

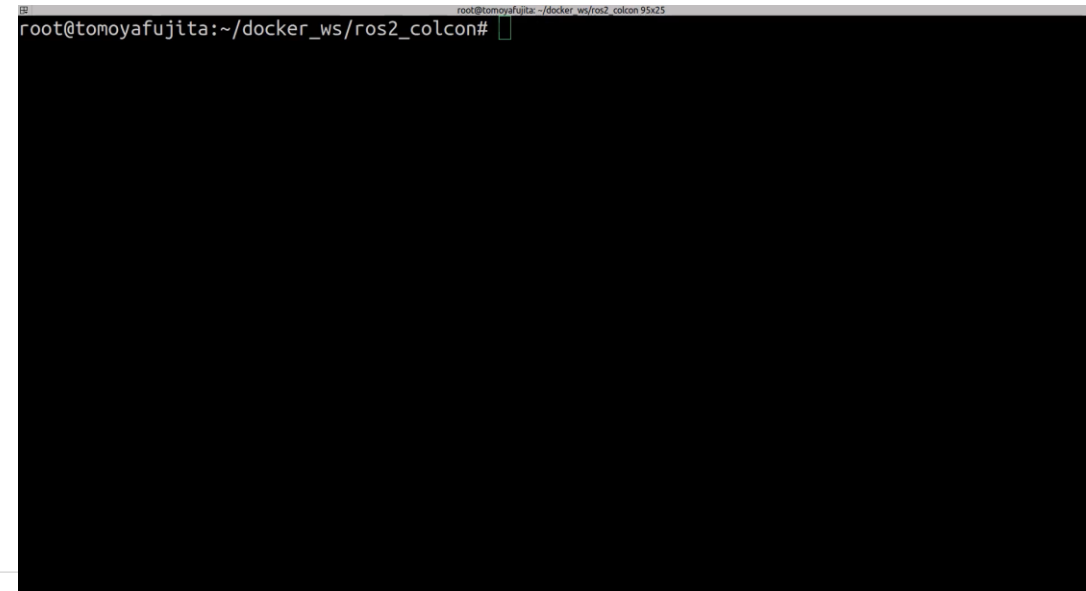
SONY

ROS 2 Logging Subsystem & rcl_logging_syslog

Tomoya Fujita
Sony Group Corporation

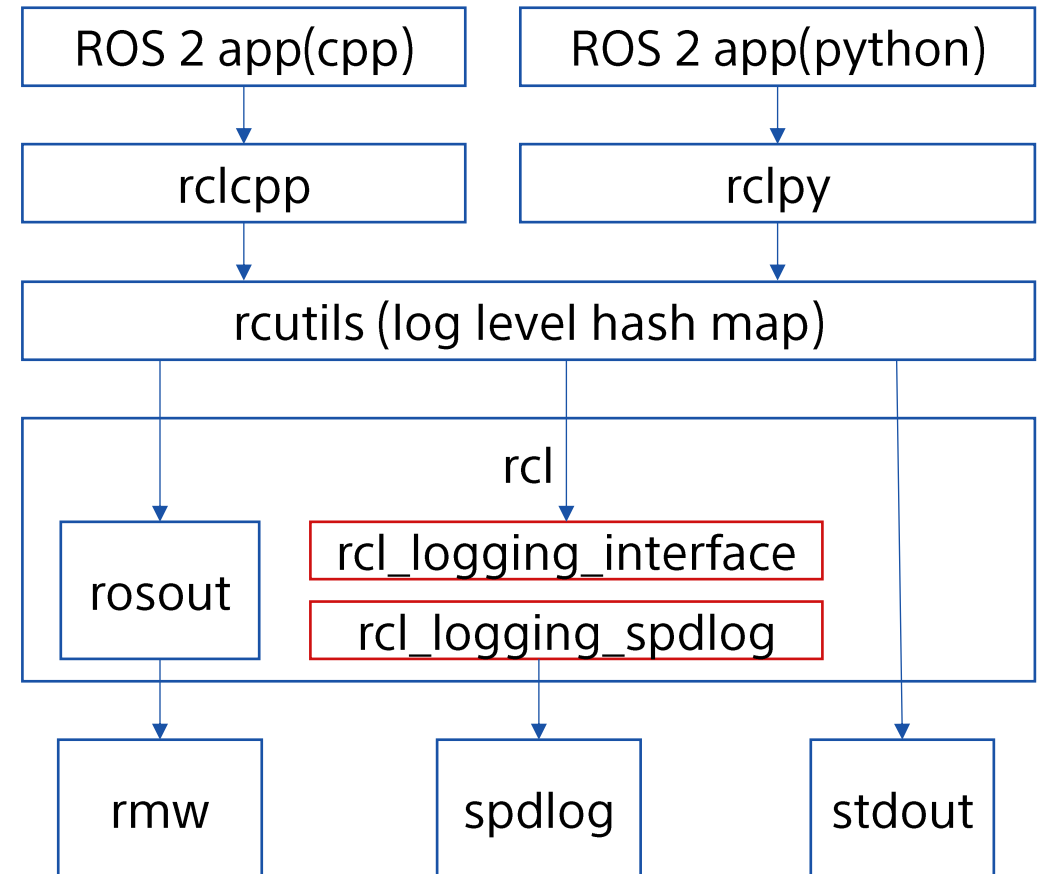
Who am I ?

