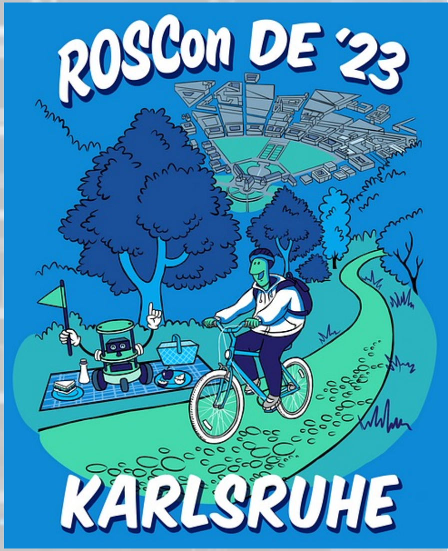
KubeROS for Deploying ROS 2 based Robotic Applications with Kubernetes

- Challenges, Concept, Architecture, and Case Study

Yongzhou Zhang [1,2]; Gergely Soti [1,2]; Björn Hein[1,2]

[1] Karlsruhe University of Applied Sciences – Institut for Robotics and Autonomous Systems (IRAS)

[2] Karlsruhe Institute of Technology (KIT)



2023/11/23

Hochschule Karlsruhe University of Applied Sciences HKA

KIT Karlsruhe Institute of Technology

Acknowledgement

Acknowledgment: parts of this work are conducted as part of the project **KI5GRob** funded by German Federal Ministry of Education and Research (**BMBF**).

Bundesministerium
für Bildung
und Forschung

Hochschule Karlsruhe
University of
Applied Sciences

HKA KUKA SCHMALZ AAT® roboception

Context

Q1: Are the onboard computing resources sufficient?

Q2: Is robot software becoming increasingly complex?

Q3: How to scale the robotic system based on ROS/ROS2?

Example: Mobile Manipulator for Industry

Q1: Are the onboard computing resources sufficient?

Q2: Is robot software becoming increasingly complex?

Q3: How to scale the robotic system based on ROS/ROS2?

One of previous project: **QBIIK** - An autonomous comminision system with learning capbility for logistics.

iGo neo
QBIIK

BÄR STILL Audi Sport KIT IFL IAR-IPR Federal Ministry of Economics and Technology

https://www.kit-technology.de/de/blog/picken-und-packen-mit-qbiik

https://www.digitale-technologien.de/DT/Redaktion/DE/Textbausteine/PAiCE_Marginalspalten/PAiCE_Projekt_QBIIK_Infokarte.pdf?__blob=publicationFile&v=1

Example: Mobile Manipulator for Industry

Q1: Are the onboard computing resources sufficient?

More CPU/GPU/RAM

Q2: Is robot software becoming increasingly complex?

Many modules, frequently updated

Q3: How to scale the robotic system based on ROS/ROS2?

Drivers for sensor & actuator:
- camera
- trailer actuator
- rotation table
- lidar
- etc.

Forklift truck (AMR):
- Localization
- Mapping
- Navigation
- Planning
- Controller
- etc.

Gripper:
- Motor controller with CANOpen
- gripper controller

User interface:
- task management system
- training data management system
- graphical user interface (GUI)

Perception:
- object detection and segmentation
- camera calibration
- grasping pose estimation
- shelf and placing slot localization

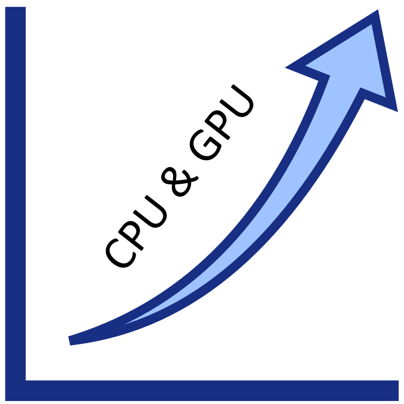
Grasping and Placing:
- grasping process control
- placing process control

Robot arm:
- motion planning
- trajectory generation
- arm controller
- fanuc driver
- safety module

Task planning:
- finit state machine
- interface to WMS

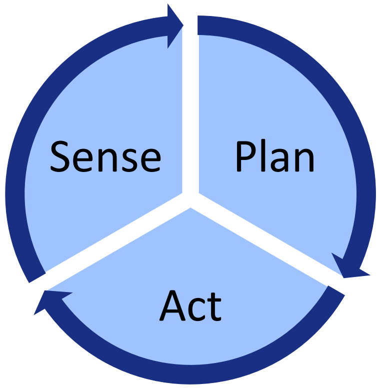
Tele-operation:
- interface for human assistant
- control interface for VR/AR

Three Challenges



Onboard computing resources are insufficient

Computing resources from cloud and edge



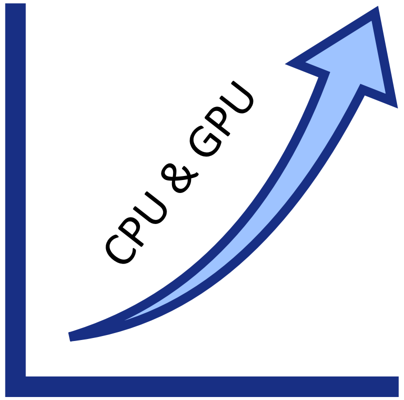
High software complexity of the entire system

Containerized software modules

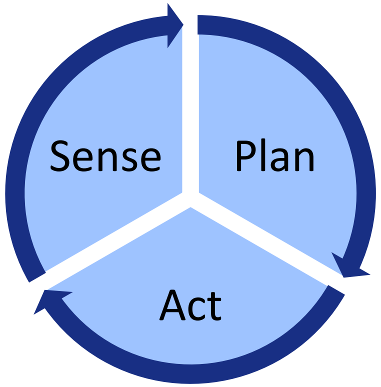
Deployment at large scale

Deployment and orchestration with Kubernetes

Leveraging the Cloud and Edge Computing



Onboard computing resources are insufficient



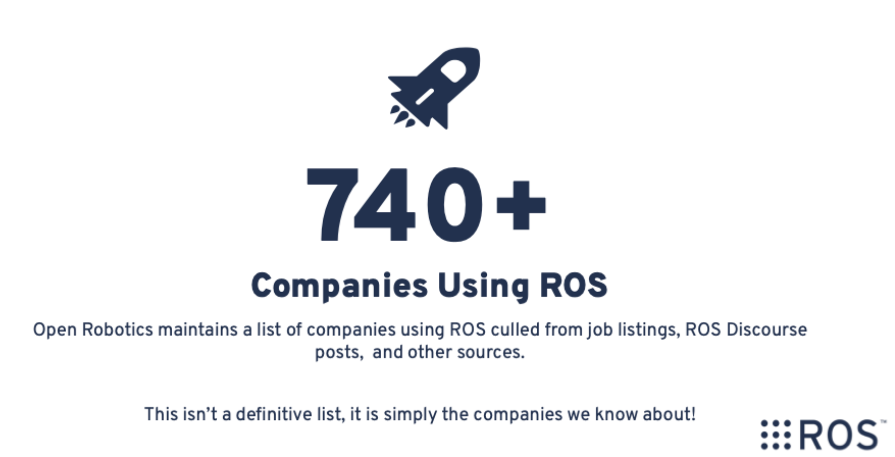
High software complexity of the entire system

Deployment at large scale

ROS2 + kubernetes

ROS 2 with Kubernetes

ROS 2 + kubernetes



740+

Companies Using ROS

Open Robotics maintains a list of companies using ROS culled from job listings, ROS Discourse posts, and other sources.

This isn't a definitive list, it is simply the companies we know about!

ROS

- Cloud OS
- Production grade container orchestration
- Flexibility and adaptability
- Scalability
- Ability to manage complex, distributed applications
- Etc.

http://download.ros.org/downloads/metrics/metrics-report-2022-07.pdf

Pains of Combining two Complex Systems

- Networking and communication
- Complex setup and configuration
- Access to the hardware
- Dynamic resource allocation
- Latency
- Containerization granularity
- Integration with existing systems
- Etc.

ROS2

kubernetes

A Layer Between Two Systems?

ROS 2

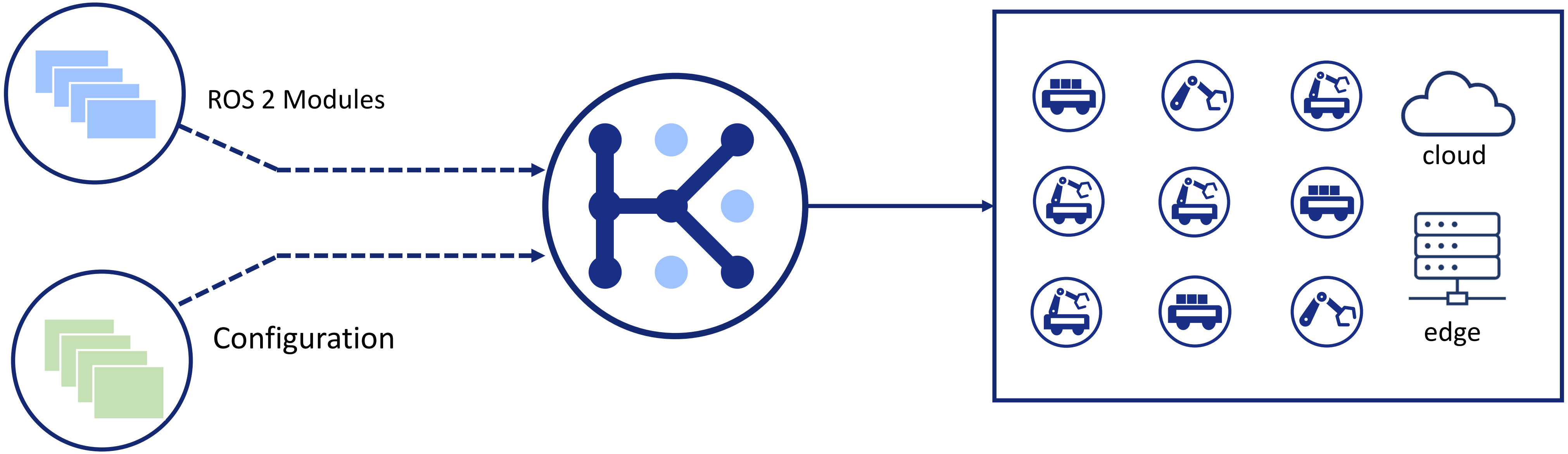
?

