November 2025

NODE Robotics

Experience and Learnings from Migrating Our Navigation Stack from ROS1 to ROS2

Speaker: Philipp Schnattinger

ROSCon FR & DE 2025 - Keynote

NODE Robotics

Fraunhofer IPA spin-off, established in 2020 and headquartered in Stuttgart, boasts a dynamic team of 25 dedicated employees.

Our remarkable journey has already seen us deploy our cutting-edge software to power more than 1,500 mobile robots, revolutionizing various industries with their productive applications.

intel ignite · SynerLeap Powered by ABB · VDMA · eXIST · NVIDIA INCEPTION PROGRAM · Fraunhofer · CyberValley · ARENA2036

We make mobile robots easy to use.

NODE OS

Autonomy Software for AGVs and AMRs

Product Family

NODE Fleet Autonomy Services

Fleet services transforming single robots into autonomous & collaborative fleets

Modules

- **NODE.maps** [Map Synchronization & Fusion]
- **NODE.flow** [Fleet Coordination & Routing]
- **NODE.jobs** [Fleet Task Execution]
- **NODE.analytics** [Fleet Performance Analytics]

- Plug and Perform
- Proven Reliability
- High Modularity
- Zero Hardware Modifications

Modules

- **NODE.localize** [Localization & Mapping]
- **NODE.move** [Hybrid Navigation & Docking]

Product Family

NODE Robot Autonomy Skills

Robot-level autonomy stack for enabling autonomous operation of the robots

Proven Deployment on 1,500+ Robots in Europe, Asia, and North America

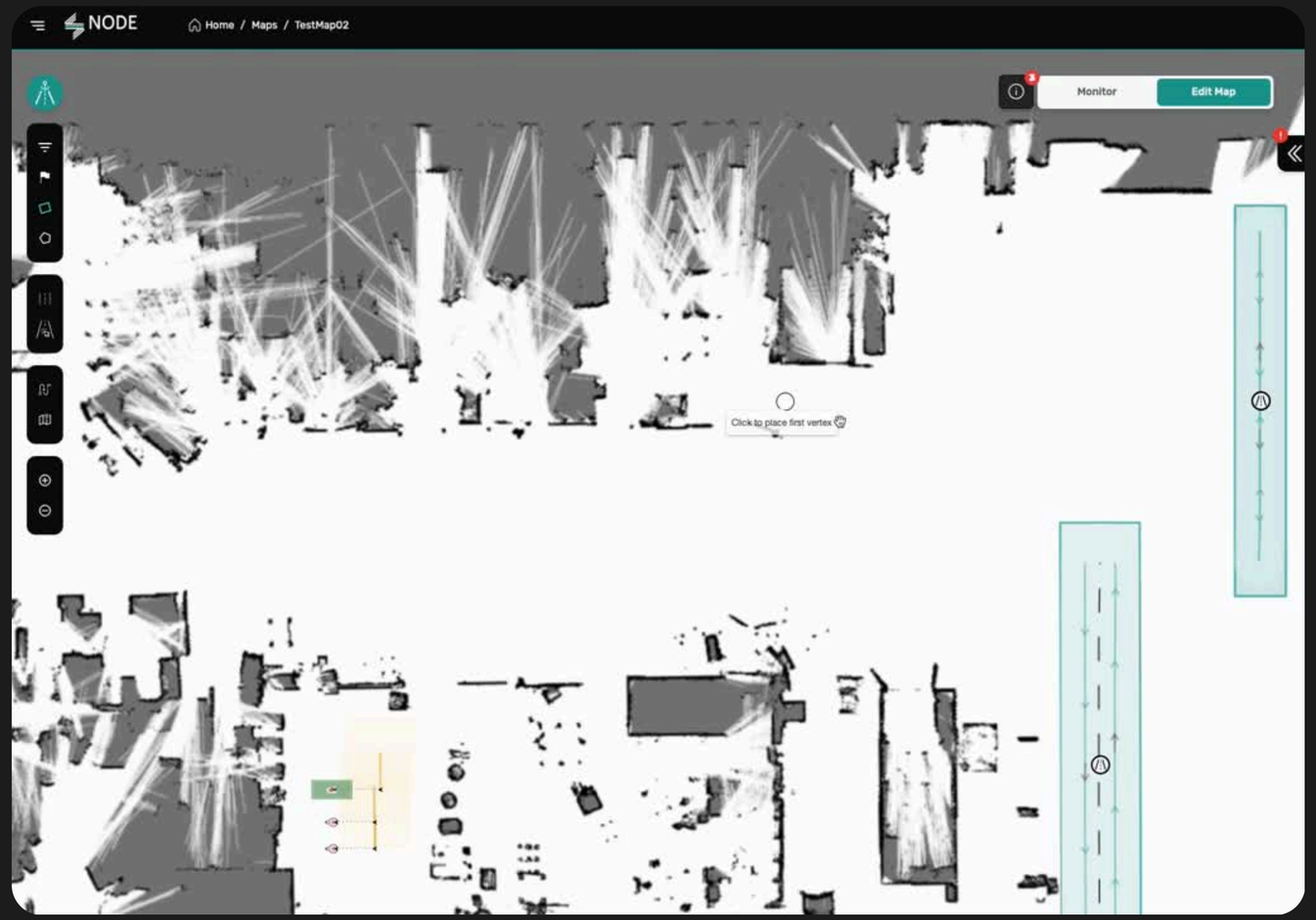
30+ Mobile Robot Types Supported with NODE.OS

Use Cases in Manufacturing, Logistics, and Service Robotics

Hybrid Navigation - Automated Road Map Generation I

Automating the setup phase:

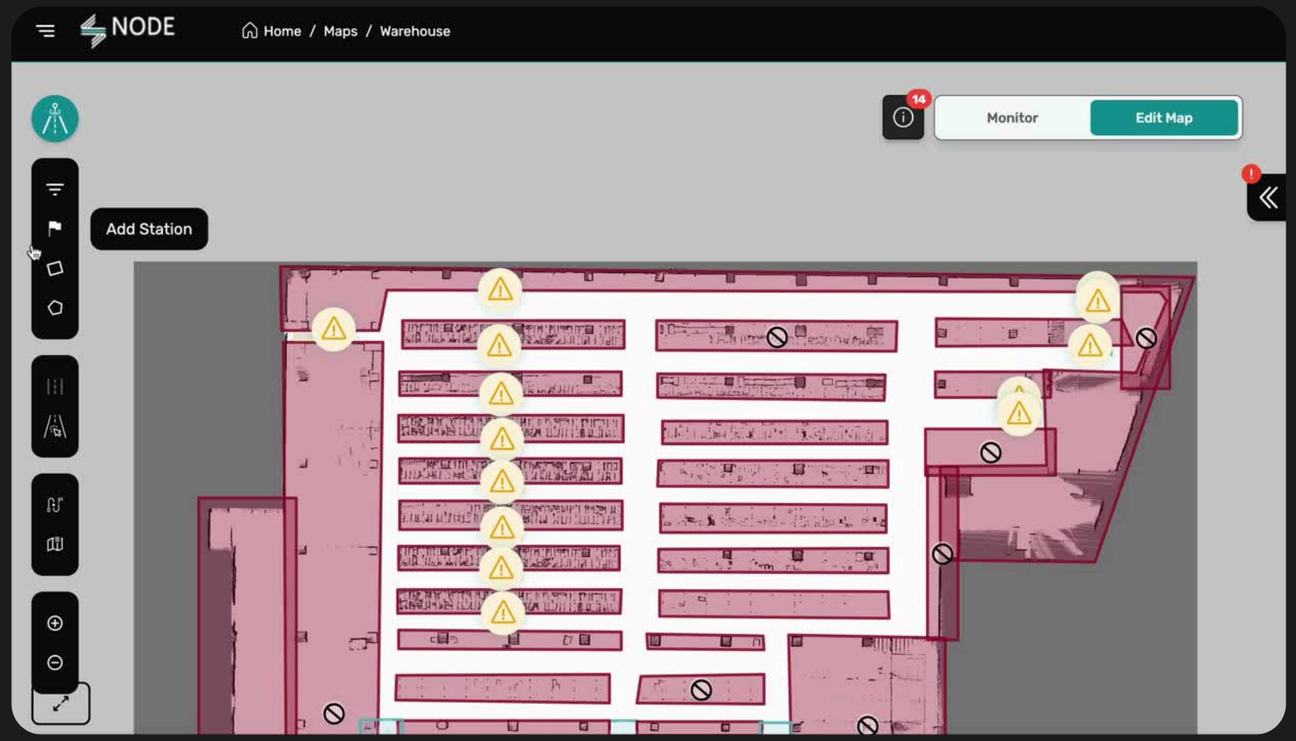
- Based on SLAM Map, Point Cloud, or Layout Map
- Considers Robot Properties: Kinematics, Footprint, Safety Fields
- Configurable Traffic Rules (One-Ways, Driving Side, etc.)