- Tomoya Fujita [fujitatomoya@github](https://github.com/fujitatomoya)
 - [ROS Project Management Committee](#)
 - ROS 2 core maintainer
 - [OpenRobotics Google Summer of Code Mentor](#)
 - ROSCon PC/OC/EC etc...
 - [IEEE Robotics & Automation Practice Editorial Board](#)
 - ROSCon 2024 Odense
 - [ros2ai : next-gen ROS 2 CLI empowered by OpenAI and Ollama](#)
- KubeCon 2024
 - [Applying Cilium at Edge with KubeEdge](#)
- CNCF
 - [Edge Native Applications Principles Whitepaper](#)
 - [Edge Native Application Design Behaviors Whitepaper](#)



ROS 2 Logging Subsystem Overview

- ROS 2 provides a flexible and configurable logging subsystem.
- Supports multiple logging backends.
 - **stdout**: Logs are printed to the standard output (default ON to console)
 - **rosout**: Logs are published to the `/rosout` topic (default ON)
 - **external**: Default **spdlog**, or your own implementation.



Enabling and Disabling Logging Backends

- Node option in the program (e.g `rclcpp::NodeOptions`)
 - Be advised that `NodeOption` only support `rosout`
 - `rclpy` also supports the option to disable `rosout`.
- Global ROS arguments via CLI. (can be overridden by `NodeOptions`)

```
# Disabling rosout log publisher
$ ros2 run foo_pkg bar_exec --ros-args --disable-rosout-logs

# Disabling console output
$ ros2 run foo_pkg bar_exec --ros-args --disable-stdout-logs

# Disable any external loggers
$ ros2 run foo_pkg bar_exec --ros-args --disable-external-lib-logs
```

Logging Severity Level

- `DEBUG`, `INFO`, `WARN`, `ERROR` and `FATAL`.
- A logger will only process log messages with severity at or higher than a specified level chosen for the logger.
- Logger names represent a hierarchy.
 - If the level of a logger named `abc.def` is unset, it will defer to the level of its parent named `abc`, and if that level is also unset, the global default logger level will be used. When the level of logger `abc` is changed, all of its descendants (e.g. `abc.def`, `abc.ghi.jkl`) will have their level impacted unless their level has been explicitly set.