KubeROS

kubernetes

KubeROS Concept

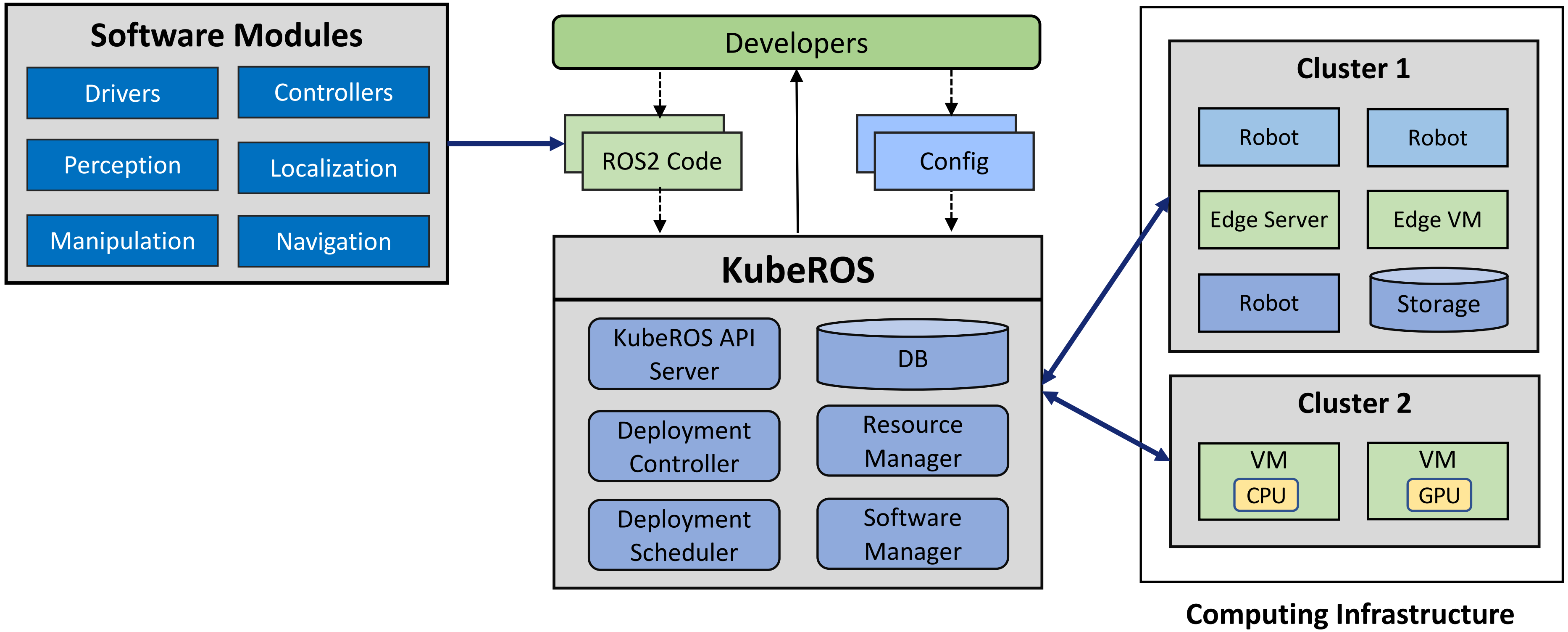
- Abstract the onboard devices, cloud/edge resources as a unified computing infrastructure
- Using Kubernetes to orchestrate the containerized ROS 2 software modules
- Hide the complex underlying framework
- Provide an easy-to-use interface for developers



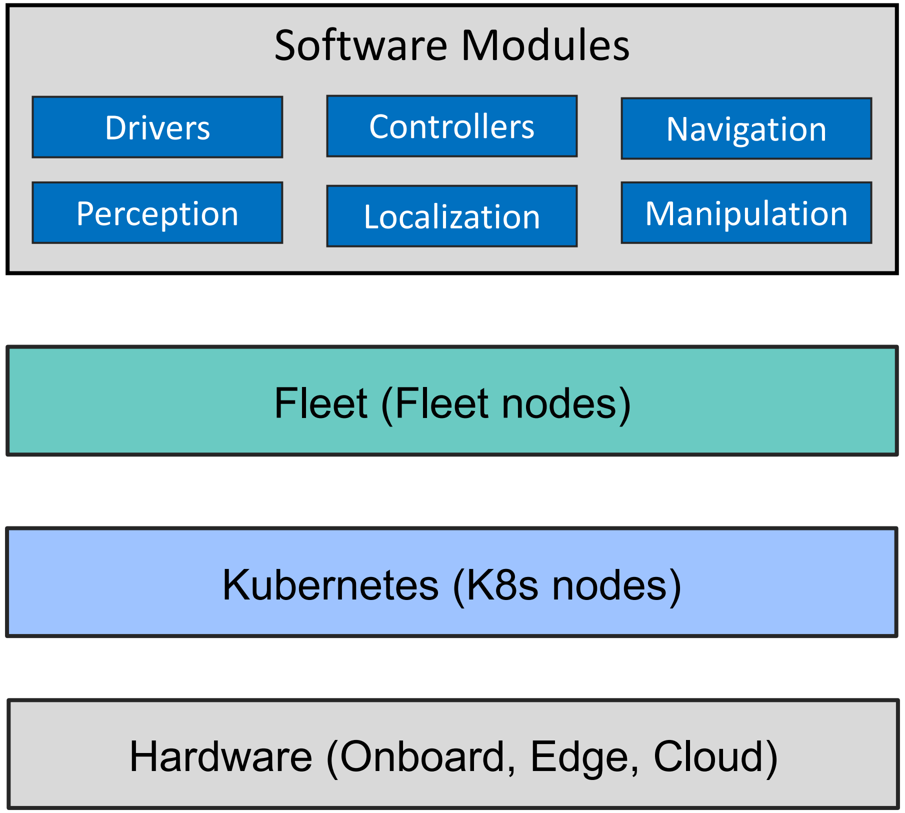
Containerized ROS 2 Modules **Automated Deployment** **Unified Computing Infrastructure**

KubeROS High-Level Architecture

- Abstract the onboard devices, cloud/edge resources as a **unified computing infrastructure**
- Using **Kubernetes** to orchestrate the containerized ROS 2 software modules



Abstraction between Software and Hardware



Software Modules

Drivers

Controllers

Navigation

Perception

Localization

Manipulation

Fleet (Fleet nodes)

Kubernetes (K8s nodes)

Hardware (Onboard, Edge, Cloud)

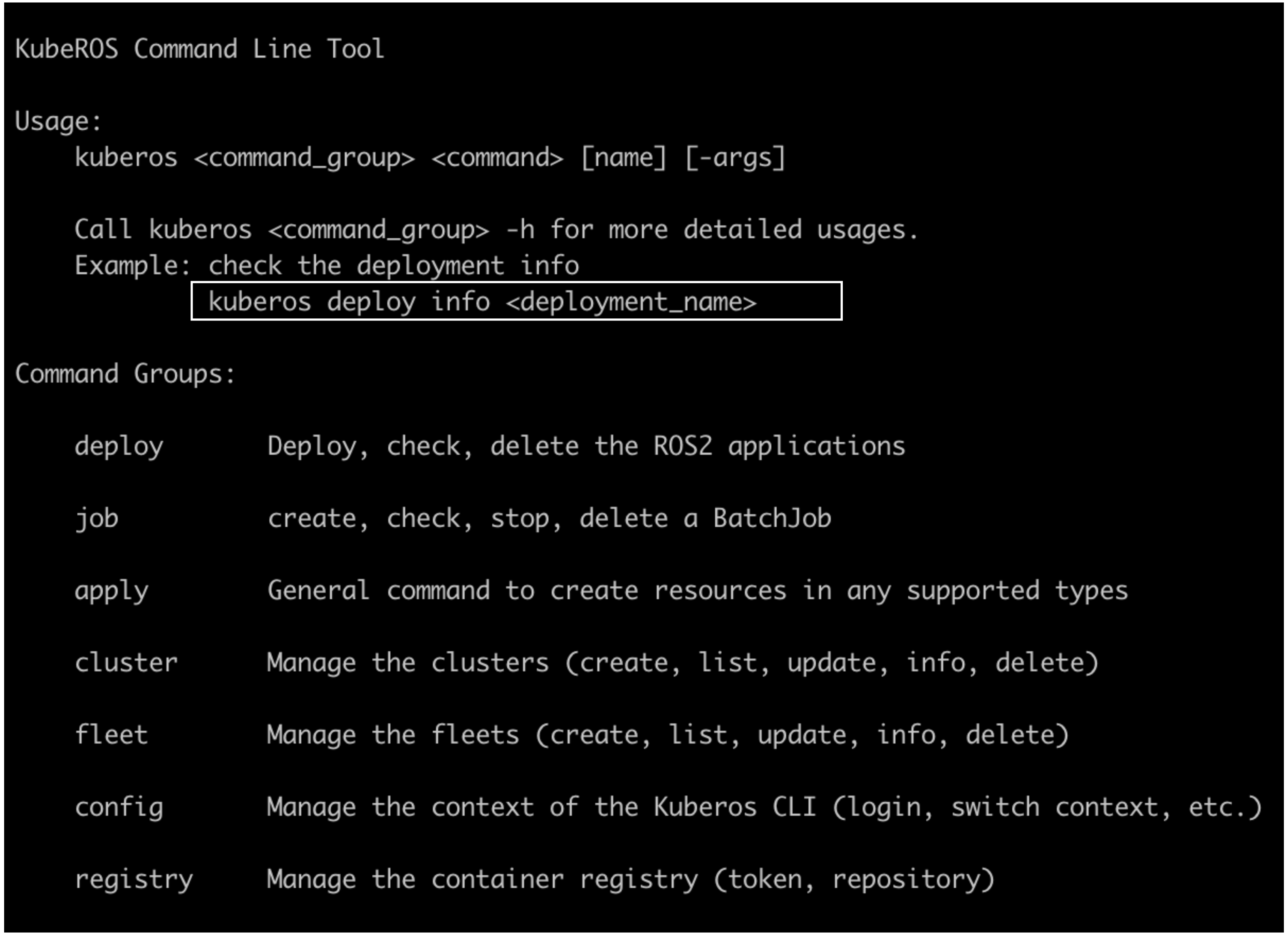
Create fleet for deployment with FleetManifest

Robot hardware specifications in ClusterInventory

Interface: KubeROS-CLI

- Similar to the ros2cli command tools
- Auto-completion is enabled
- Install via pip

```
$ pip install kuberos-cli
```

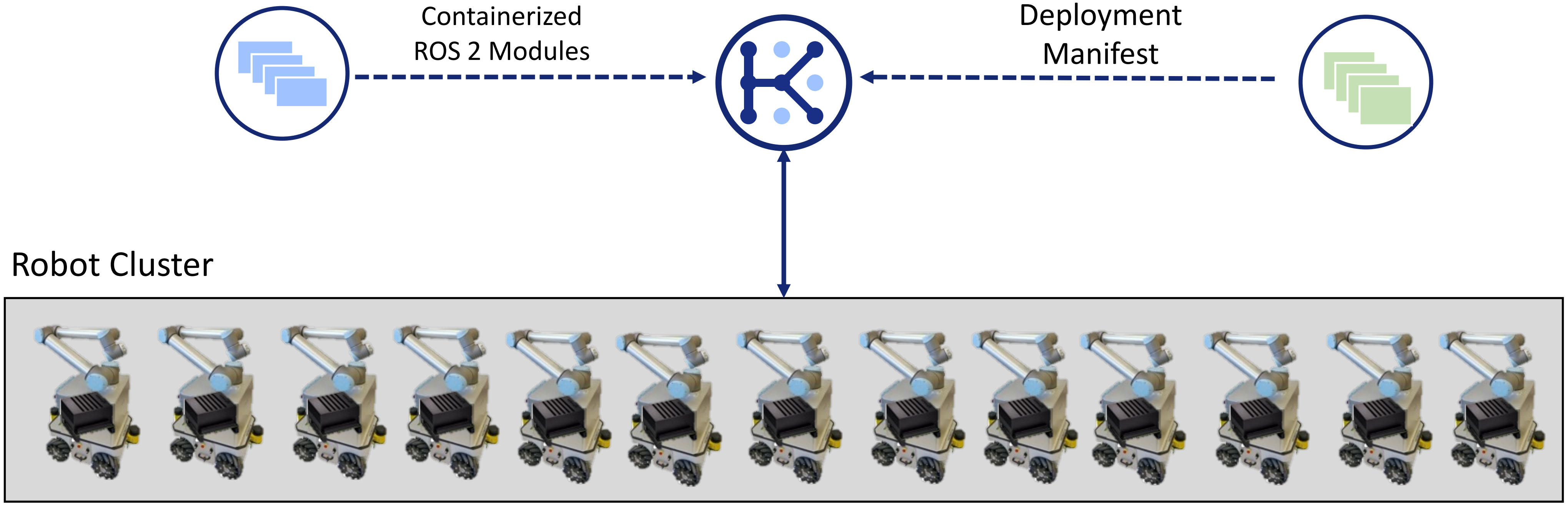


```
KubeROS Command Line Tool

Usage:
    kuberos <command_group> <command> [name] [-args]

    Call kuberos <command_group> -h for more detailed usages.
    Example: check the deployment info
             kuberos deploy info <deployment_name>

Command Groups:

    deploy       Deploy, check, delete the ROS2 applications

    job          create, check, stop, delete a BatchJob

    apply        General command to create resources in any supported types

    cluster      Manage the clusters (create, list, update, info, delete)

    fleet        Manage the fleets (create, list, update, info, delete)

    config       Manage the context of the Kuberos CLI (login, switch context, etc.)

    registry     Manage the container registry (token, repository)
```

