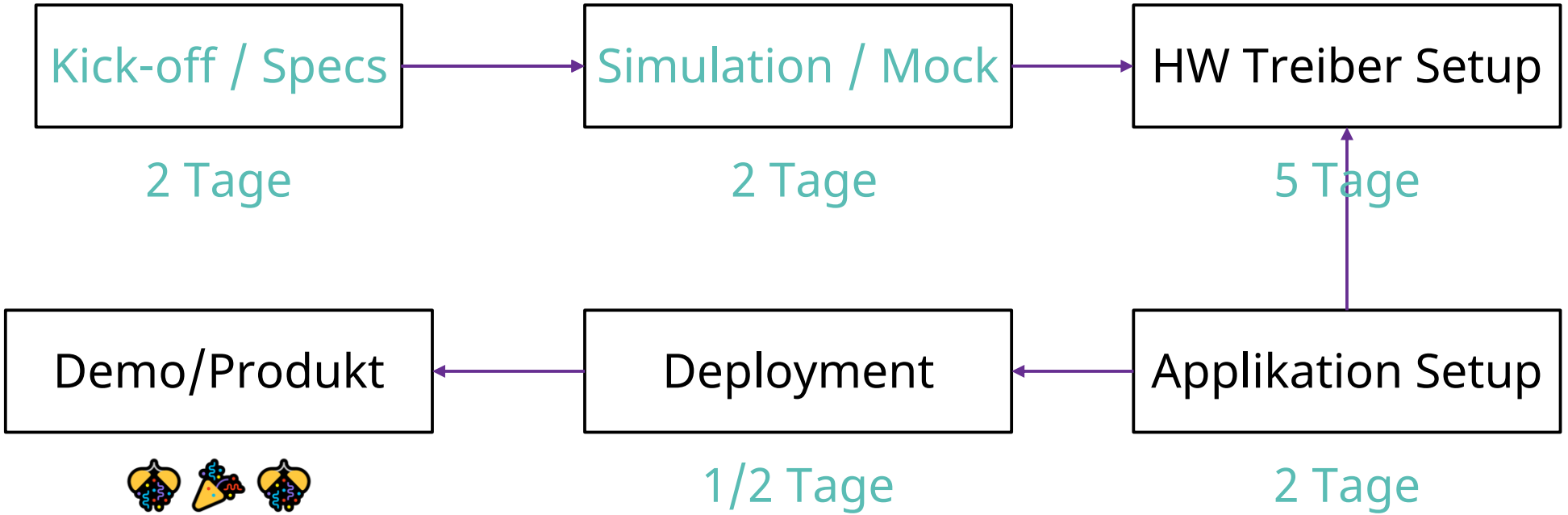
Wie sieht das konkret aus?



Wie lange dauert es?



Was wollen wir?



Tu zuerst das Notwendige, dann das Mögliche, und plötzlich schaffst du das Unmögliche.

Franz von Assisi

Kick-off → Simulation

- Sammeln von Anforderungen
- Klärung der Hardware
- Erstellung von URDF
  - Manuell durch FreeCAD, oder Export-Plugins für andere CAD Software
- Erstellung von Robot Description Paketes
  - RTW Vorlagen
- Integration in ros2_control
  - RTW Vorlagen – MockHW out of the box, Gazebo ~~auch~~ fast

