

ROS 2 Logging Subsystem

- 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. 🚀 TODAY's TOPIC 🚀

Enabling and Disabling Logging Backends

- Node option in the program. (e.g `rclcpp::NodeOptions`)
- 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.

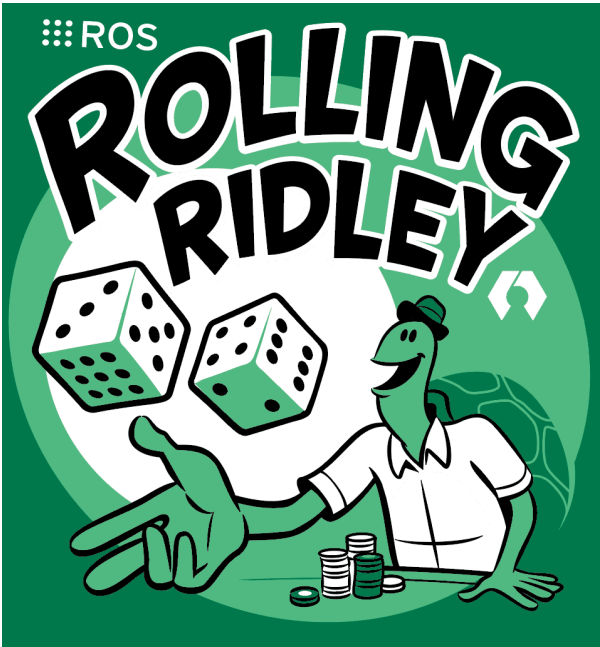
For more information about ROS 2 logging...

- logging level configuration service (default OFF)
- logging level configuration via CLI
- Console output format and output colorizing
- Setting the log file name prefix
- logging environmental variables

rcl_logging_syslog

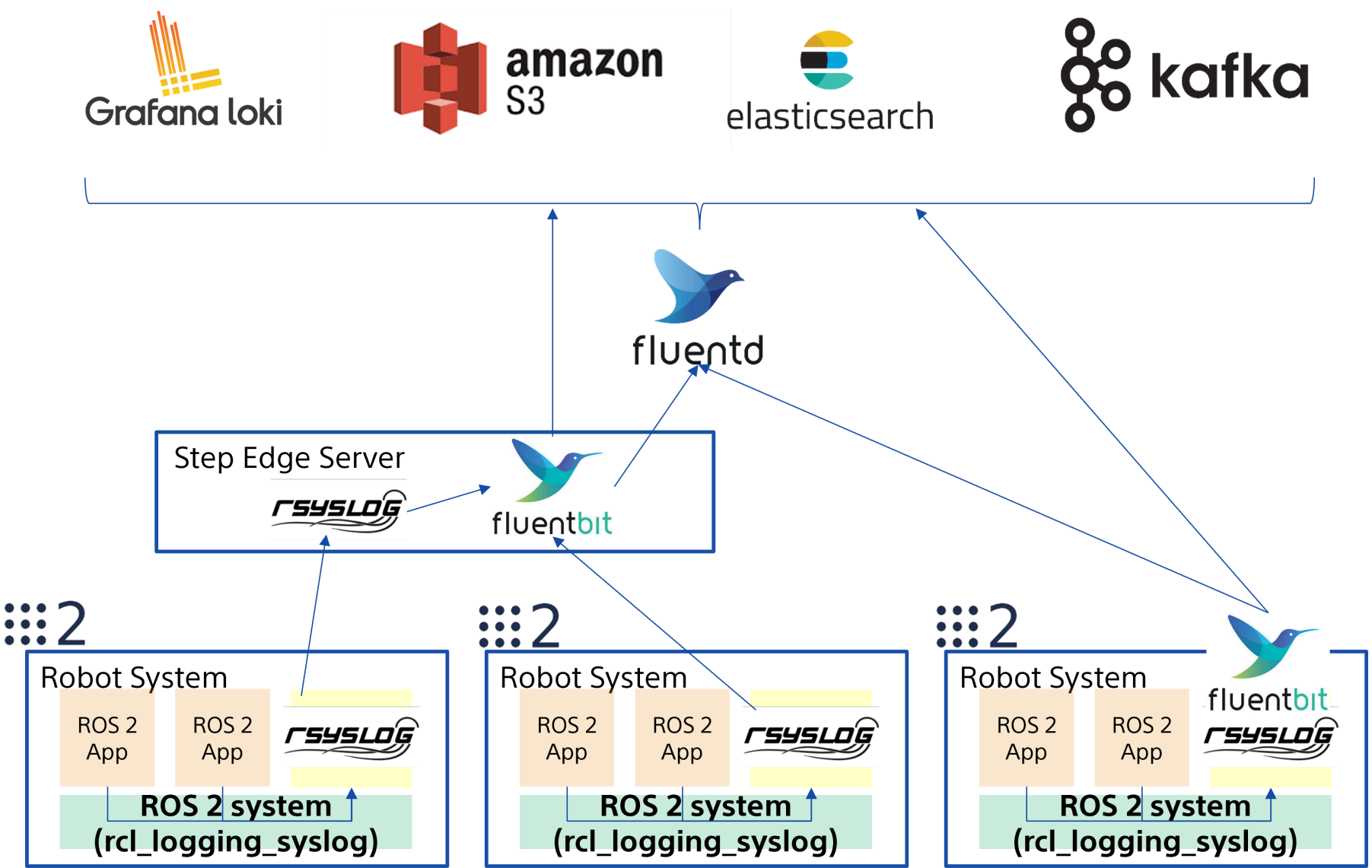
- ROS 2 rcl logging implementation built on top of [syslog\(3\)](#).
- Connects with [rsyslog](#) and [FluentBit](#) / etc...