Hybrid Navigation - Automated Road Map Generation II

Automating the setup phase:

- Based on SLAM Map, Point Cloud, or Layout Map
- Considers Robot Properties: Kinematics, Footprint, Safety Fields
- Configurable Traffic Rules (One-Ways, Driving Side, etc.)



Agenda

01 ROS”1" Navigation
02 ROS2 - Let’s make things better
03 DDS - Are we too dumb or is it a problem

NODE Stack Evolution Over Time

2013

Starting as Fraunhofer IPA Research team in mobile robot navigation

2015

First SLAM-based AGV in automotive assembly at AUDI plant

Credit: Bär Automation

2018

BMW collaboration and first rollout of BMWs ‘STR’ running with NODE navigation

Credit: BMW Group

2020

Incorporation of NODE Robotics in Stuttgart

NODE

2021

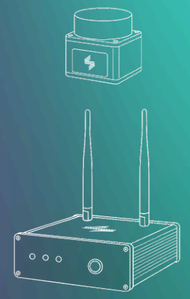
Make Robots easy to use and add our own UI

2024

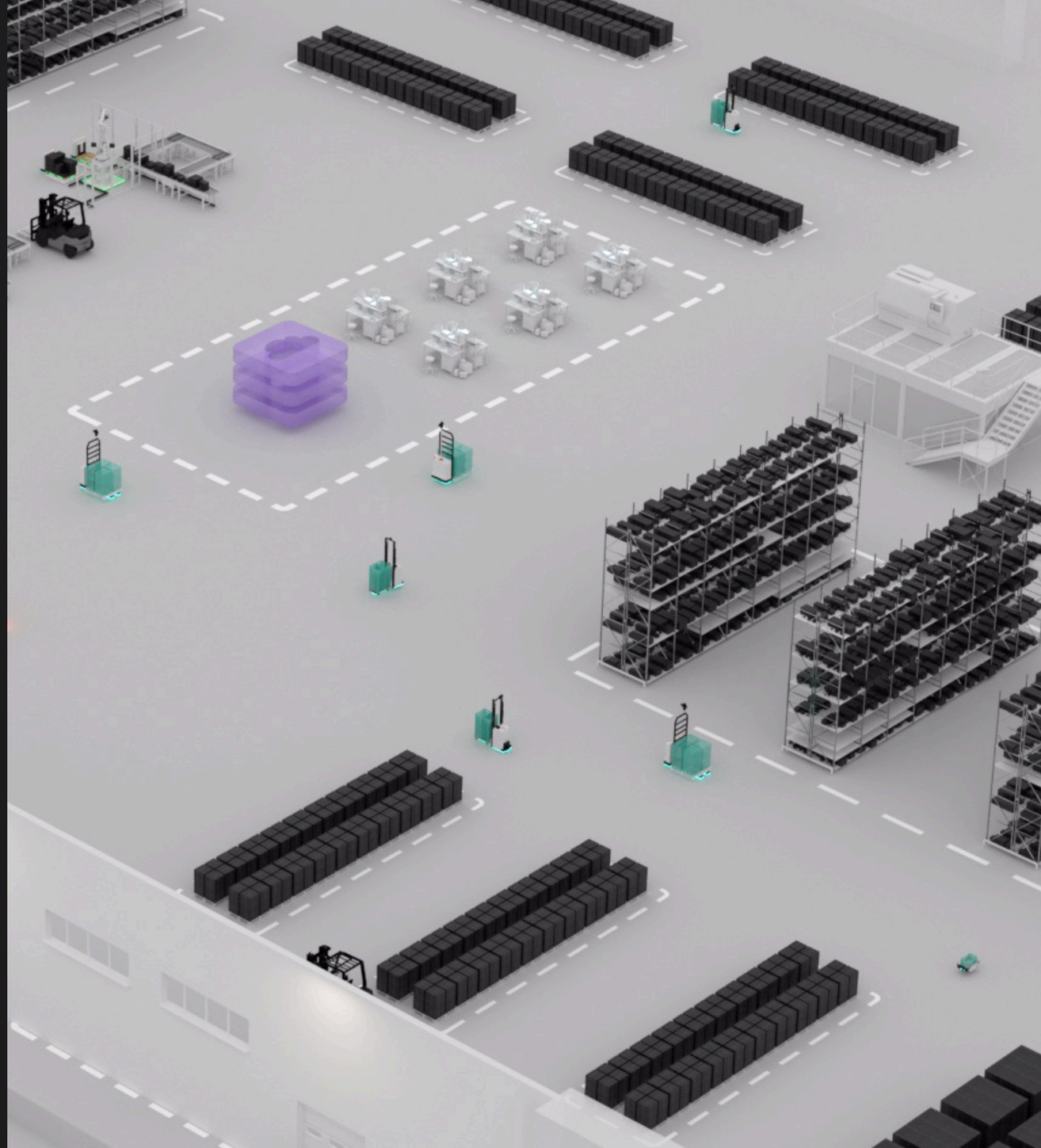
Running everything in Jazzy

2025

Let’s do - hardware NODE.tracking



ROS"1" Navigation



Starting Point :