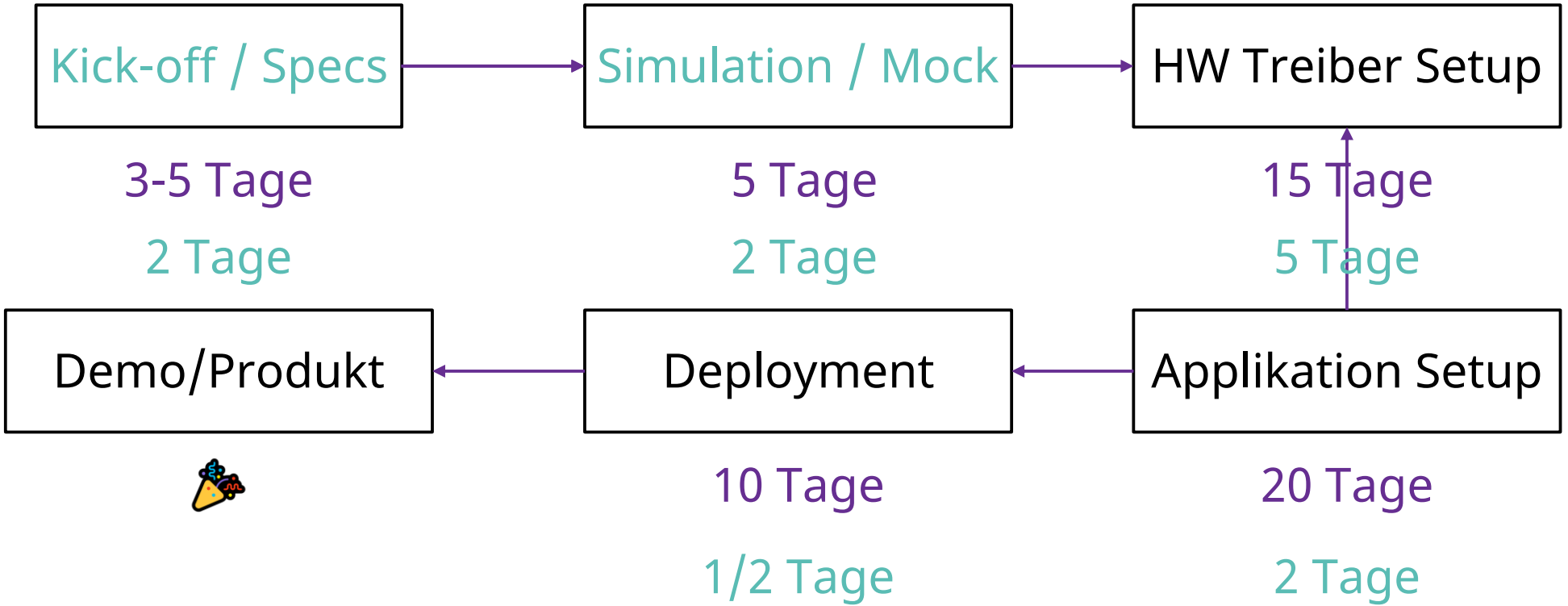
HW Treiber → Applikation

- Integration in ros2_control
  - RTW Vorlagen sehr hilfreich
- Nav2 und MoveIt Integration
  - RTW + Privaten; Setup Assistant, usw...
- Applikation Ablauf
  - Private Scripts, Behavior Trees, sehr viel manuelle Arbeit
- Roboter Einmessen/Kalibrieren
  - Erfahrungsbasierter Prozess

Deployment

- Zielhardware Aussuchen
  - Sehr viel Auswahl am Markt
- Beitreibsystem mit RT Kernel
  - RT_PREEMPT
- Regelung deployen mit RT Kernel
  - ros2_control macht das automatisch, falls möglich
- Applikation Deployen
  - Docker, vcs (*.repos* Files), SNAP, Flatpack, AppImage, etc.

Was wollen wir?



Zielhardware Aussuchen

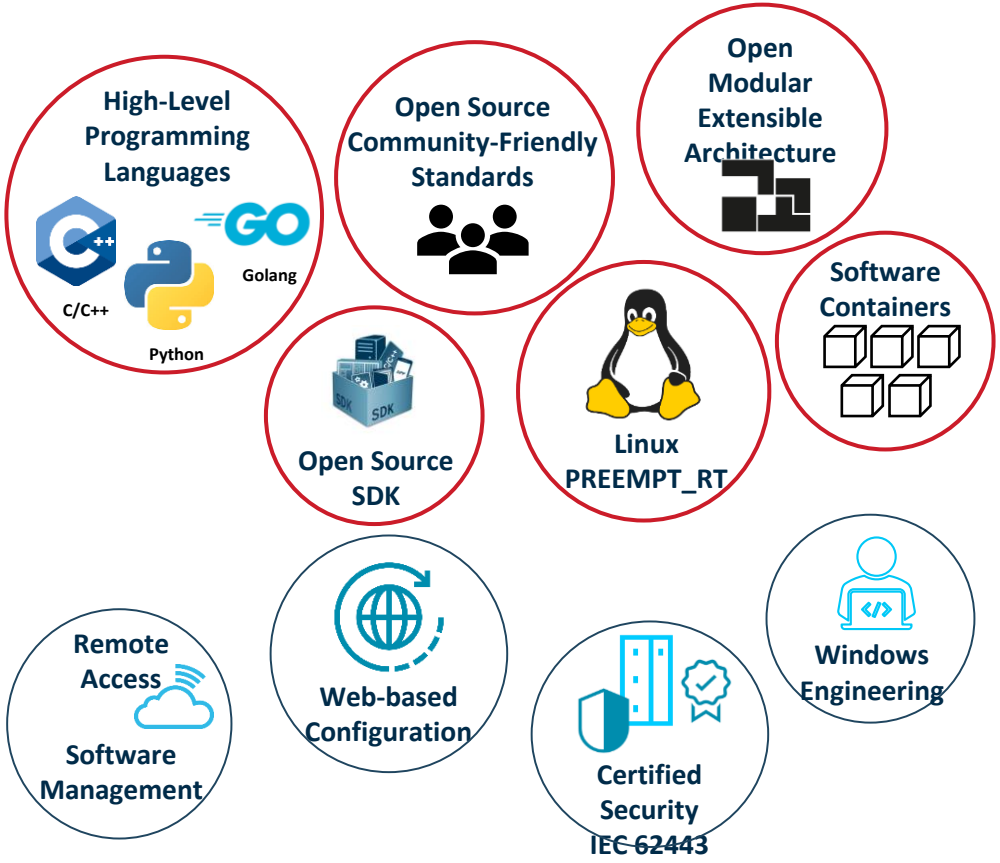
- Linux RT_PREEMPT support
- „Industriebus“ tauglich
- Einfache Integration in existierende Zellen
  - SPS Unterstützung, Hutschiene Montage