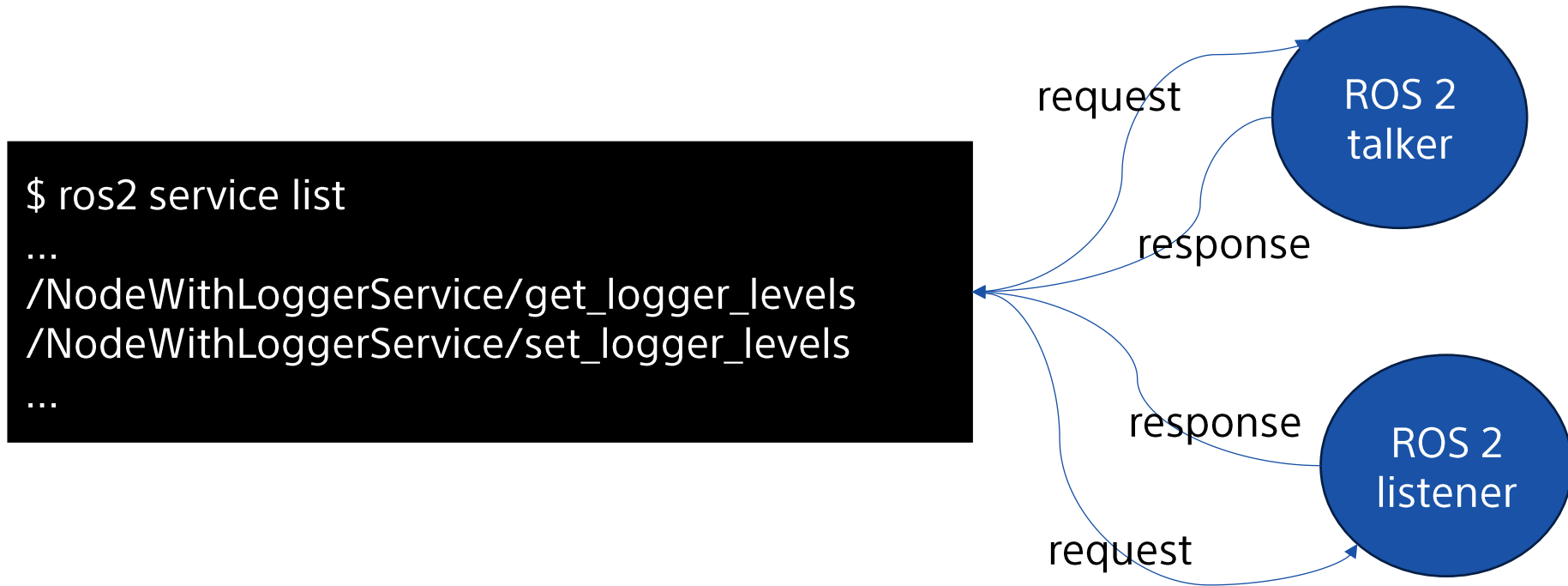
Set Logging Level via ros2run

```
# configures the default severity for any unset logger to the debug severity level
$ ros2 run logging_demo logging_demo_main --ros-args --log-level debug

# configures the severity for specific logger (node)
$ ros2 run logging_demo logging_demo_main --ros-args --log-level
logger_usage_demo:=debug
```

Logging Level Configuration Service

- It allows user to set the logging level of node at runtime.
- Use `rclcpp::NodeOptions().enable_logger_service(true)` (default false)



- See more details for [Logger level configuration: externally](#)

What else can we do?

- Change console [output format](#) and [color](#).
 - E.g) file name, line number, different format for timestamp...
- `ros2 topic echo /rosout` to see logging data in live
- [Set the log file name prefix](#).

```
...
stamp:
  sec: 1758182455
  nanosec: 534956691
level: 20
name: talker
msg: 'Publishing: "Hello World: 27"'
file:
/root/ros2_ws/colcon_ws/src/ros2/demos
/demo_nodes_cpp/src/topics/talker.cpp
function: operator()
line: 47
...
```

```
$ ros2 run demo_nodes_cpp talker --ros-args --log-file-name filename
```

- [Logging Environmental Variables](#)
 - `ROS_LOG_DIR`: Control the logging directory that is used for writing logging messages to disk.
 - `RCUTILS_LOGGING_USE_STDOUT`: Control what stream output messages go to. If this is unset or 0, use stderr. If this is 1, use stdout.
 - ...