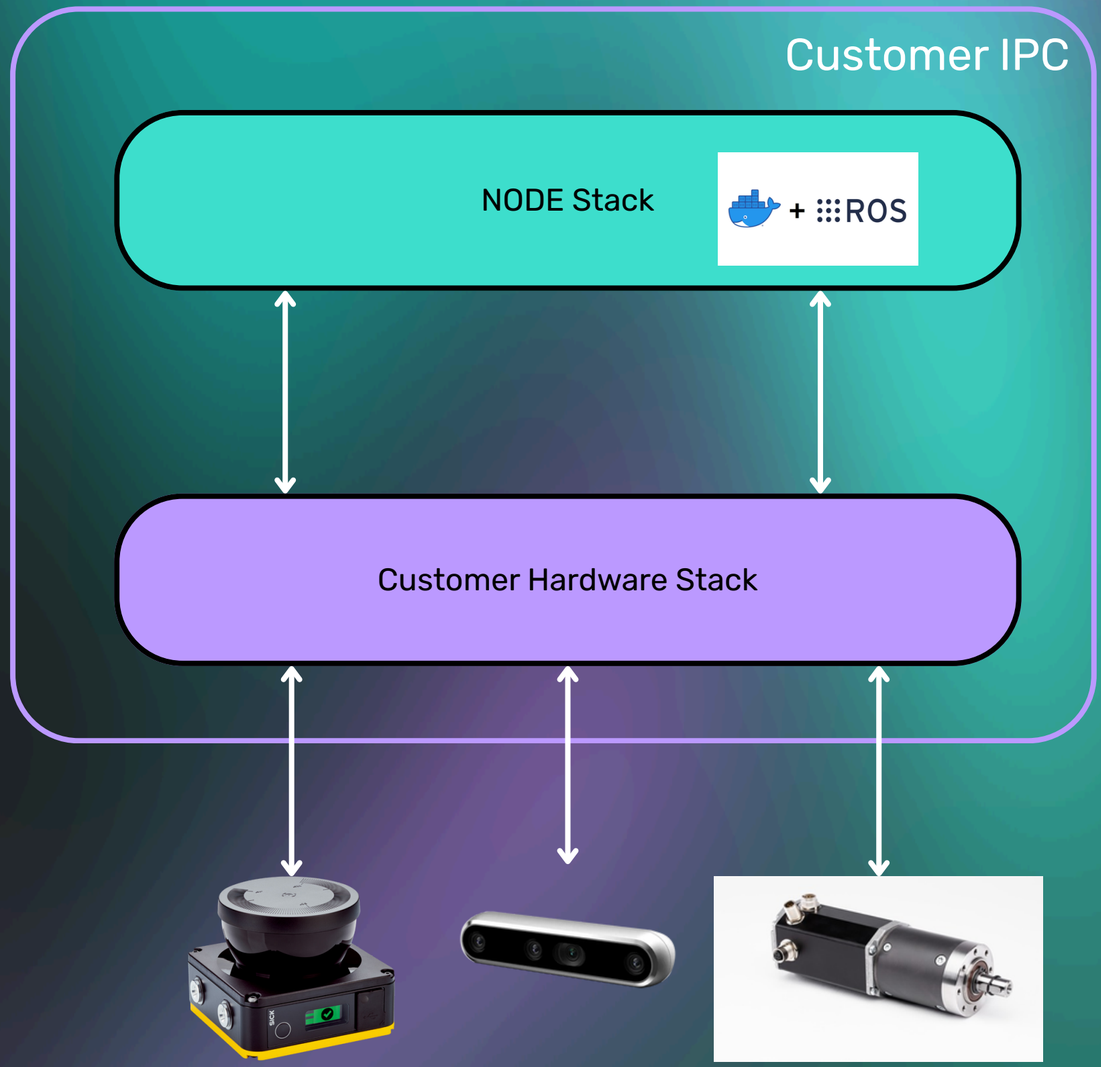
Just a software stack,
no hardware

Tools We Utilize

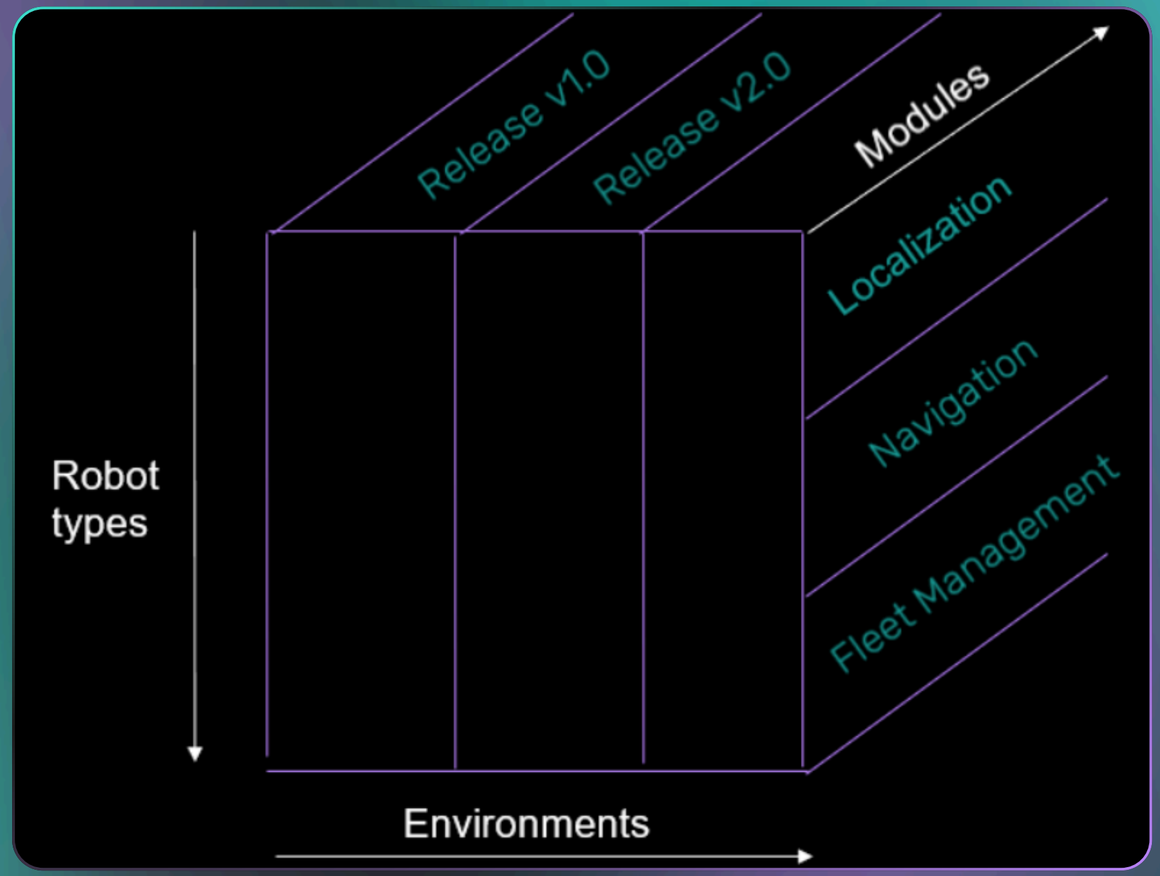
- Docker and docker compose for deployment
- Custom-developed provisioning tools

Challenges:

- Communication between distributions
- Reliance on the quality of customer sensor data
- Lack of control over the hardware layer on the ROS side
- Variability in PC setups



Software stack scope

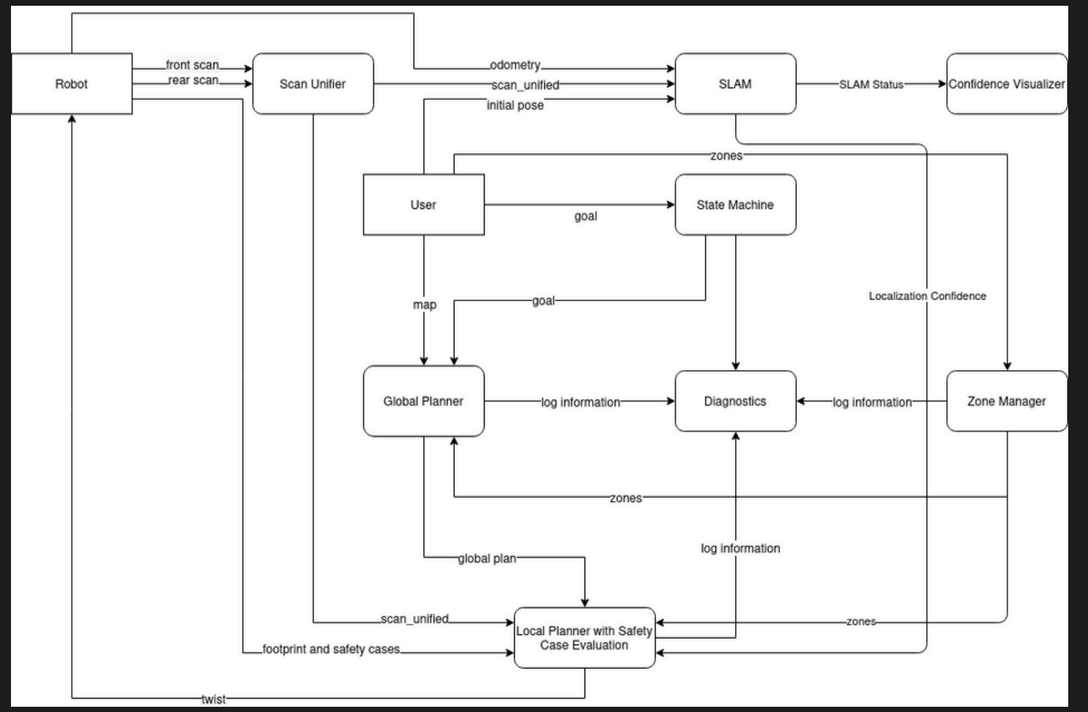


ROS"1" navigation state

- Interconnection to the stack is easy
- One Distro to rule them all and Docker
- Discovery goes through roscore
- Grown over time
- Across ROS Distro Support possible