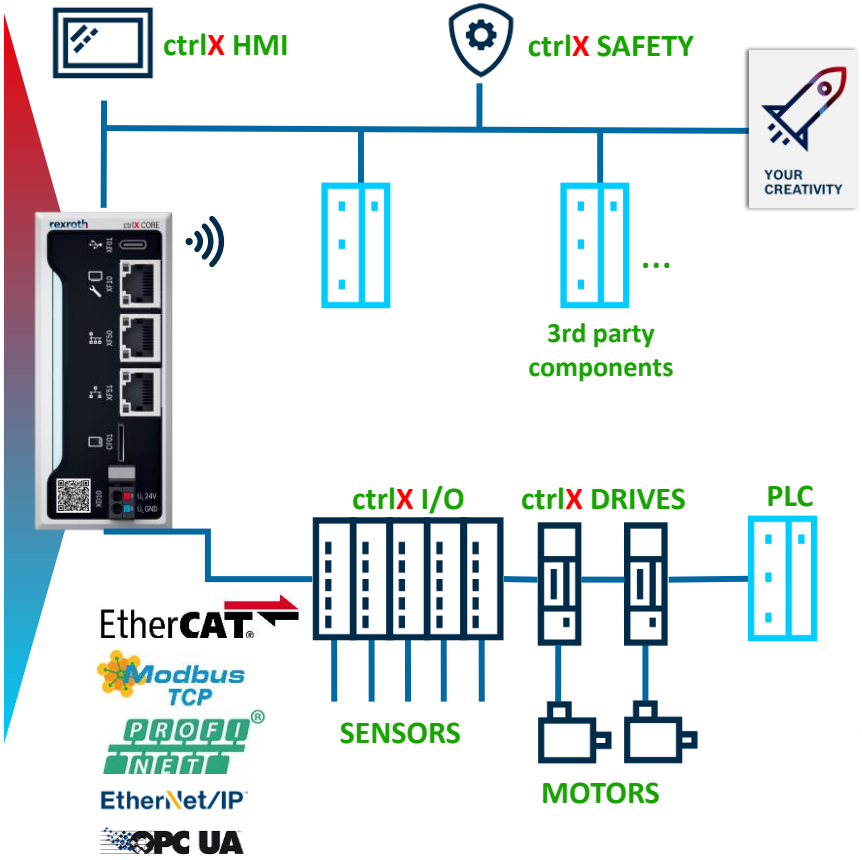
- z.B. IPCs, RevolutionPi, …

Aufsetzen einen RT PCs

- Setup Network
- Setup a control PC
- Setup real-time kernel
- Setup ros2 + ros2_control
- Find correct driver + version + compile
- Test connection and stability
- Installiere Anwendungen des Kundes 😱

Motek Messe 2022...