https://github.com/kuberos-io/kuberos-cli

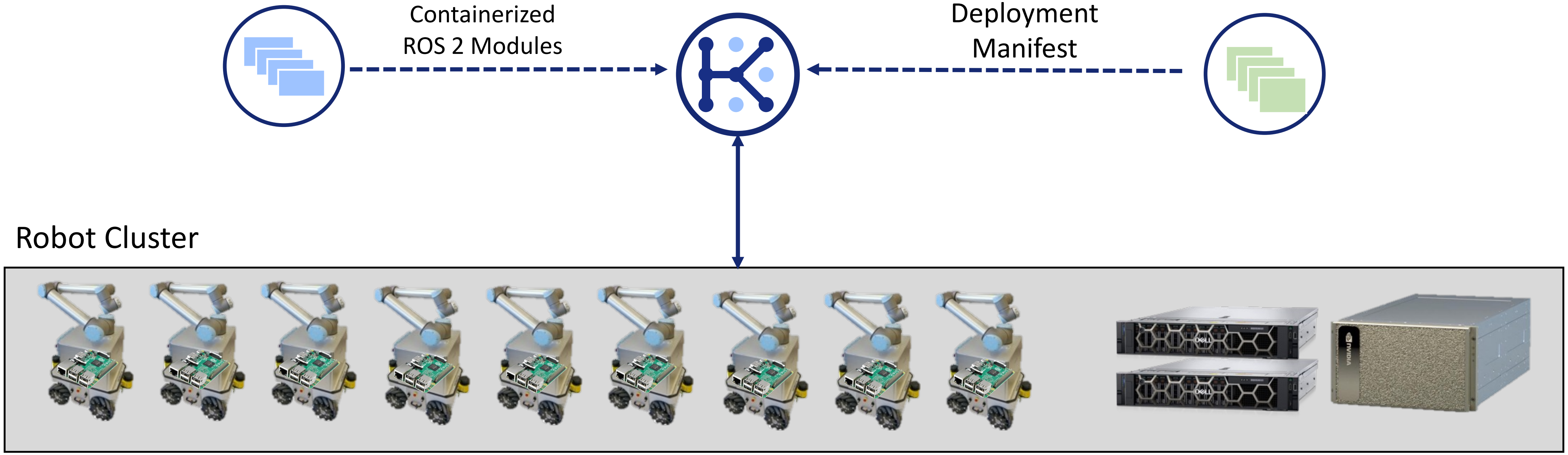
Robot Onboard Only Deployment

- Large robot fleet (10+)
- 10+ containers for one application
- Sufficient onboard computer resources



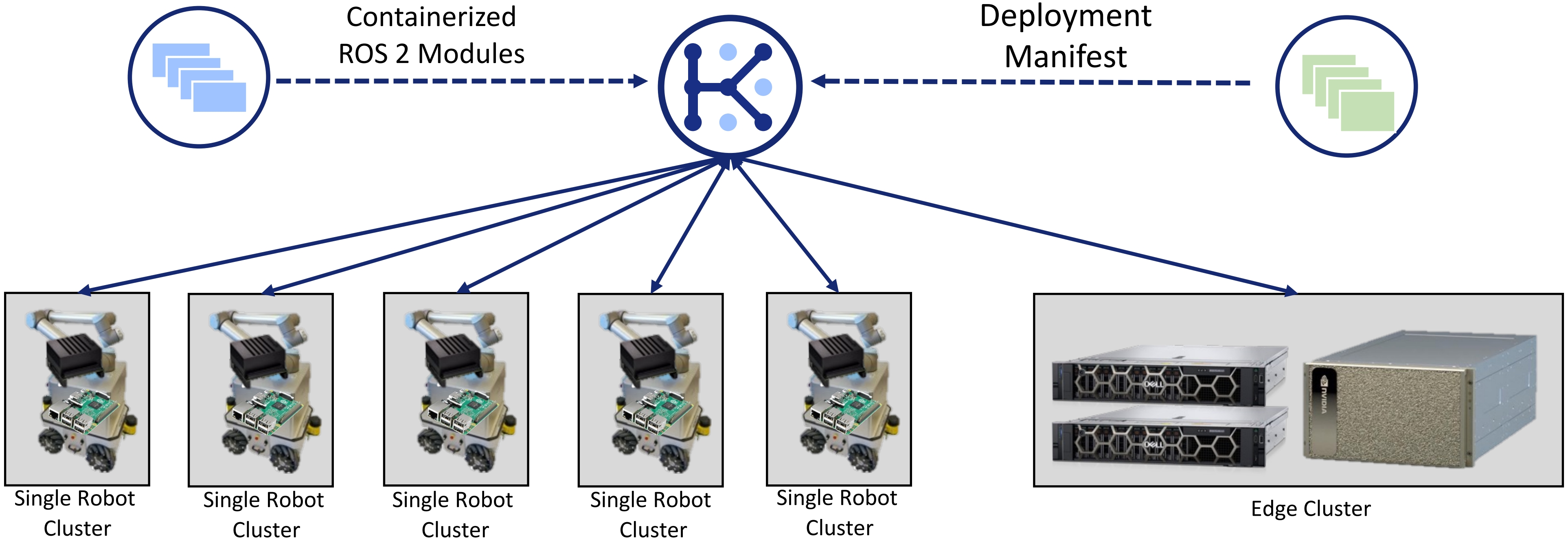
Robots with Edge/Cloud Computing

- Large robot fleet (10+)
- 10+ containers for one application
- Onboard computer resources are insufficient



Robot as Standalone Cluster

- Large robot fleet (10+)
- Multiple onboard computers
- Self-healing even during network connection loss