NODE ROS1 Stack

Customer ROS1 Stack



How to combine ROS with other interfaces and technology

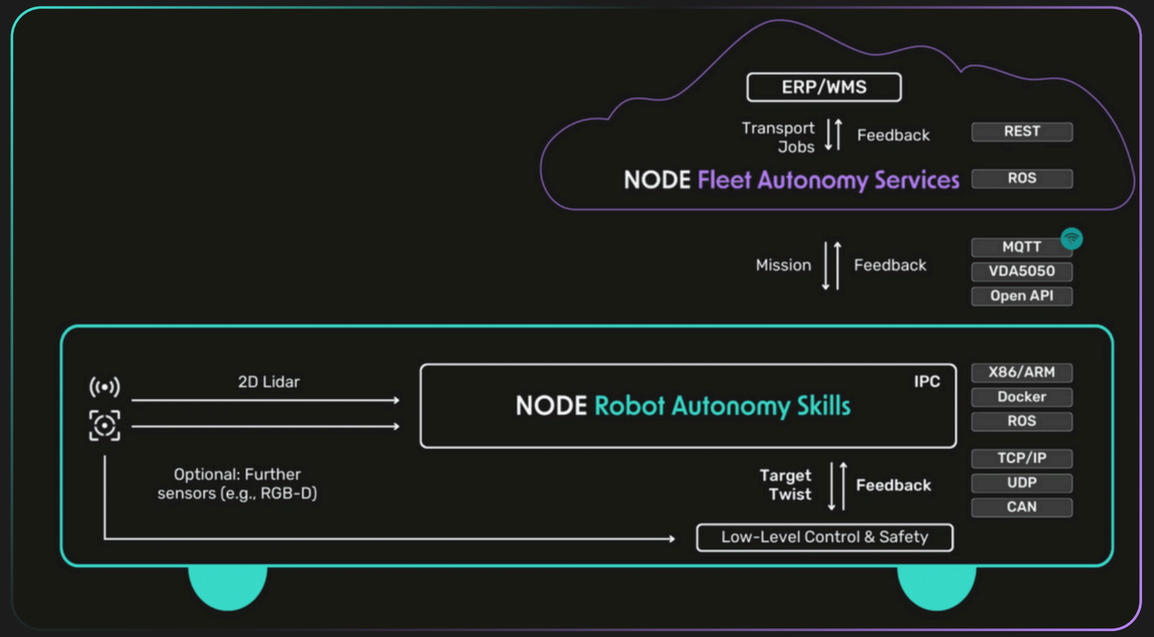
We need to add a UI to our Navigation stack for non - ROS Users

We come from ROS - all of the sudden we need to include other APIs

Continiously growing construct over time

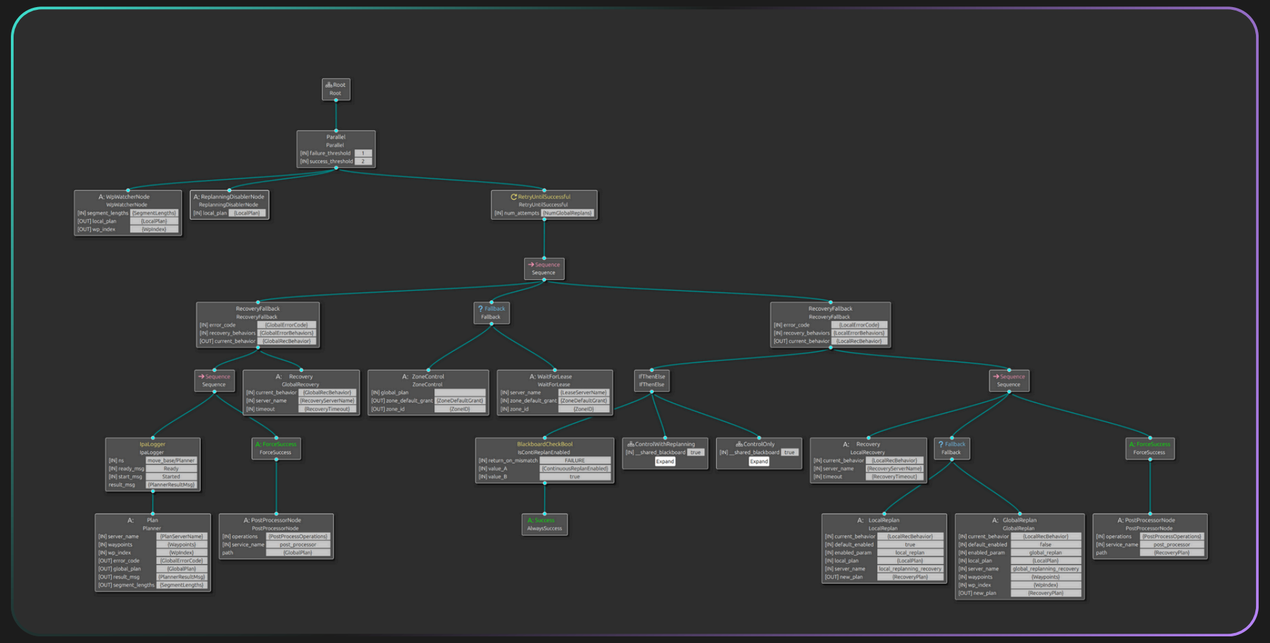
Who is aggregating what? - Diagnostics, VDA5050 Information

Let’s do what we know. Make ROS and wrap things around



Grown software stack over time in ROS1

What is the best state machine tooling for navigation
- Switching from Smach to Behavior Trees



VDA5050 Integration - Runtime Generation of Behavior Trees

We came late to Behavior Trees

Nobody spoke about docking, multiple planners different navigation typse.

VDA5050 actions, docking and what we call skills

How can we incorporate custom or complex behavior that doesn't simply involve getting from point A to B?

Utilize a custom YAML format

Integrate runtime-generated behavior trees

Incorporate elements that are challenging to visualize and comprehend

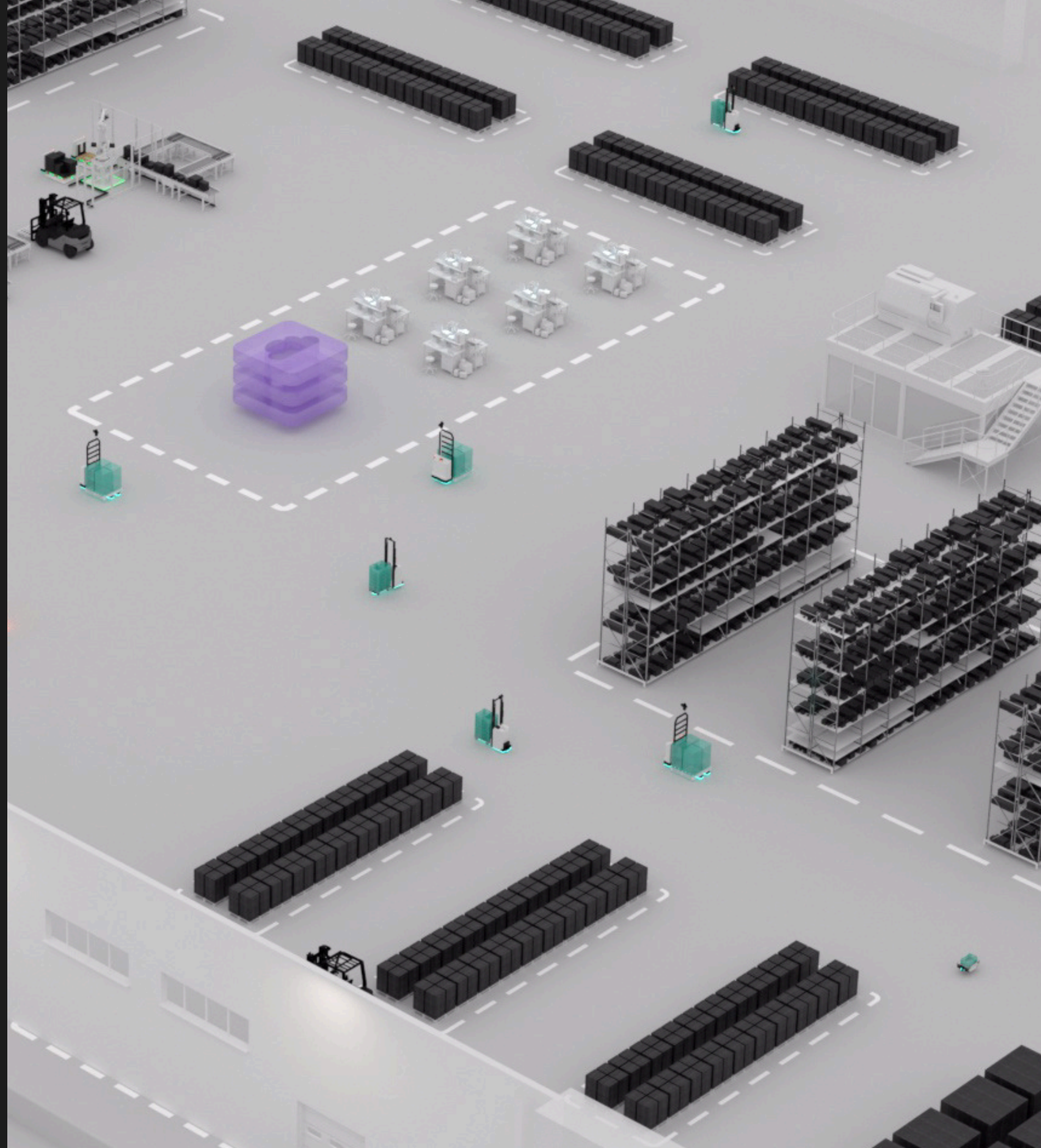
Hard to read and maintain but cool once you have set it up

```yaml
move_out_of_line: sequence
move_out_of_line_params:
  actions:
    - mute_safety
    - lift_up_line
    - move_forward_pick_up
    - unmute_safety
    - lift_up

pickup_line_fallback: fallback
pickup_line_fallback_params:
  actions:
    - pickup_line_and_move_out_of_line
    - move_out_of_line_and_fail

pickup_line_and_move_out_of_line: sequence
pickup_line_and_move_out_of_line_params:
  actions:
    - pickup_line
    - disable_addon_ocf
    - mute_safety
    - lift_up_line
    - move_forward_pick_up
    - unmute_safety
    - enable_addon_ocf
    - lift_up

move_out_of_line_and_fail: force_failure
move_out_of_line_and_fail_params:
  actions:
    - move_out_of_line_with_lowered_fork

move_out_of_line_with_lowered_fork: sequence
move_out_of_line_with_lowered_fork_params:
  actions:
    - disable_addon_ocf
    - mute_safety
    - move_forward_pick_up
    - unmute_safety
    - enable_addon_ocf
    - lift_down
```