Questions / Problems

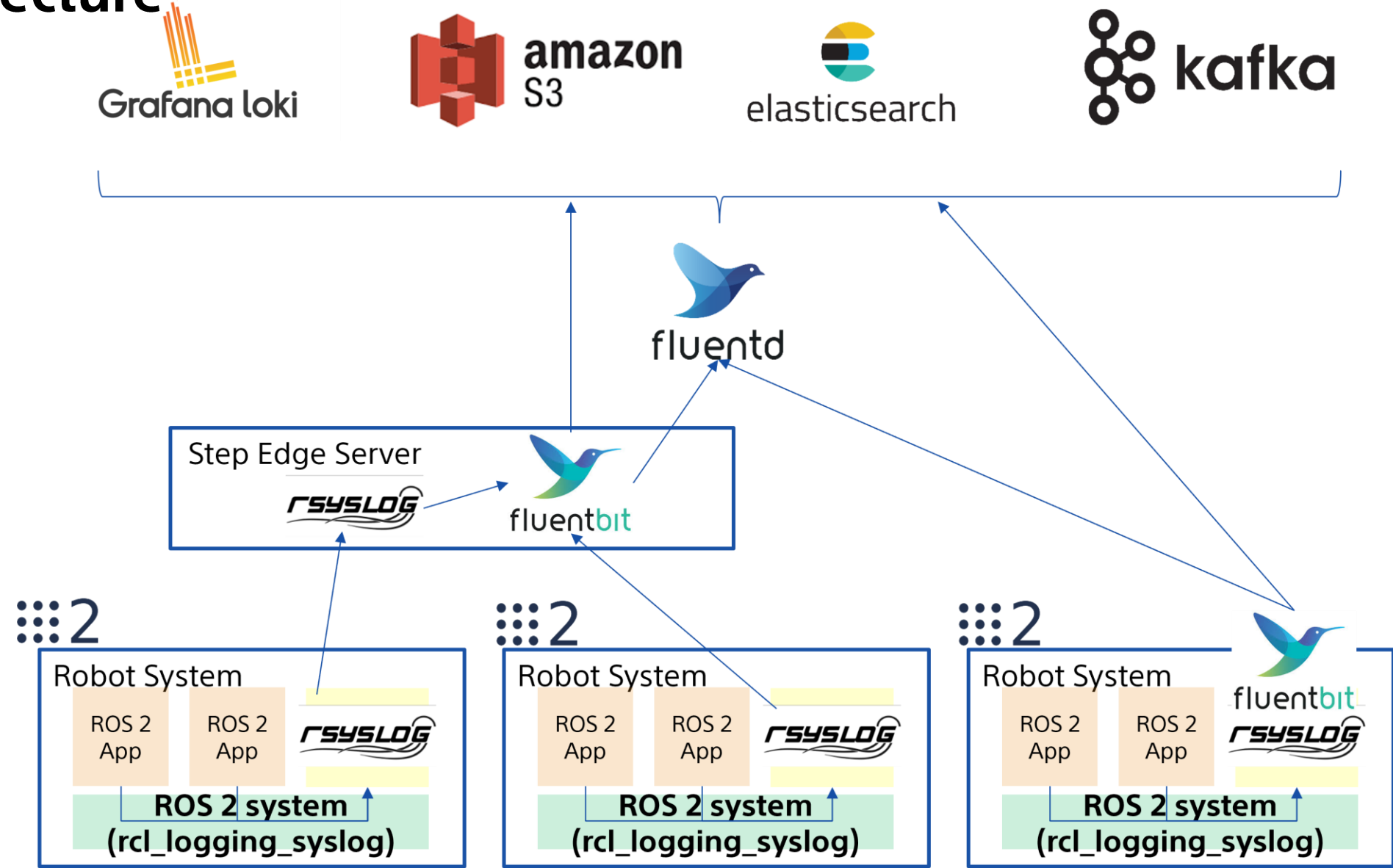
- How do we manage hundreds and thousands robots log data?
- I want to receive the event if critical error happens.
- Entire system observability.
- Individual logging level for different logging pipeline.
- Configure log behavior without any code change.
- Log data pipeline and forward capability support.
- I also want to collect Non-ROS application system log all together.