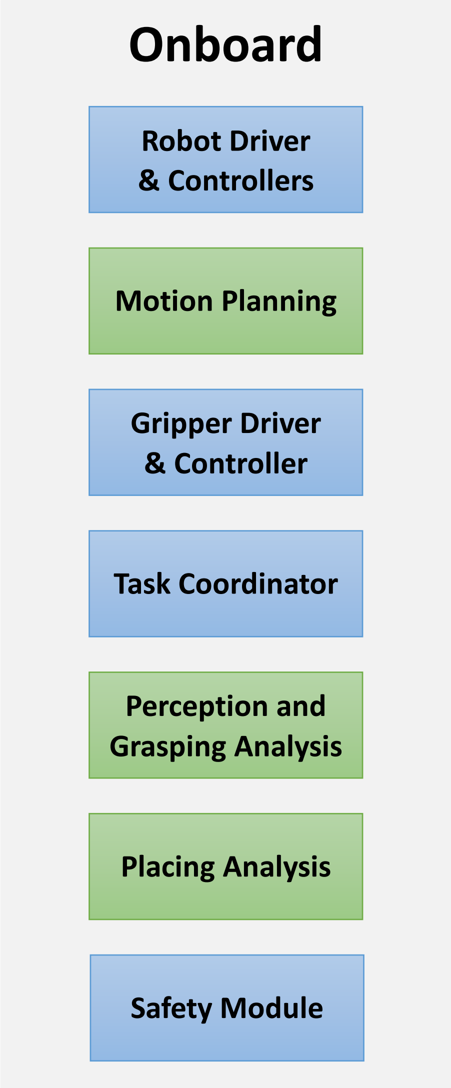


Case Study

A: Pick-and-Place with Manipulators

B: Navigation of Mobile Robots

Case Study A: Pick and Place with Manipulators



Onboard

Robot Driver & Controllers

Motion Planning

Gripper Driver & Controller

Task Coordinator

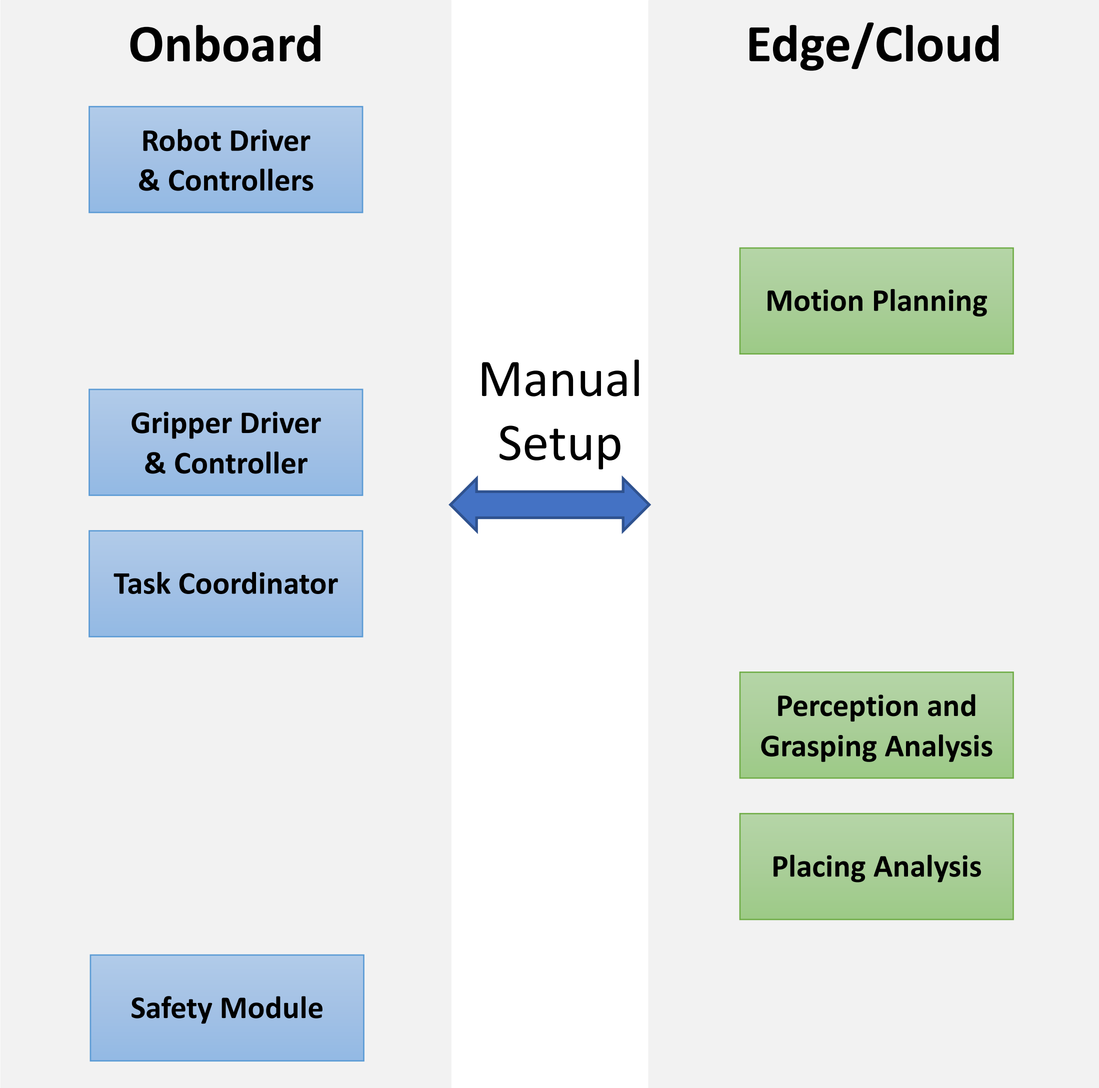
Perception and Grasping Analysis

Placing Analysis

Safety Module

A **high performance** onboard computer is required

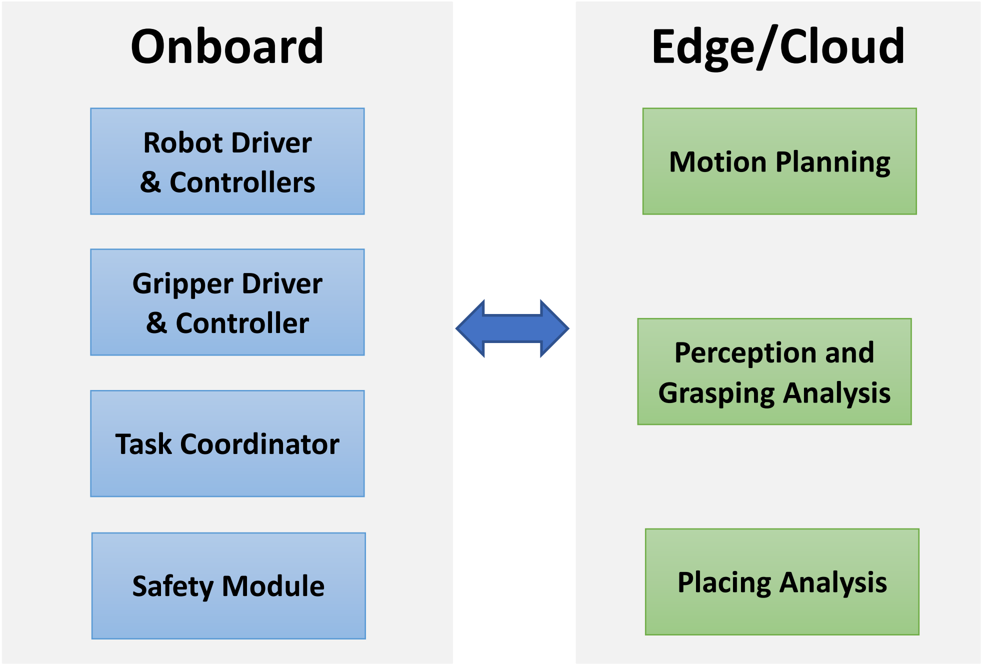
Case Study A: Pick and Place with Manipulators



Case Study A: Pick and Place with Manipulators

Classic Setup

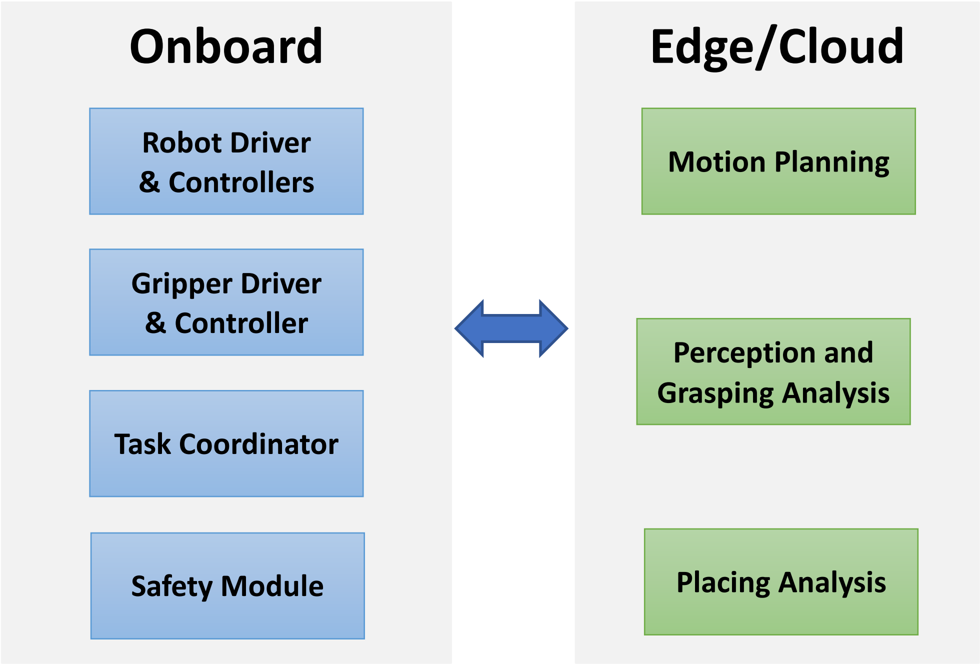
- Setup networking
- Configure DDS domain ID
- Set namespace for multi-robots
- Bash-script to start different processes


Onboard
Robot Driver & Controllers
Gripper Driver & Controller
Task Coordinator
Safety Module
Edge/Cloud
Motion Planning
Perception and Grasping Analysis
Placing Analysis

Case Study A: Pick and Place with Manipulators

Classic Setup

- Setup networking
- Configure DDS domain ID
- Set namespace for multi-robots
- Bash-script to start different processes



Onboard

Robot Driver & Controllers

Gripper Driver & Controller

Task Coordinator

Safety Module

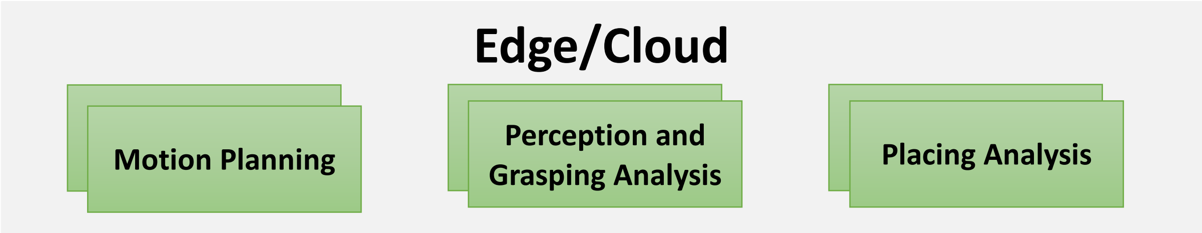
Edge/Cloud

Motion Planning

Perception and Grasping Analysis

Placing Analysis

It becomes more complex as the system scale increases.



Edge/Cloud

Motion Planning

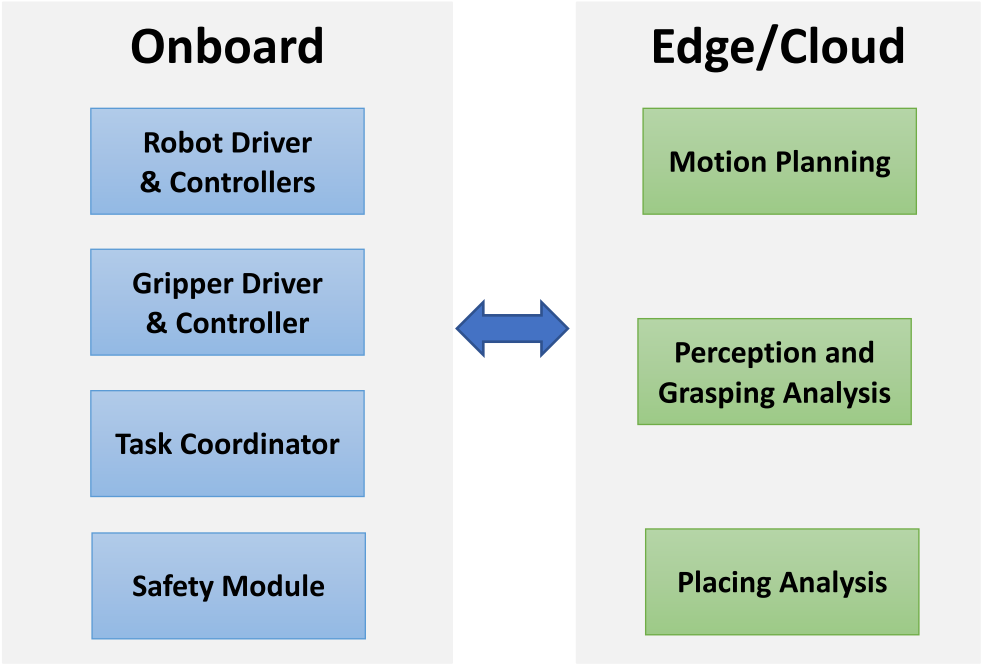
Perception and Grasping Analysis

Placing Analysis

Case Study A: Pick and Place with Manipulators

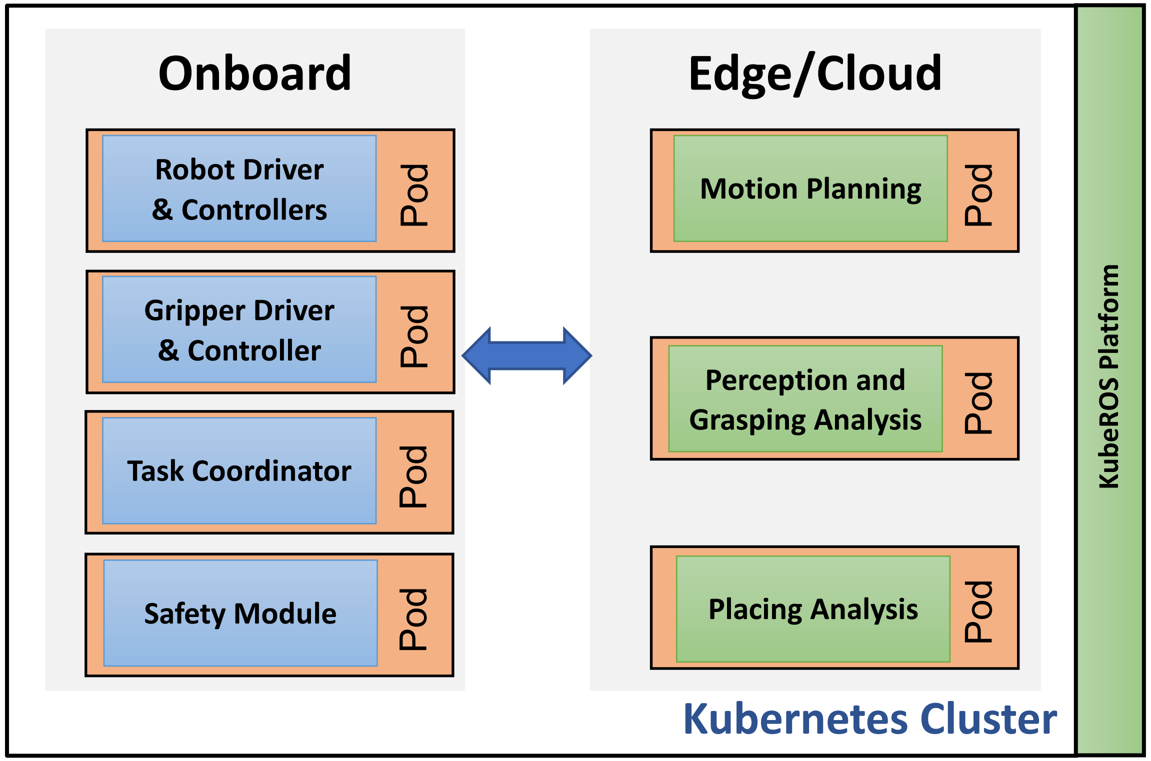
Classic Setup

- Setup networking
- Configure DDS domain ID
- Set namespace for multi-robots
- Bash-script to start different processes