Summary of ROS"1"

- Proven stack in production over years - still running
- Connectivity and integration is easy with the ROS“1”
- Some design decisions might be not that great
- No Lifecycle Management
- Custom Behavior based on Yaml and Behavior Trees

ROS2 Migartion and new Challenges



Why did we actually migrate?

- EOL of ROS Noetc
- Cybersecurity Risks
- New features out there which we want to add
- Improved design and capabilities

ROS2 journey so far

Let’s setup a branch and build it

Make all the packages as COLCON IGNORE

Migrate the libraries and delete a lot of things

NAV2 interface updates - early adopt to Iron

Running everything in Jazzy

Problems in the beginning

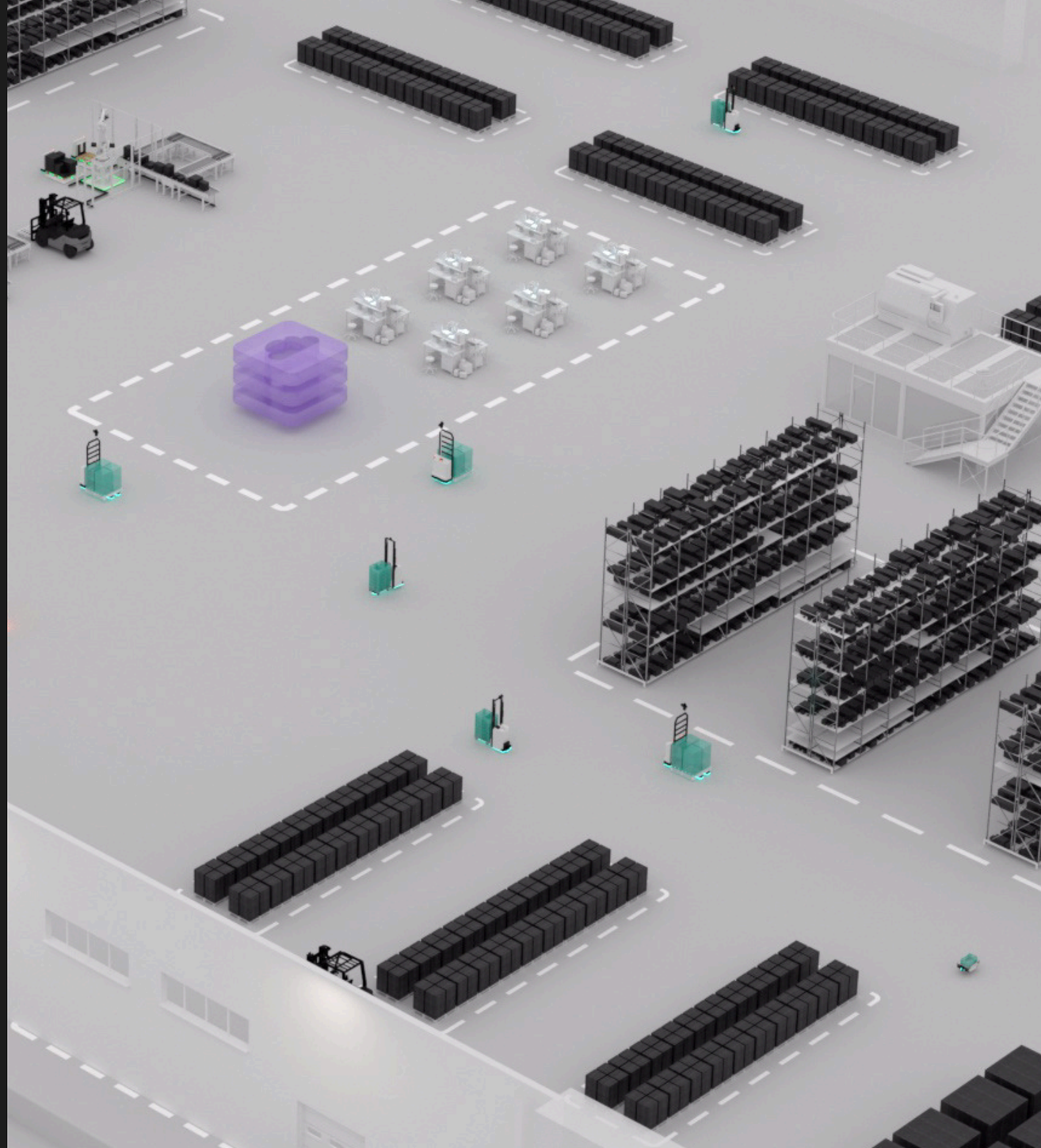
Early Adopter Problems - not stable, API breaking changes, missing understanding of ROS 2

No clue about how DDS works - still the case

Most customer still runs ROS”1" stack.

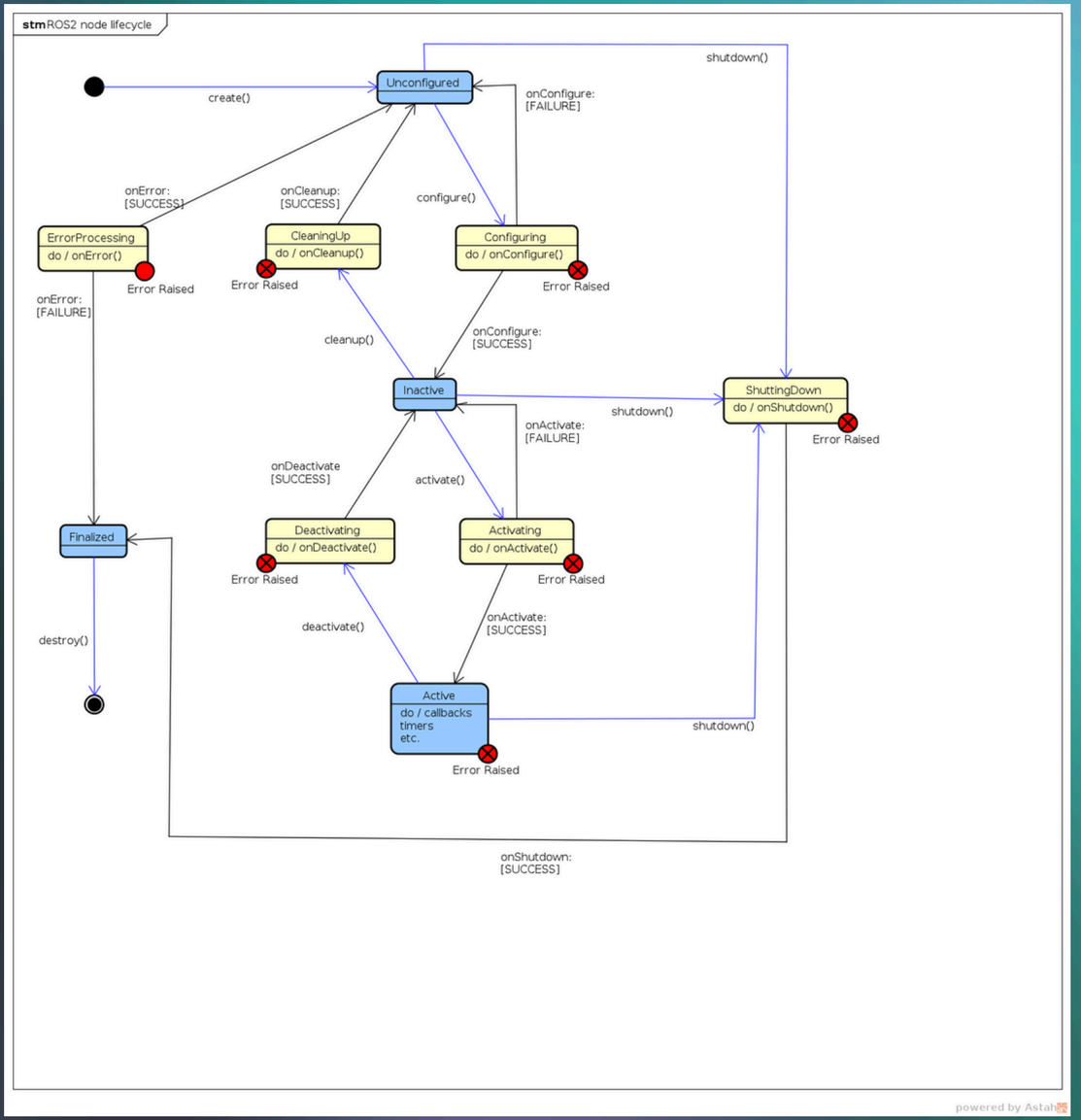
Quiet frequent updates needed from the latest state - we need to adopt new features early