rcl_logging_syslog

- Alternate implementation for rcl_logging_interface
- ROS 2 rcl logging implementation built on top of [syslog\(3\)](#)
- Connects with [rsyslog](#) and [FluentBit](#) etc...




Architecture





```
root@tomoyafujita: ~/docker_ws/ros2_colcon 78x20
root@tomoyafujita:~/docker_ws/ros2_colcon# ros2 run demo_nodes_cpp talker
```


```
root@tomoyafujita: ~/docker_ws/ros2_colcon 78x20
root@tomoyafujita:~/docker_ws/ros2_colcon# ros2 run demo_nodes_py listener
```


```
/bin/bash 136x17
tomoyafujita@~/DVT/fluent-bit >/opt/fluent-bit/bin/fluent-bit --config=./fluent-bit.conf
```


 Home

 Starred

 Dashboards


 Explore

 Alerting

 Connections

Add new connection

Data sources

 Administration

Search or jump to...

ctrl+k

Home > Connections > Data sources


Data sources

+ Add new data source

View and manage your connected data source connections

Search by name or type

Sort by A-Z

 loki

Loki | http://43.135.146.89:3100 | default

Build a dashboard

Explore

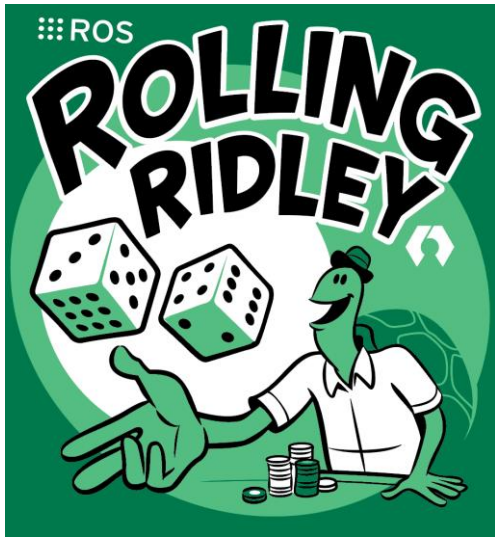
```
root@tomoyafujita:~/ros2_ws/colcon_ws# ros2 run demo_nodes_cpp talker

root@tomoyafujita:~/ros2_ws/colcon_ws 100x13
root@tomoyafujita:~/ros2_ws/colcon_ws# ros2 run demo_nodes_py listener

root@tomoyafujita:~/ros2_ws/colcon_ws 100x13
root@tomoyafujita:~/ros2_ws/colcon_ws# ros2 run demo_nodes_cpp add_two_ints_server

root@tomoyafujita:~/ros2_ws/colcon_ws 100x13
root@tomoyafujita:~/ros2_ws/colcon_ws# while true; do ros2 run demo_nodes_cpp add_two_ints_client; s
leep 1; done
```

Supported Distribution



rsyslog

a.k.a rocket-fast system for log processing 🚀 🚀 🚀

rsyslog is available in default Ubuntu distribution managed by system service, performative, and many configuration supported including log data pipeline.

So that user can choose the logging configuration depending on the application requirement and use case, sometimes file system sink, sometimes forwarding to remote rsyslogd, or even FluentBit.