Onboard
Robot Driver & Controllers
Gripper Driver & Controller
Task Coordinator
Safety Module
Edge/Cloud
Motion Planning
Perception and Grasping Analysis
Placing Analysis

With Kubernetes and KubeROS

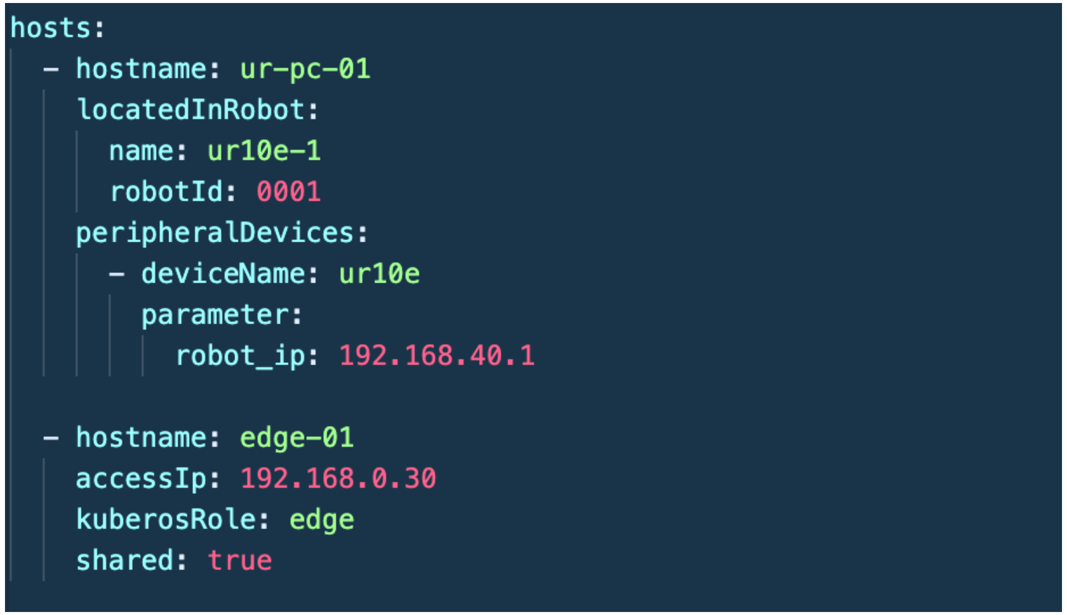
- Each module is run in an isolated container
- Manageable hardware via ClusterInventory
- Declarative software modules in ApplicationDeployment



Case Study A: Pick and Place with Manipulators

Infrastructure as a Code (IaaC):

Describe the infrastructure in the ClusterInventory with robot specifications

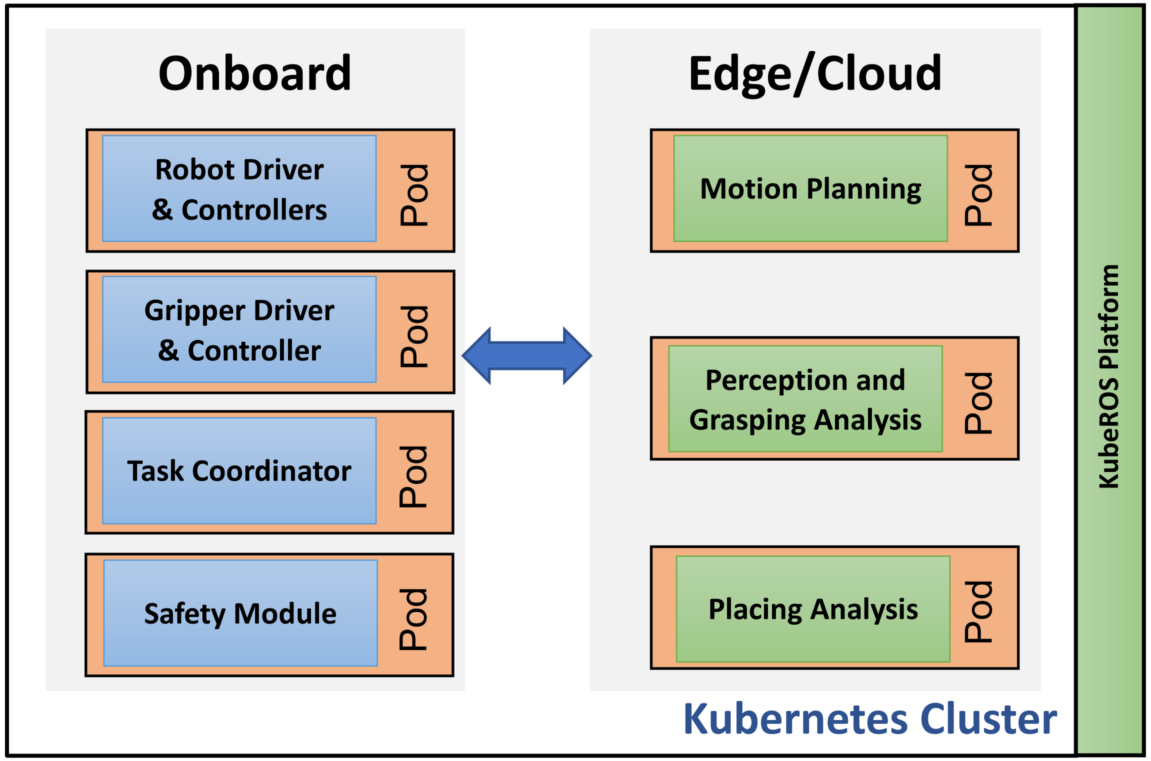


```yaml
hosts:
  - hostname: ur-pc-01
    locatedInRobot:
      name: ur10e-1
      robotId: 0001
    peripheralDevices:
      - deviceName: ur10e
        parameter:
          robot_ip: 192.168.40.1

  - hostname: edge-01
    accessIp: 192.168.0.30
    kuberosRole: edge
    shared: true
```

```
$ kuberos cluster update -f ur-cluster.yaml
```

With Kubernetes and KubeROS

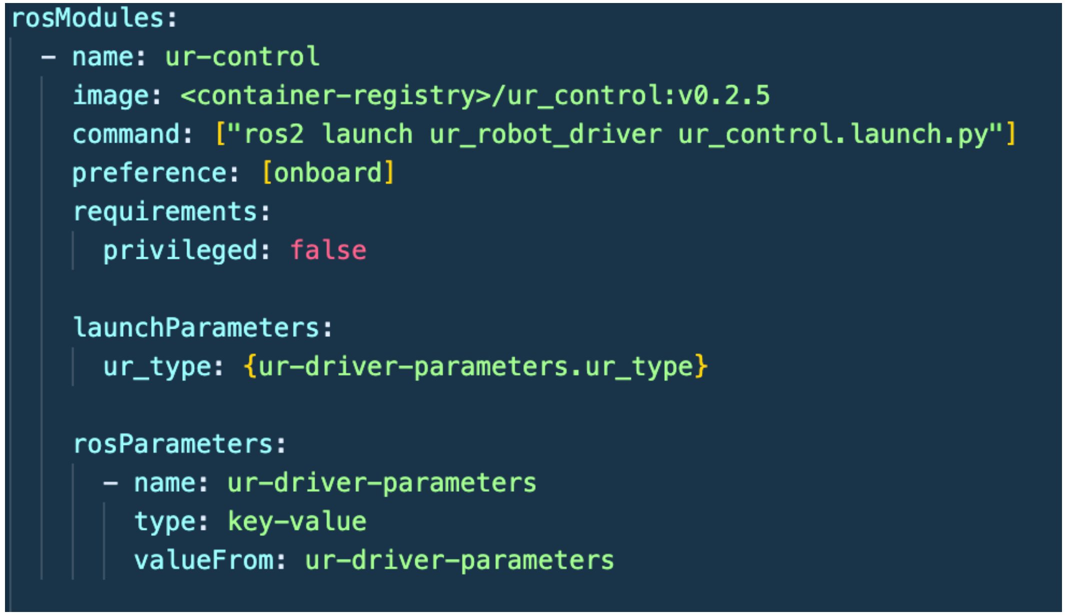
- Each module is run in an isolated container
- Manageable hardware via ClusterInventory
- Declarative software modules in ApplicationDeployment



Case Study A: Pick and Place with Manipulators

ROS 2 Modules for the deployment:

container image, launch parameters,
ROS parameters, deployment requirements

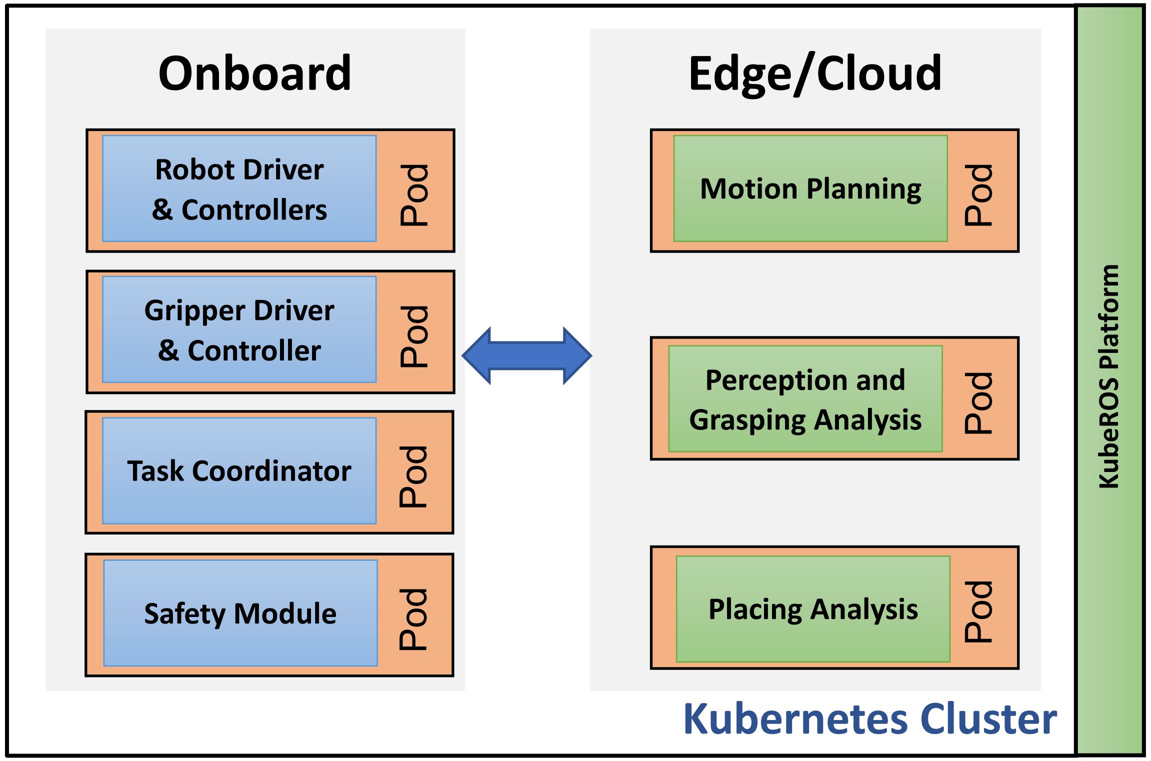


```
rosModules:
  - name: ur-control
    image: <container-registry>/ur_control:v0.2.5
    command: ["ros2 launch ur_robot_driver ur_control.launch.py"]
    preference: [onboard]
    requirements:
      privileged: false

    launchParameters:
      ur_type: {ur-driver-parameters.ur_type}

    rosParameters:
      - name: ur-driver-parameters
        type: key-value
        valueFrom: ur-driver-parameters
```

```
$ kuberos deploy create -f ur-manipulation.yaml
```