Deployment: Docker

- Standard Entwicklungsumgebung
- RTW Unterstützung out-of-the-box
- In Kombination mit vcs super-mächtig
- Deployment:
  - Standalone image export
  - Echtzeit funktioniert auch gut
  - CPU/Arch Unterschiede könnten schwieriger sein

Deployment: vcs (.repos Files)

- Standard Entwicklungsumgebung
- RTW Unterstützung out-of-the-box
- In Kombination mit vcs super-mächtig
- Deployment:
  - Bare-bone installation
  - Echtzeit
  - Manchmal nicht 100% reproduzierbar (CPU/Arch unterscheide)

Deployment: SNAP

- Puh… es *is(ch)* schon anders
- Unveränderbare Containers
- Unterschiedliche Architekturen
  - Amd64 – soll auch amd64 gebaut werden
- Besser Security als Docker (wenn noch jemand den physichen Zugriff auf das System hat)

Wie SNAPen?

```
name: ros2-rolling-kuka-experimental-sm-sc
version: '0.1'
summary: ROS 2 rolling kuka experimental driver
description: 'This example launches the ROS2 kuka experimental driver.

    '
confinement: strict
base: core22

parts:
    ros2-kuka-experimental:
        plugin: colcon
        source: https://github.com/StoglRobotics-forks/kuka_experimental.git
        source-branch: sub_modules_v2
        stage-packages:
        - libtinyxml-dev
        - ros-rolling-pluginlib
        - ros-rolling-rclcpp
        - ros-rolling-rclpy
        - ros-rolling-ros-base
        - ros-rolling-ros2launch
        - ros-rolling-std-msgs
        - ros-rolling-xacro
        organize:
          usr/lib/*-linux-gnu/blas/*: usr/lib/
          usr/lib/*-linux-gnu/lapack/*: usr/lib/
        build-packages:
        - libtinyxml-dev
        - ros-rolling-ament-lint-auto
        build-environment:
        -   ROS_VERSION: '2'
        -   ROS_DISTRO: rolling
    ros2-rolling/ros2-launch:
      source: 'https://github.com/StoglRobotics-forks/ros2_rolling_extension.git'
      source-branch: main
      plugin: nil
      override-build: >-
        install -D -m 0755 launch
        ${CRAFT_PART_INSTALL}/snap/command-chain/ros2-launch
      build-packages:
        - ros-rolling-ros-environment
        - ros-rolling-ros-workspace
        - ros-rolling-ament-index-cpp
        - ros-rolling-ament-index-python
```

```
apps:
    ros2-rolling-kuka-experimental-sm-sc:
        command: ''
        command-chain: [snap/command-chain/ros2-launch]
        plugs:
        - network
        - network-bind
        daemon: simple
        passthrough:
            restart-condition: always
            restart-delay: 5s
        environment:
            ROS_VERSION: '2'
            ROS_DISTRO: rolling
            PYTHONPATH: $SNAP/opt/ros/rolling/lib/python3.10/site-packages:$SNAP/usr/lib/
            python3/dist-packages:${PYTHONPATH}
            ROS_HOME: $SNAP_USER_DATA/ros
package-repositories:
-   type: apt
    url: http://repo.ros2.org/ubuntu/main
    components:
    - main
    formats:
    - deb
    key-id: C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
    key-server: keyserver.ubuntu.com
    suites:
    - jammy
```

2023 Version

Wie SNAPen?

```yaml
name: ros2-humble-datalyer-cpp
version: '2.2.0'
summary: Snap with simple ROS 2 listener
description: |
  A simple ROS 2 listener which receives ROS 2 messages on topic 'ros2_simple_cpp'.

base: core22
grade: stable
confinement: strict

parts:
  ros-app:
    plugin: dump
    source: ../../../install/
    stage-packages:
      - libzmq5
      - ctrlx-datalayer
    override-build: |
        snapcraftctl build

  helper-scripts:
    plugin: dump
    source: helper_scripts/
    organize:
      ./run_datalayer_hw.sh : usr/bin/run_datalayer_hw
      ./dds_config/cyclone_dds.xml: etc/dds_config/cyclone_dds.xml

apps:
  run-datalayer-hw:
    command: usr/bin/run_datalayer_hw
    plugs:
      - ros-base
      - network
      - network-bind
    daemon: simple
    passthrough:
      restart-condition: always
      restart-delay: 10s
```