- **Lightweight and Efficient**
 - suitable for environments with limited computational power.
- **High Performance**
 - capable of handling high-volume data streams with minimal latency. It leverages asynchronous I/O and efficient data processing techniques to ensure optimal performance.
- **Flexibility**
 - supports a wide range of data sources and destinations.
- **Extensibility**
 - highly extensible through plugins including custom ones.
- **Scalability**
 - easily scaled horizontally to handle increasing data volumes by deploying multiple instances.
- **Reliability**
 - features like fault tolerance and retry mechanisms to ensure data reliability.

How to use

```
export RCL_LOGGING_IMPLEMENTATION=rcl_logging_syslog  
colcon build --symlink-install --cmake-clean-cache --packages-select rcl_logging_syslog rcl
```

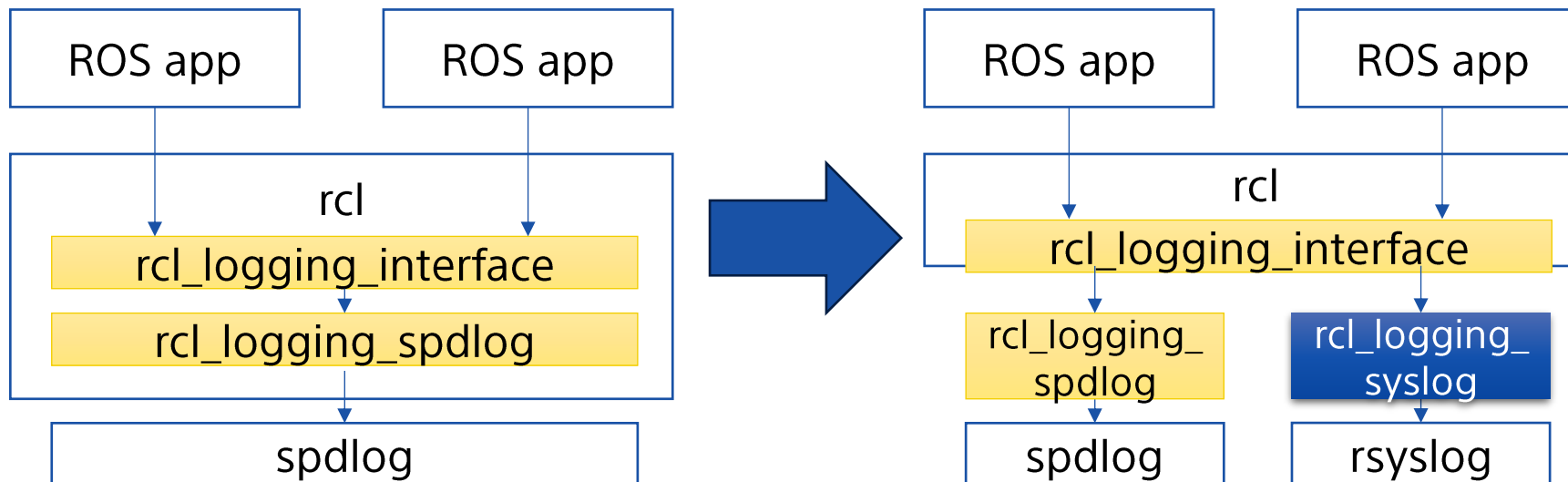
If you use docker, either binding [-v /dev/log:/dev/log](#) or enable [rsyslogd](#) in the container.

See more details for [Installation Tutorial](#)



Future Plan

- Package is not available yet, because...
- rcl_logging_interface implementation must be build and integrated with rcl binary. That says, we cannot change the rcl logging implementation at runtime without rebuilding the source code. This needs to be fixed
[Feature Request: Select the logger without rebuilding](#)



Issues and PRs always welcome 🚀

source code, documentation, presentation slides,
everything is here.

https://github.com/fujitatomoya/rcl_logging_syslog