With Kubernetes and KubeROS

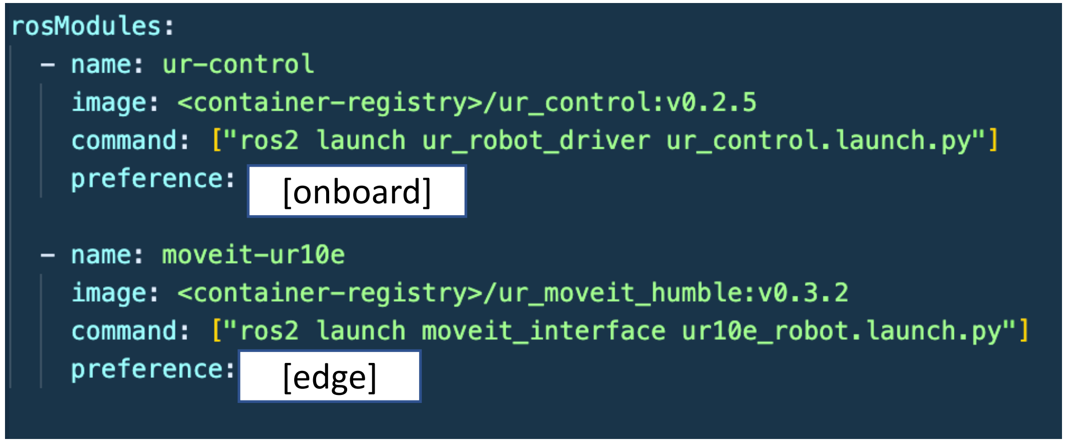
- Each module is run in an isolated container
- Manageable hardware via ClusterInventory
- Declarative software modules in ApplicationDeployment



Case Study A: Pick and Place with Manipulators

Offload to the edge:

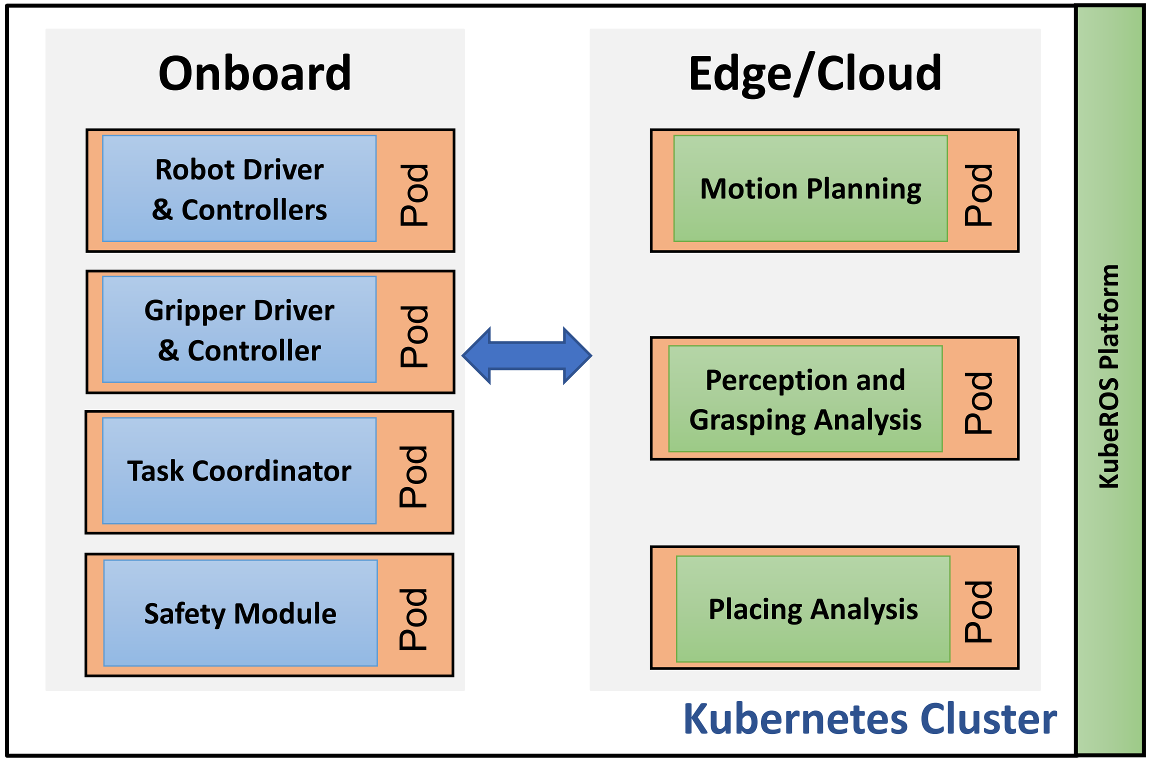
change the preference



```
rosModules:
  - name: ur-control
    image: <container-registry>/ur_control:v0.2.5
    command: ["ros2 launch ur_robot_driver ur_control.launch.py"]
    preference: [onboard]

  - name: moveit-ur10e
    image: <container-registry>/ur_moveit_humble:v0.3.2
    command: ["ros2 launch moveit_interface ur10e_robot.launch.py"]
    preference: [edge]
```

With Kubernetes and KubeROS

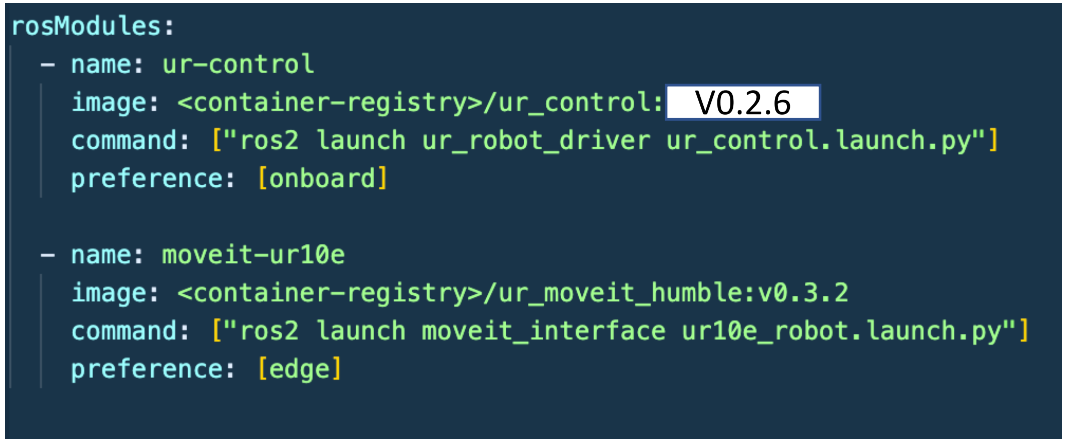
- Each module is run in an isolated container
- Manageable hardware via ClusterInventory
- Declarative software modules in ApplicationDeployment



Case Study A: Pick and Place with Manipulators

Software update:

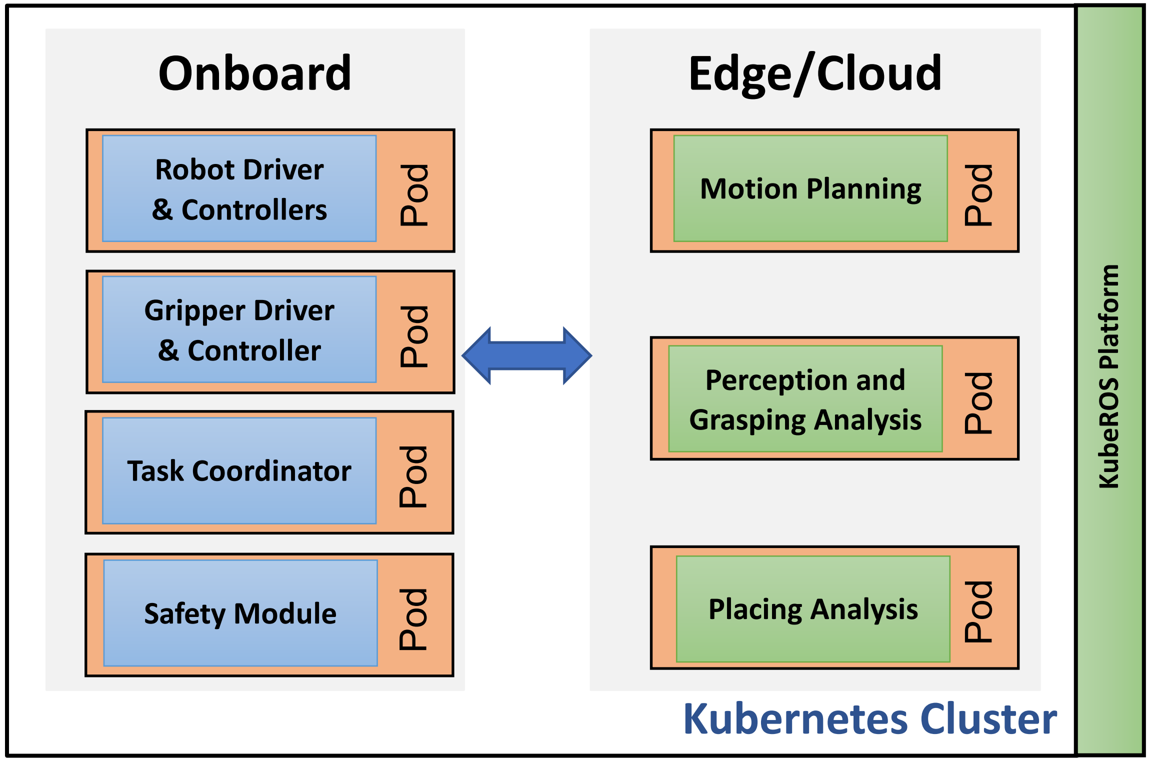
update the container image tag or address



```
rosModules:
  - name: ur-control
    image: <container-registry>/ur_control: V0.2.6
    command: ["ros2 launch ur_robot_driver ur_control.launch.py"]
    preference: [onboard]

  - name: moveit-ur10e
    image: <container-registry>/ur_moveit_humble:v0.3.2
    command: ["ros2 launch moveit_interface ur10e_robot.launch.py"]
    preference: [edge]
```