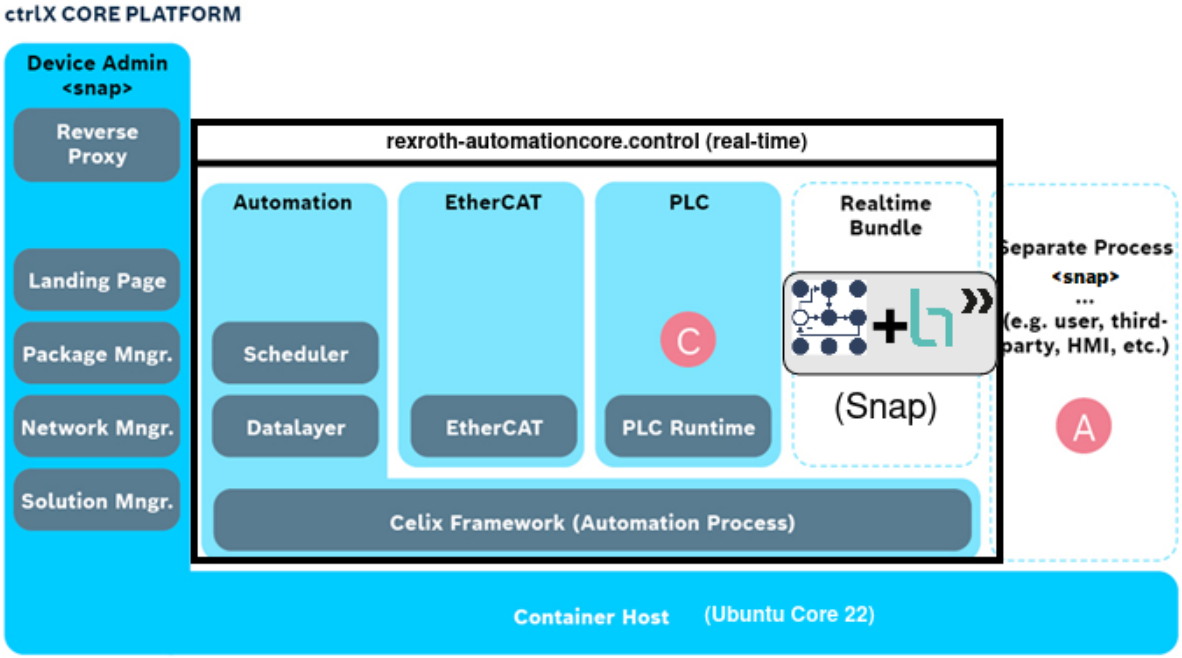
```yaml
plugs:
  ros-base:
    interface: content
    content: executables
    target: $SNAP/rosruntime

  datalayer:
    interface: content
    content: datalayer
    target: $SNAP_DATA/.datalayer
```

2024 Version

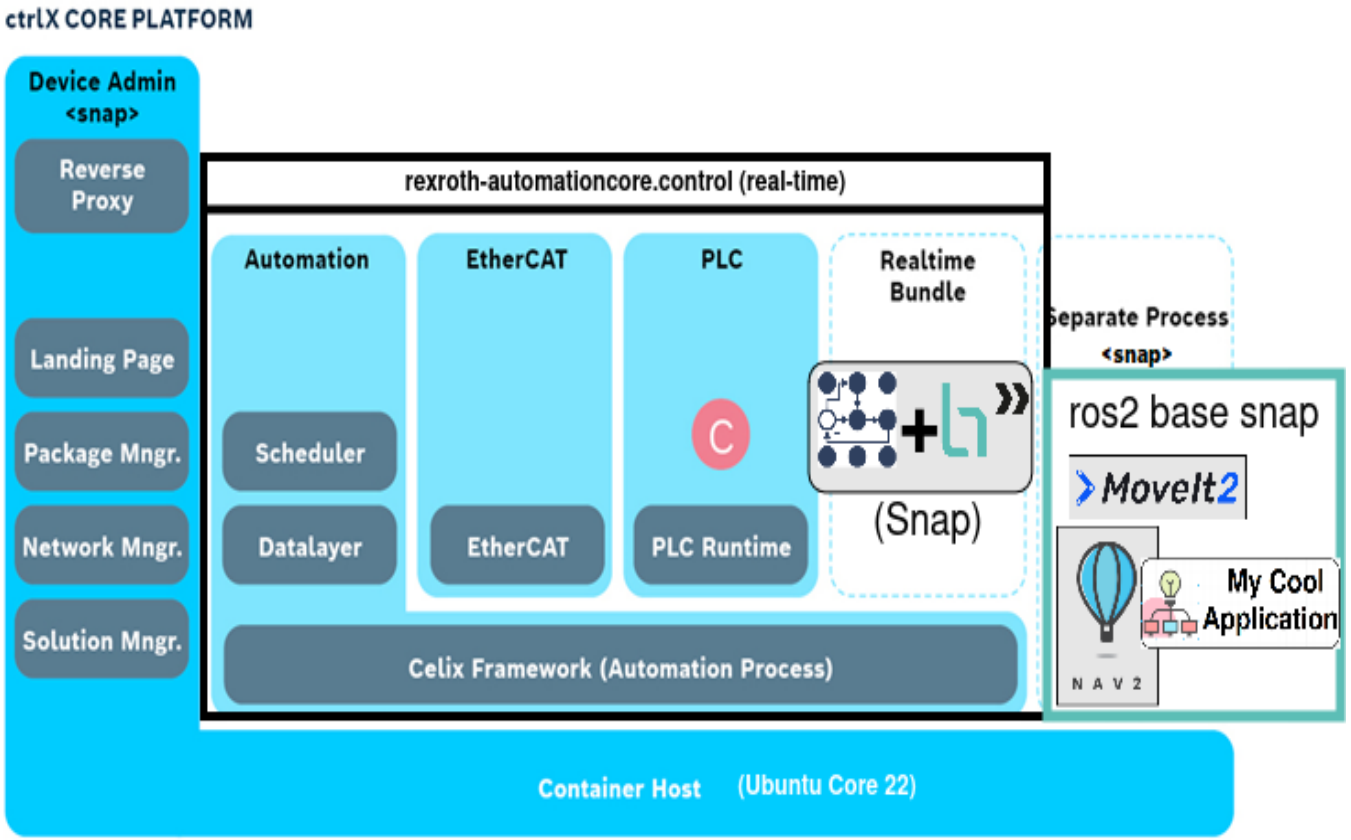
What about RT?