New Design Decisions and Tooling



Lifecycle manager

- Introduced a new Lifecycle Manager
- Gained control over startup and states
- Implemented states for:
  - Mapping
  - Planning
  - Localization



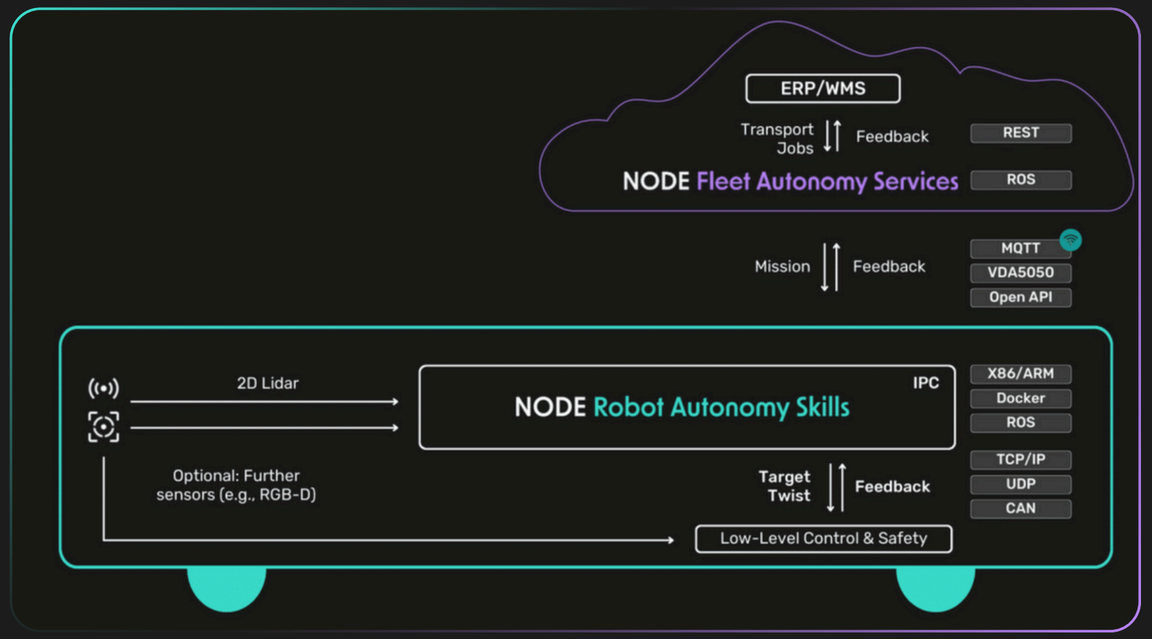
Unification of APIs and usability accross interfaces

Bring VDA5050 and REST APIs into main stack

REST API, VDA5050, ROS - we don’t care

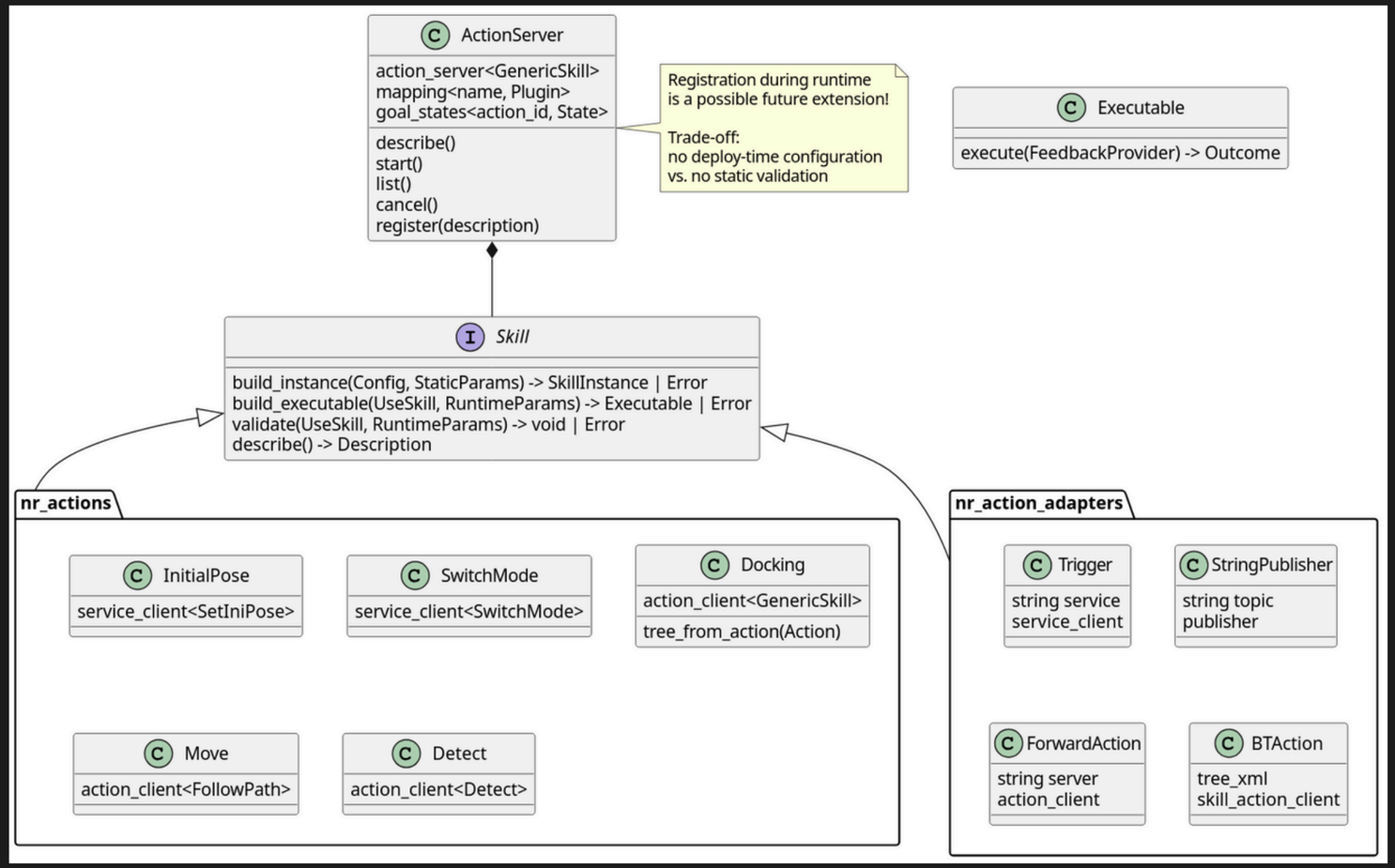
Reduced size and amount of images for deployment