With Kubernetes and KubeROS

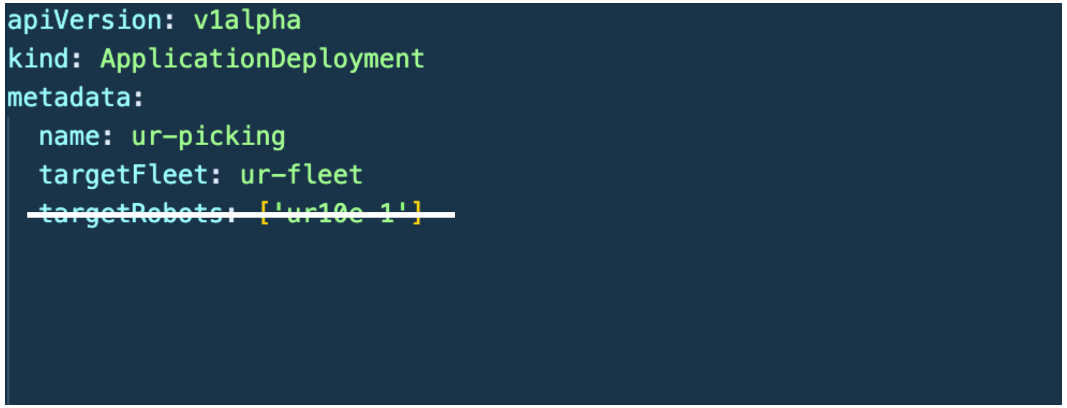
- Each module is run in an isolated container
- Manageable hardware via ClusterInventory
- Declarative software modules in ApplicationDeployment



Case Study A: Pick and Place with Manipulators

Scale to the entire fleet:

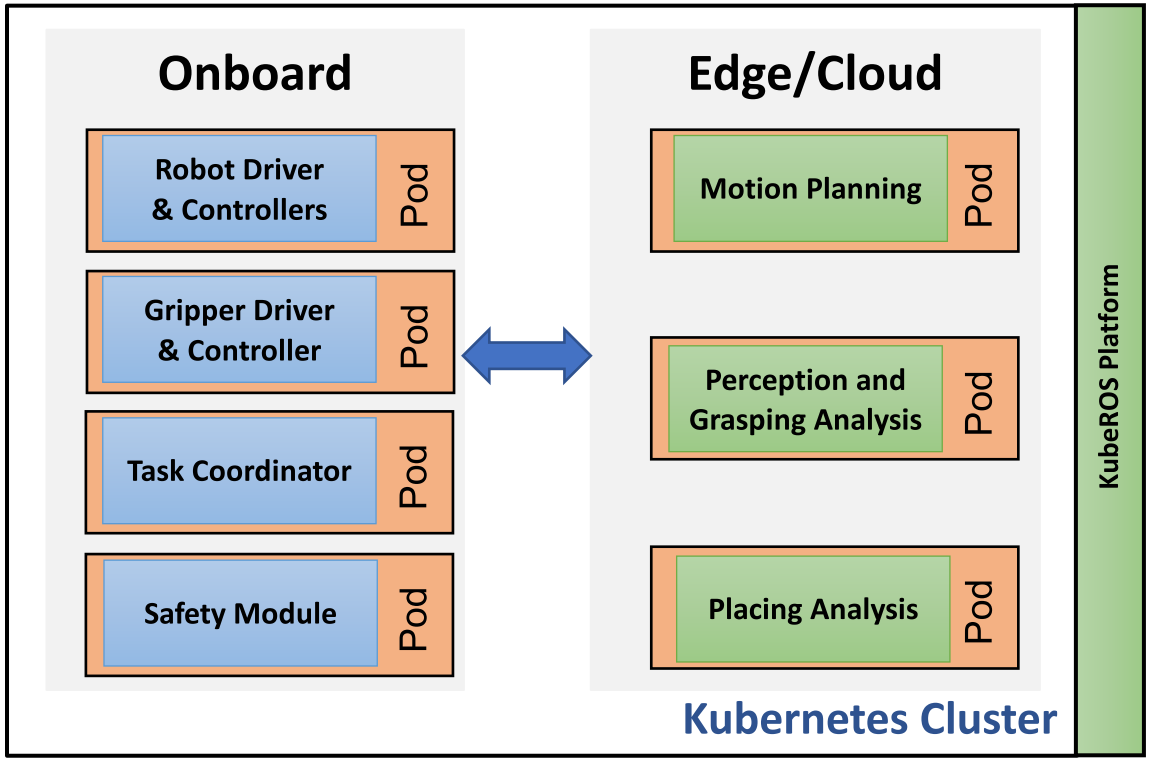
remove the specified targetRobots



```
apiVersion: v1alpha
kind: ApplicationDeployment
metadata:
  name: ur-picking
  targetFleet: ur-fleet
  ~~targetRobots: ['ur10e_1']~~
```

With Kubernetes and KubeROS

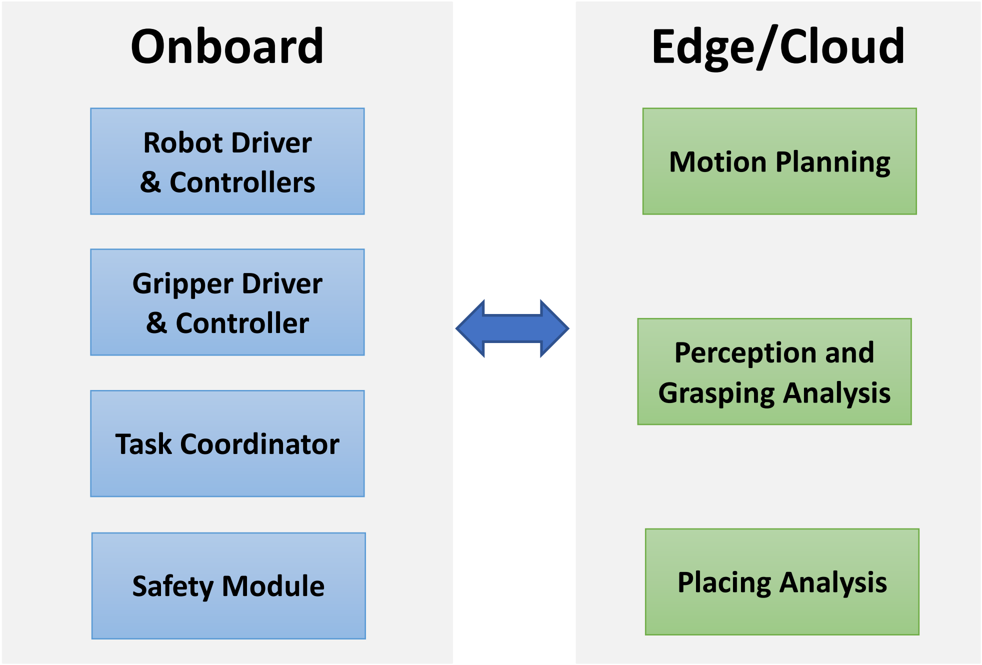
- Each module is run in an isolated container
- Manageable hardware via ClusterInventory
- Declarative software modules in ApplicationDeployment



Case Study A: Pick and Place with Manipulators

Classic Setup

- Setup networking
- Configure DDS domain ID
- Set namespace for multi-robots
- Bash-script to start different processes