- Celix bundle in SNAP

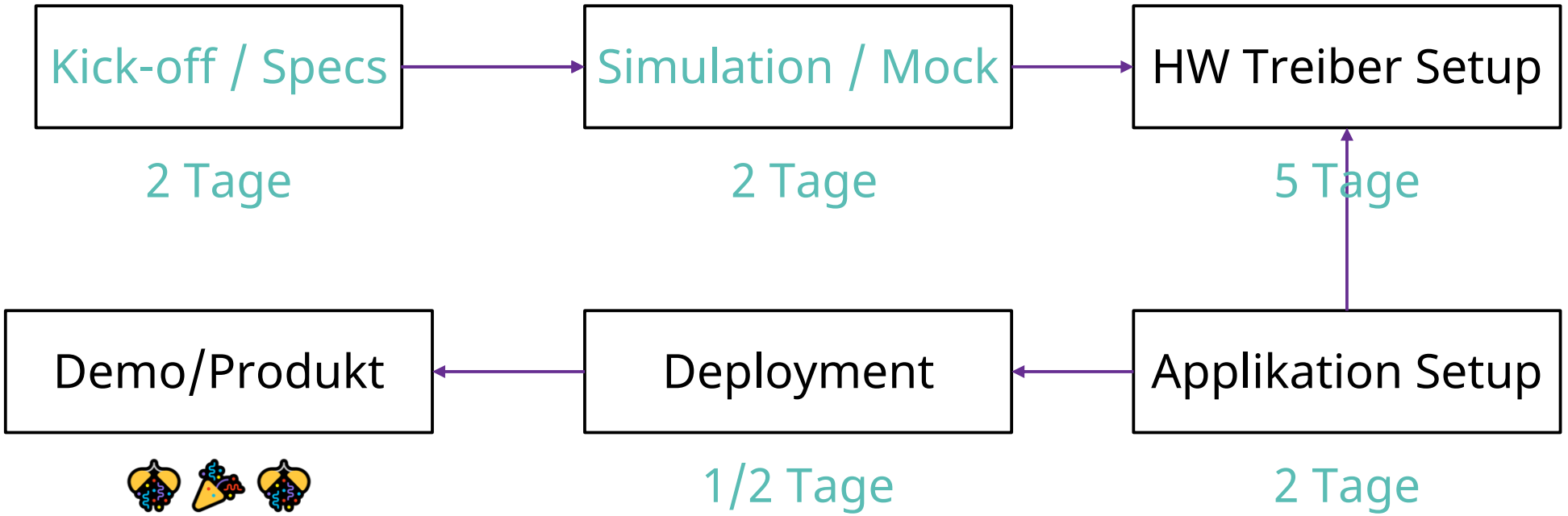


Depoyment auf CtrlX

Depoyment auf CtrlX



Was wollen wir?



robotizer
BY STOGL ROBOTICS

Deployment von ROS 2 Anwendungen für 24/7 Einsatz

DR. DENIS

@

ROSCON DE

2024