Improved design and capabilities for customer APIs

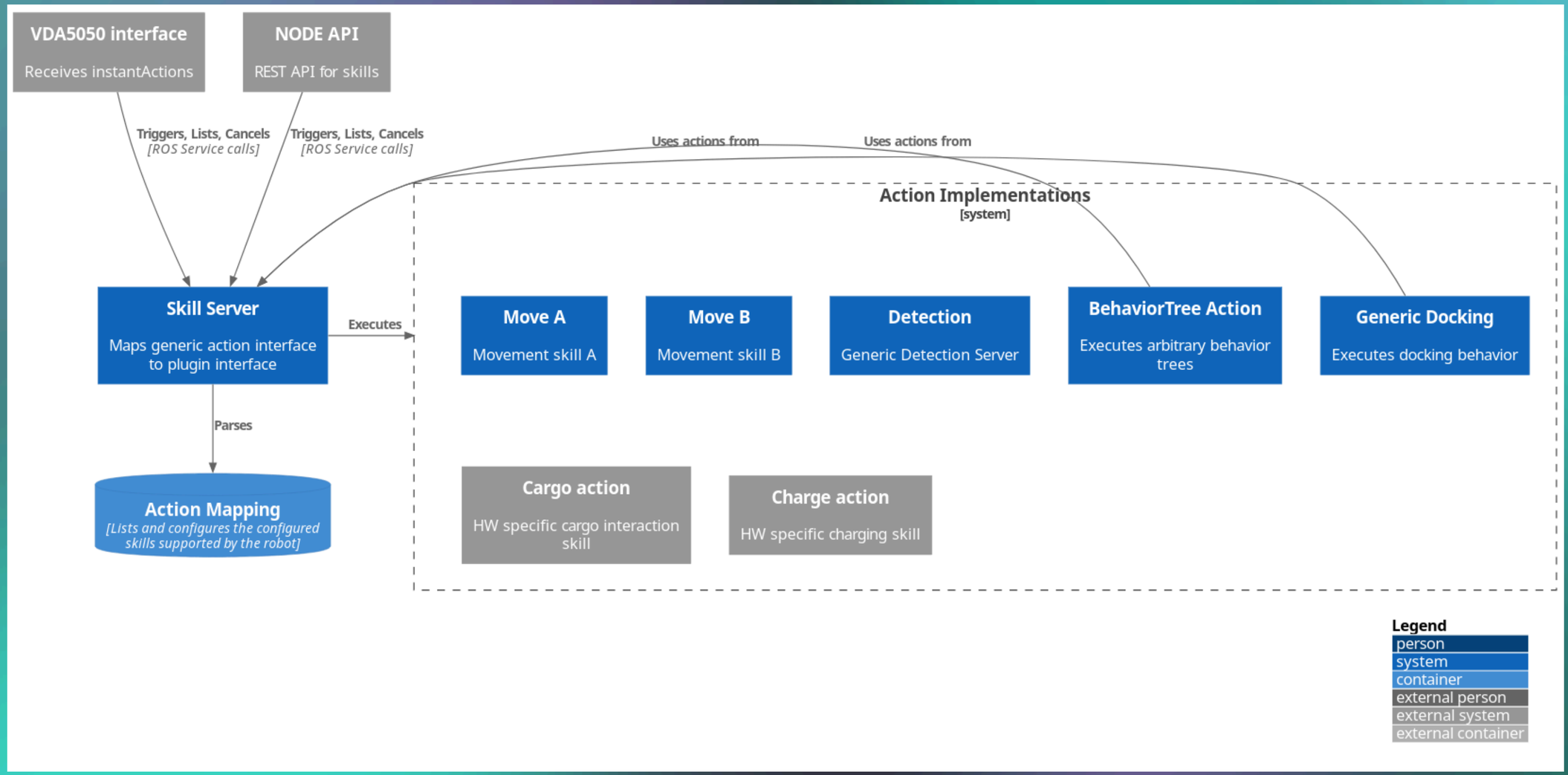


Skills and better generic interfaces to combine technologies

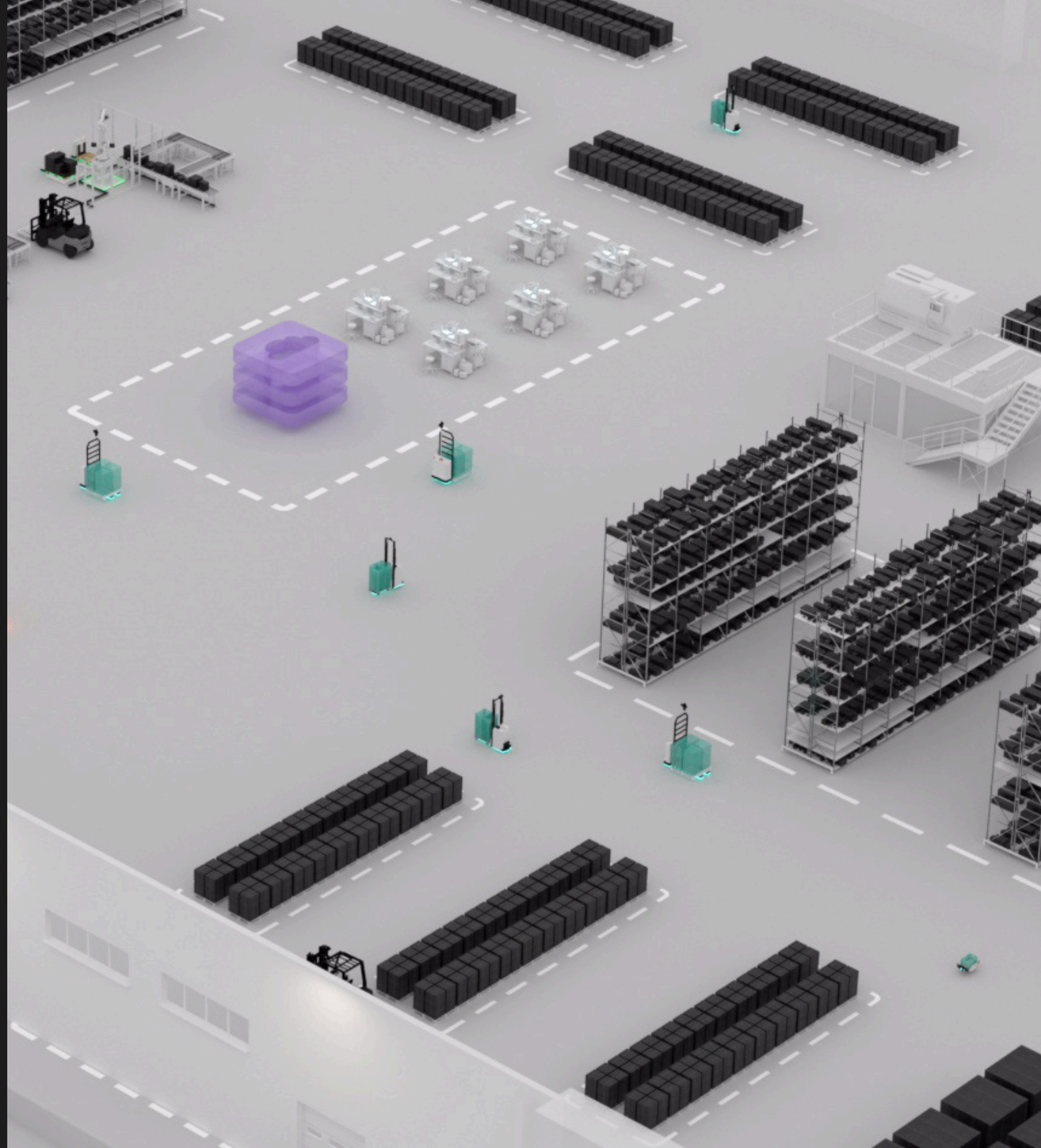
- One generic Action interface
- Gives customer possibility to write their own skill
- Easy to configure standard skills
- Visualize your skill with Groot



Skills and better generic interfaces to combine technologies



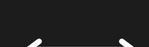
DDS -
Still a Problem



DDS setups - why so complicated

- Multiple middlewares make integration more complex
- Middleware configuration makes it even worse
- Switiching between “defaults” over ROS Distros
- Customers come with own or have not thought about it yet
- Increase in onboarding time and complexity