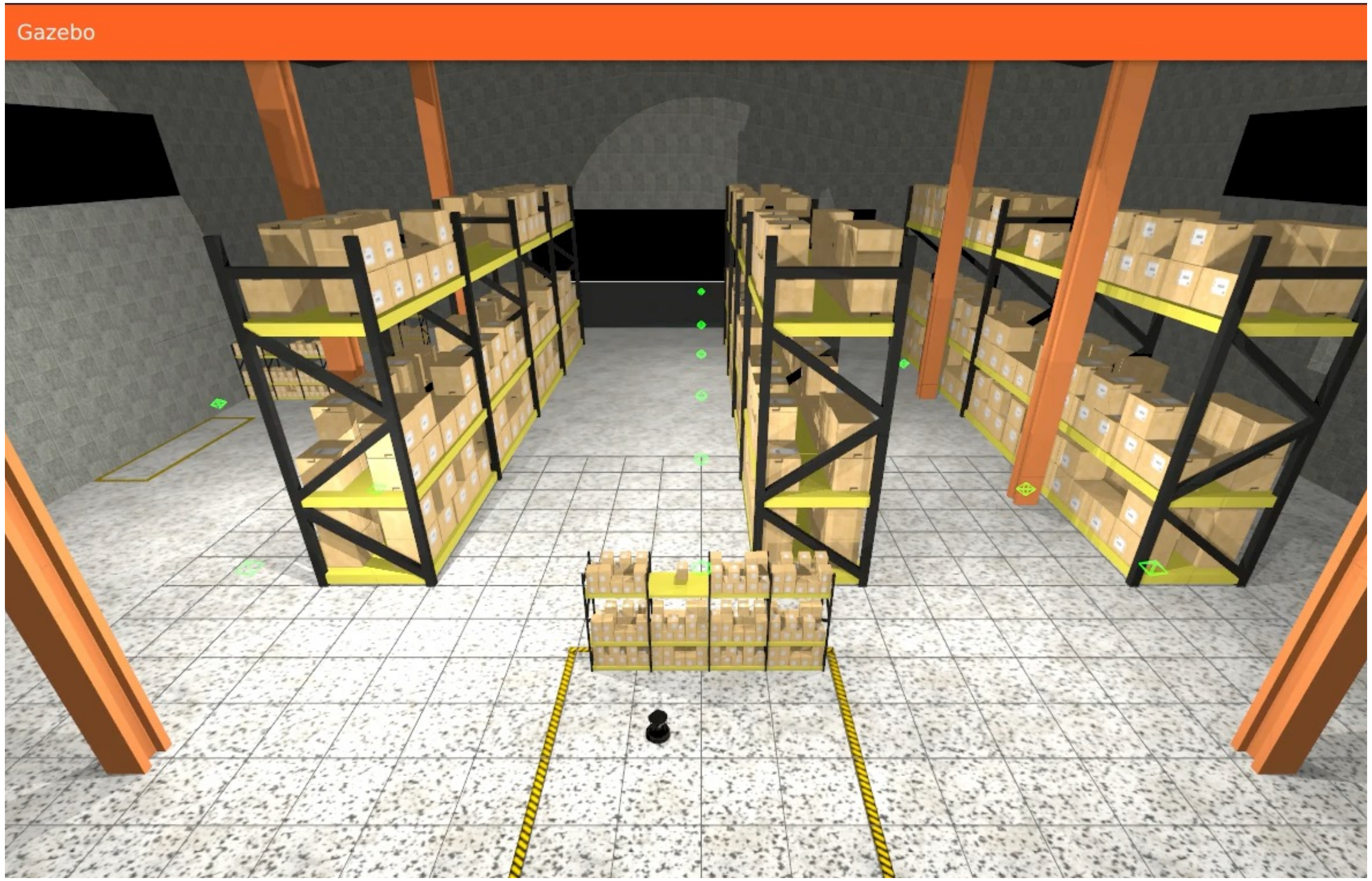


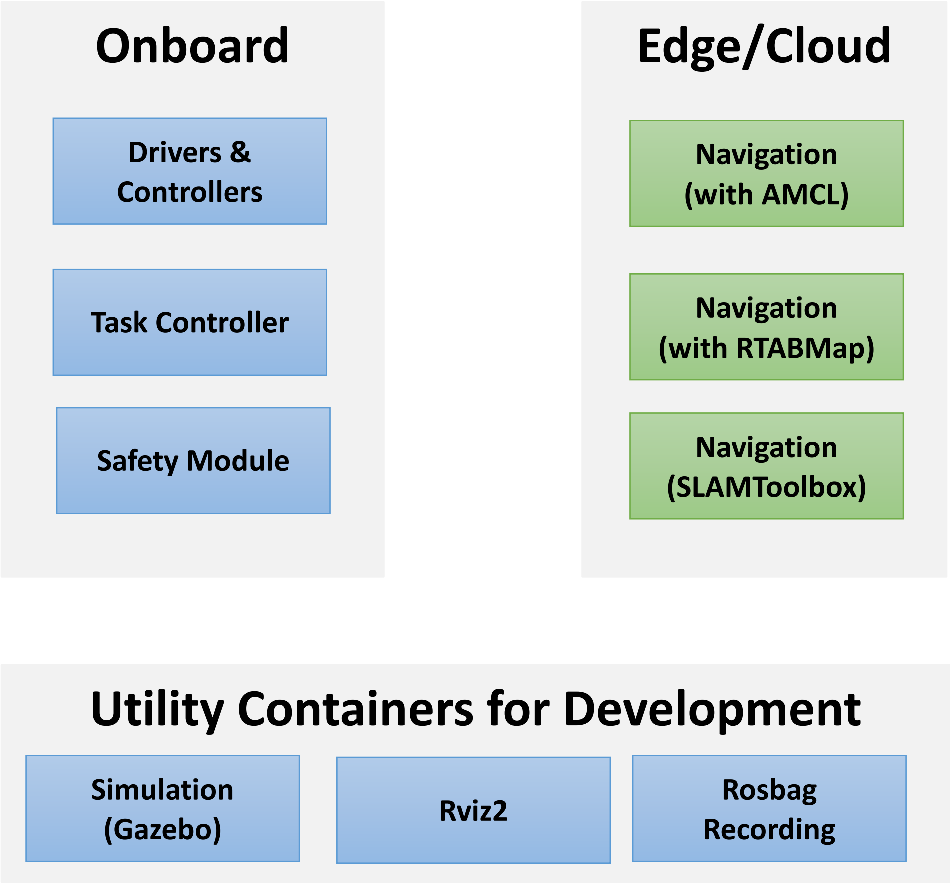
Onboard
Robot Driver & Controllers
Gripper Driver & Controller
Task Coordinator
Safety Module
Edge/Cloud
Motion Planning
Perception and Grasping Analysis
Placing Analysis

Advantages with KubeROS

- Easy access to the edge/cloud
- Flexibility and adaptability
- Scalability
- Reusability of modules
- Simplified launch files
- Reproducible deployment

Case Study B: Navigation of Mobile Robots





Onboard

Drivers & Controllers

Task Controller

Safety Module

Edge/Cloud

Navigation (with AMCL)

Navigation (with RTABMap)

Navigation (SLAMToolbox)

Utility Containers for Development

Simulation (Gazebo)

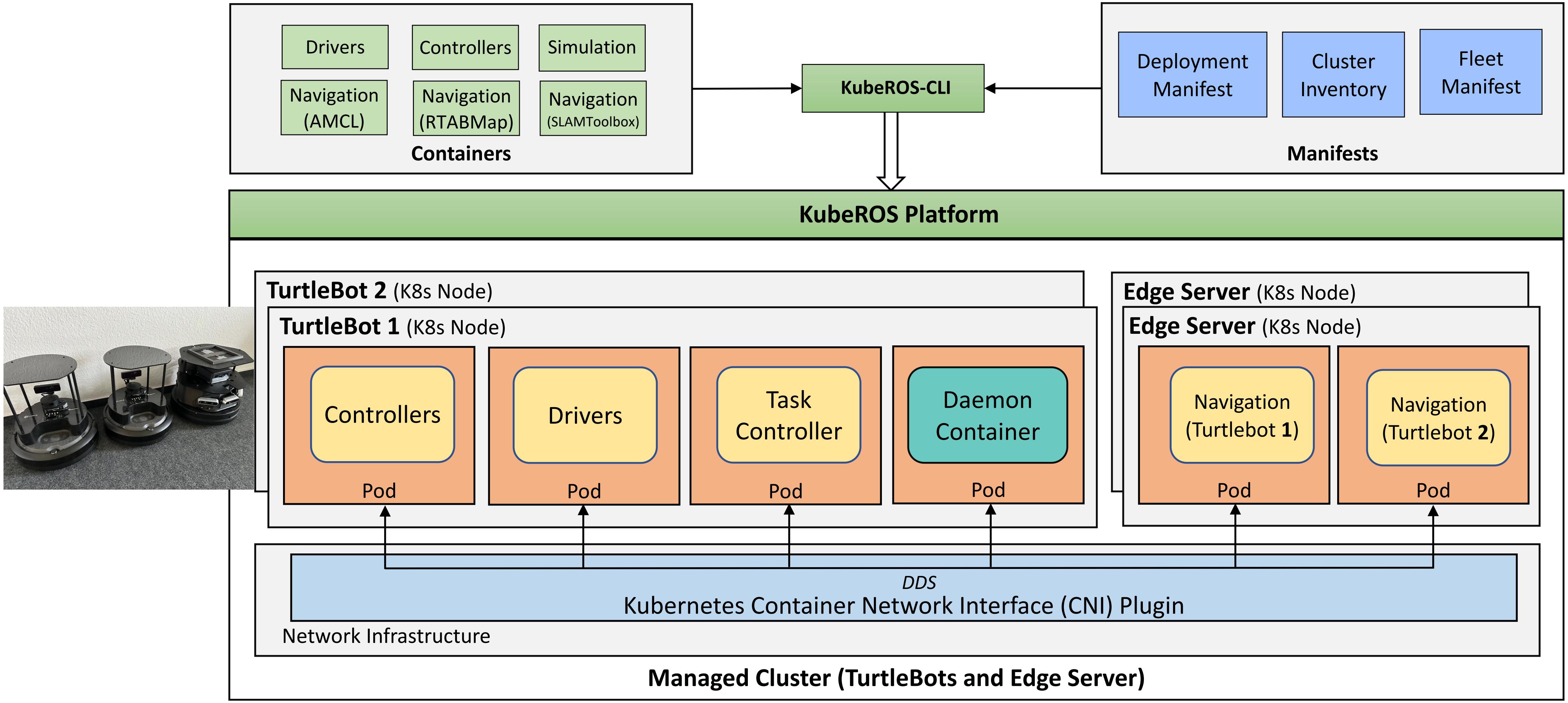
Rviz2

Rosbag Recording

Contributors: Frederik Pasch; Florian Mirus

https://github.com/kuberos-io/kuberos-nav-slam

Case Study B: Navigation of Mobile Robots



Drivers

Controllers

Simulation

Navigation (AMCL)

Navigation (RTABMap)

Navigation (SLAMToolbox)

Containers

KubeROS-CLI

Deployment Manifest

Cluster Inventory

Fleet Manifest

Manifests

KubeROS Platform

TurtleBot 2 (K8s Node)

TurtleBot 1 (K8s Node)

Controllers

Pod

Drivers

Pod

Task Controller

Pod

Daemon Container

Pod

Edge Server (K8s Node)

Edge Server (K8s Node)

Navigation (Turtlebot **1**)

Pod

Navigation (Turtlebot **2**)

Pod

DDS

Kubernetes Container Network Interface (CNI) Plugin

Network Infrastructure

Managed Cluster (TurtleBots and Edge Server)

Contributors: Frederik Pasch; Florian Mirus

https://github.com/kuberos-io/kuberos-nav-slam

Summary

Benefits with KubeROS:

- Simplify access to the edge or the cloud
- Improved software flexibility, adaptability, und maintainability
- Enable large-scale deployment
- Easy to use (after setup)

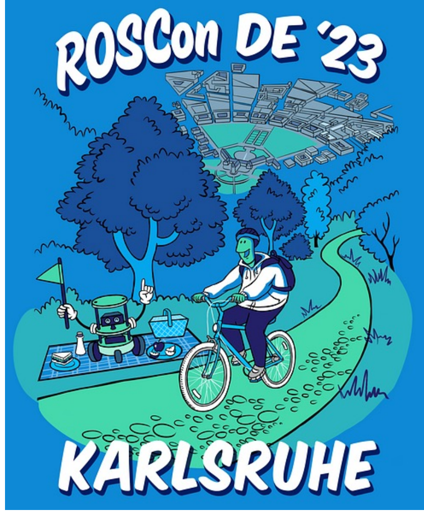
Problems:

- Kubernetes setup requires experience
- A general hardware interface
- Application decomposition and containerization (granularity)

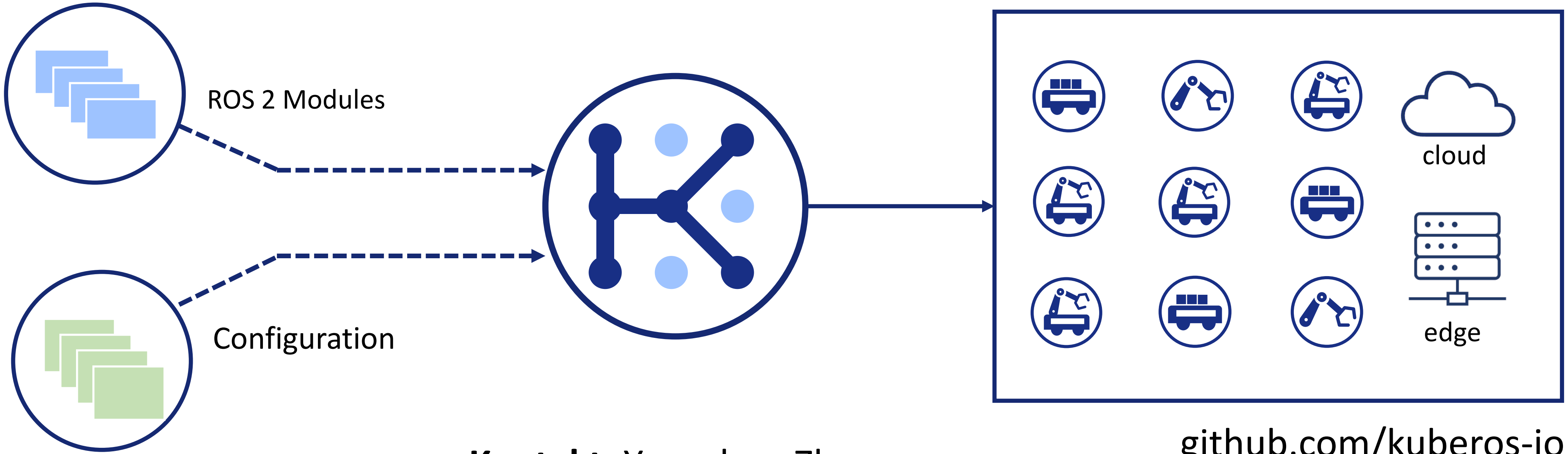
KubeROS status:

- Under active development [Prototype]
- Parts of the code available in GitHub: github.com/kuberos-io

KubeROS



Vielen Dank!



Kontakt: Yongzhou Zhang
yongzhou.zhang.d@gmail.com

github.com/kuberos-io

Acknowledgment: parts of this work are conducted as part of the project **KI5GRob** funded by German Federal Ministry of Education and Research (**BMBF**).

Hochschule Karlsruhe
University of
Applied Sciences

HKA

KIT
Karlsruhe Institute of Technology