NODE Stack



ROS

Customer ROS2 Stack

DDS at NODE - What are we using

CycloneDDS is default RMW at the moment

Zenoh works for some use cases - for others
it also shows problems

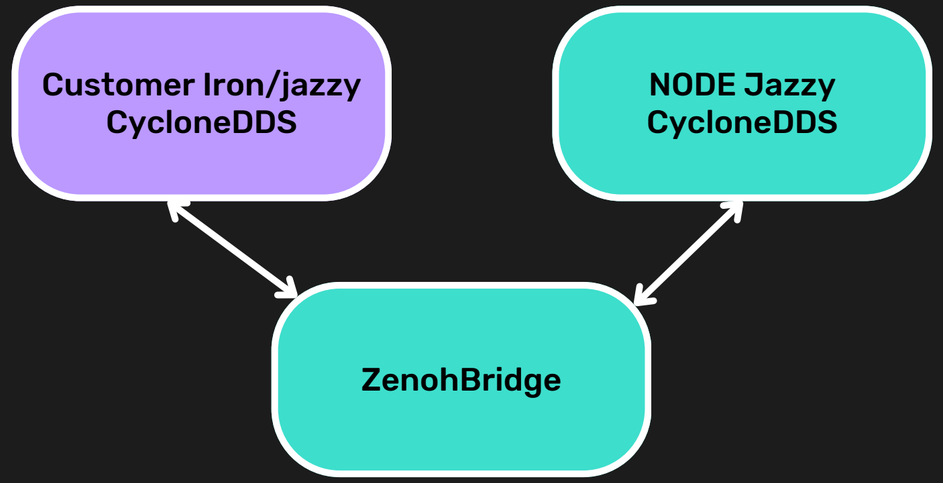
Zenoh bridge setup is also one solution we have running

Interdistro communication breaks CLI on some Distros

Middleware	NODE Distro - Customer Distro
CycloneDDS	Jazzy- Jazzy Jazzy - Iron Jazzy - Humble
CycloneDDS - Double Zenoh Bridge - CycloneDDS	Jazzy-Jazzy Jazzy - Iron
FastRTPS	Not used
Zenoh	Jazzy - Jazzy
CycloneDDS- ROS1_bridge	Jazzy - Kinetic Jazzy - Noetic
Zenoh - ROS1_bridge	Jazzy - Noetic

Discovery with ARM/NVidia seems to be problematic

- Discovery takes ages during startup
- Problems independent of middleware
- CycloneDDS not usable because discovery takes very long and leads to crashes
- Python launch files seem to be nice, but startup load is a thing.

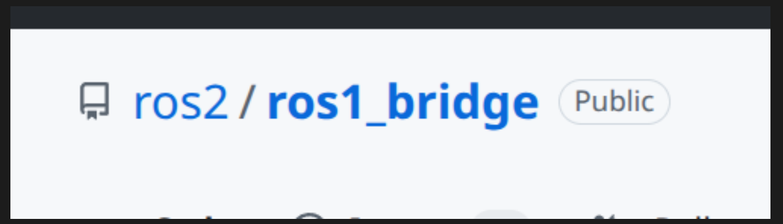


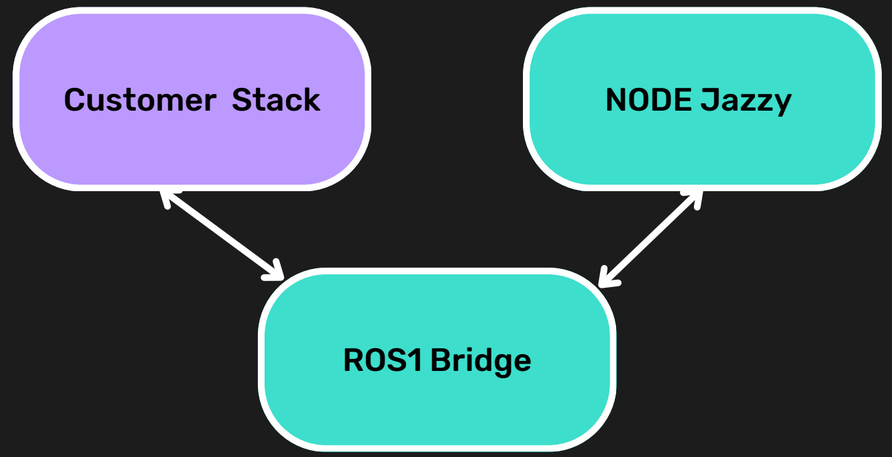
ROS2 and legacy

ROS2 Jazzy in combination with ROS1 legacy

Only works with hardware communication for tf, twist, odometry

No complex or custom messages





How to visualize my data?

What do we do if RViz is not working/possible?

Network does not allow multicast

Other setup often not possible

How to visualize my data? - Why not Zenoh Bridge it

CycloneDDS and RViz without multicast is a pain

Unicast setup possible but also nothing for fast debugging

Deploy zenoh bridges on the fly

```
# Set a fixed container name
CONTAINER_NAME="zenoh_bridge"
CYCLONEDDS_URI="'<CycloneDDS><Domain><Discovery><MaxAutoParticipantIndex>1000</MaxAutoParticipantIndex></Discovery><General><Interfaces><N

# Command to run on the SSH target
REMOTE_COMMAND="docker run -d --name $CONTAINER_NAME --rm \
  --network host \
  -e RMW_IMPLEMENTATION=rmw_cyclonedds_cpp \
  -e ROS_DOMAIN_ID=$ROS_DOMAIN_ID \
  -e CYCLONEDDS_URI=$CYCLONEDDS_URI\
  eclipse/zenoh-bridge-ros2dds:latest --no-multicast-scouting"

# Stop and remove any existing container with the same name on the remote SSH target, ignoring errors if not found
echo "Stopping and removing any existing container with the same name on the remote SSH target, ignoring errors if not found"
ssh "$SSH_TARGET" "docker ps -q --filter name=$CONTAINER_NAME | grep -q . && docker stop $CONTAINER_NAME && docker rm $CONTAINER_NAME || t

# Run the command on the remote SSH target
echo "Starting bridge on remote"
ssh "$SSH_TARGET" "$REMOTE_COMMAND"

# If the remote command succeeded, run the local Docker command
if [ $? -eq 0 ]; then
  # Stop and remove any existing local container with the same name, ignoring errors if not found
  docker ps -q --filter name=$CONTAINER_NAME | grep -q . && docker stop $CONTAINER_NAME && docker rm $CONTAINER_NAME || true

  LOCAL_COMMAND="docker run -d --name $CONTAINER_NAME --rm \
    --network host \
    -e RMW_IMPLEMENTATION=rmw_cyclonedds_cpp \
    -e ROS_DOMAIN_ID=$ROS_DOMAIN_ID \
    -e CYCLONEDDS_URI=$CYCLONEDDS_URI\
    eclipse/zenoh-bridge-ros2dds:latest -e tcp/$ROBOT_IP:7447"

```

How to visualize my data?

What do we do if RViz is not working/possible?

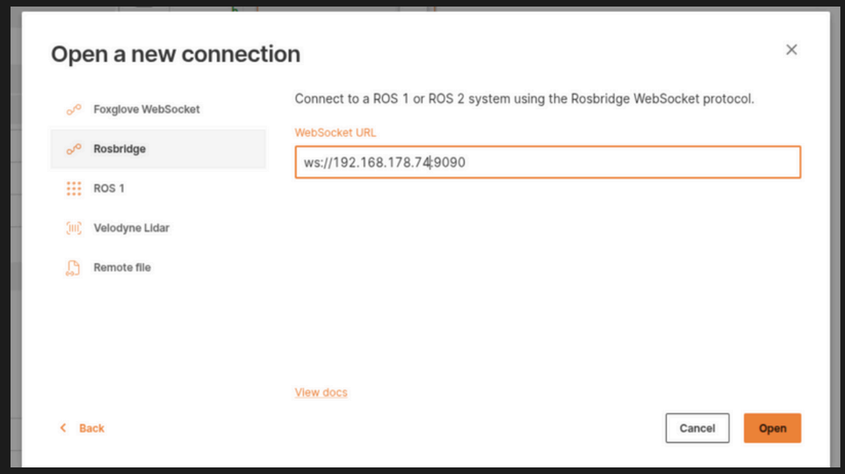
Lichtblick Suite:

https://github.com/lichtblick-suite/lichtblick

How to visualize my data?

Lichtblick

- Initially comes from Foxglove
- WebBased Visulaization
- Can be run with ROS2 Web Bridge



What have we learned?

Early Adoption is nice but way to production takes some time

Customer go their own way
We need to find ways to adapt

Robotics is more than ROS
Consider this in design decisions

Unification can give benefits in the long run

Thanks!

NODE Robotics

Website

LinkedIn

Philipp Schnattinger

Chief Robotics Officer

Open positions!

Join us and help make mobile robots easy to use.