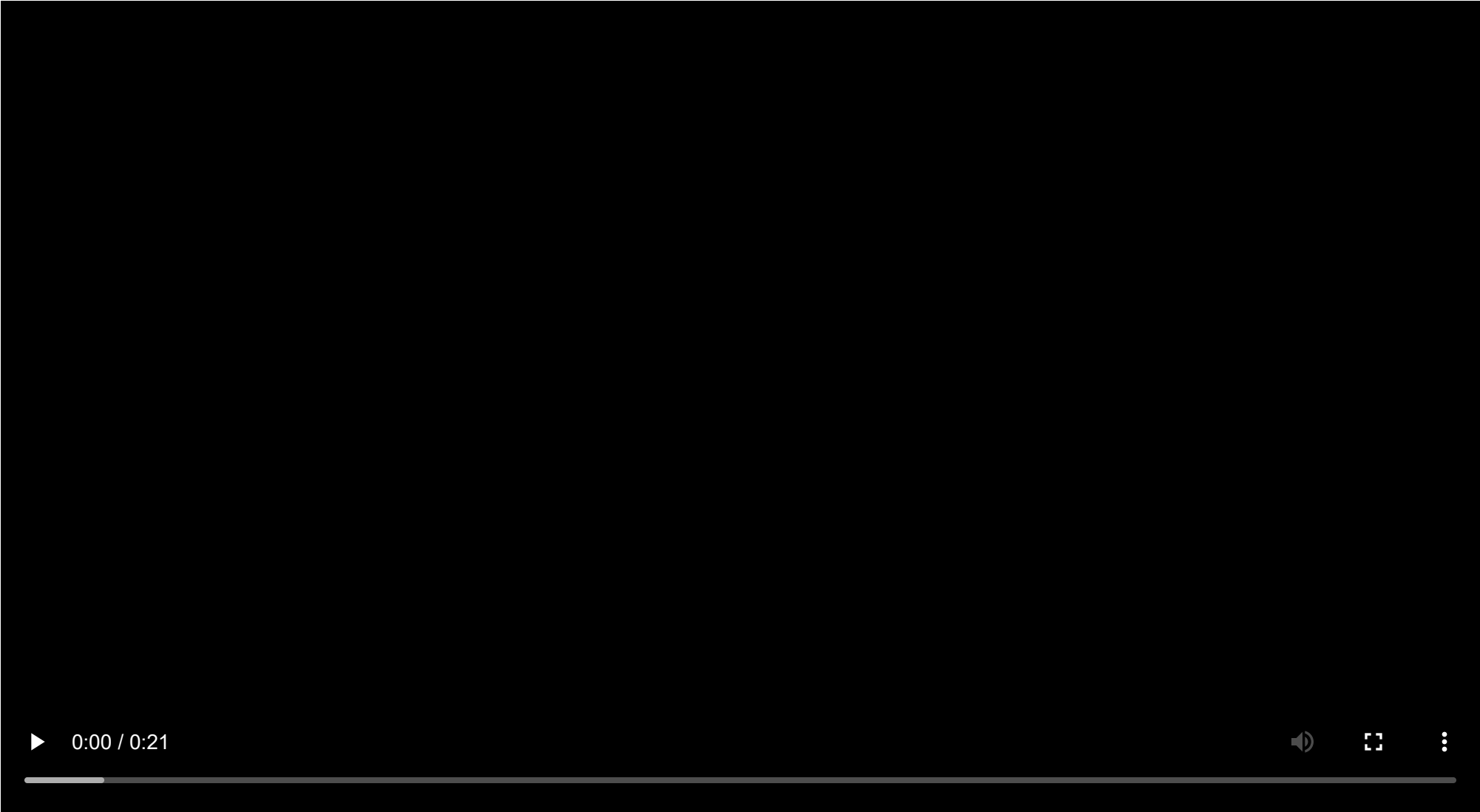


Objectives

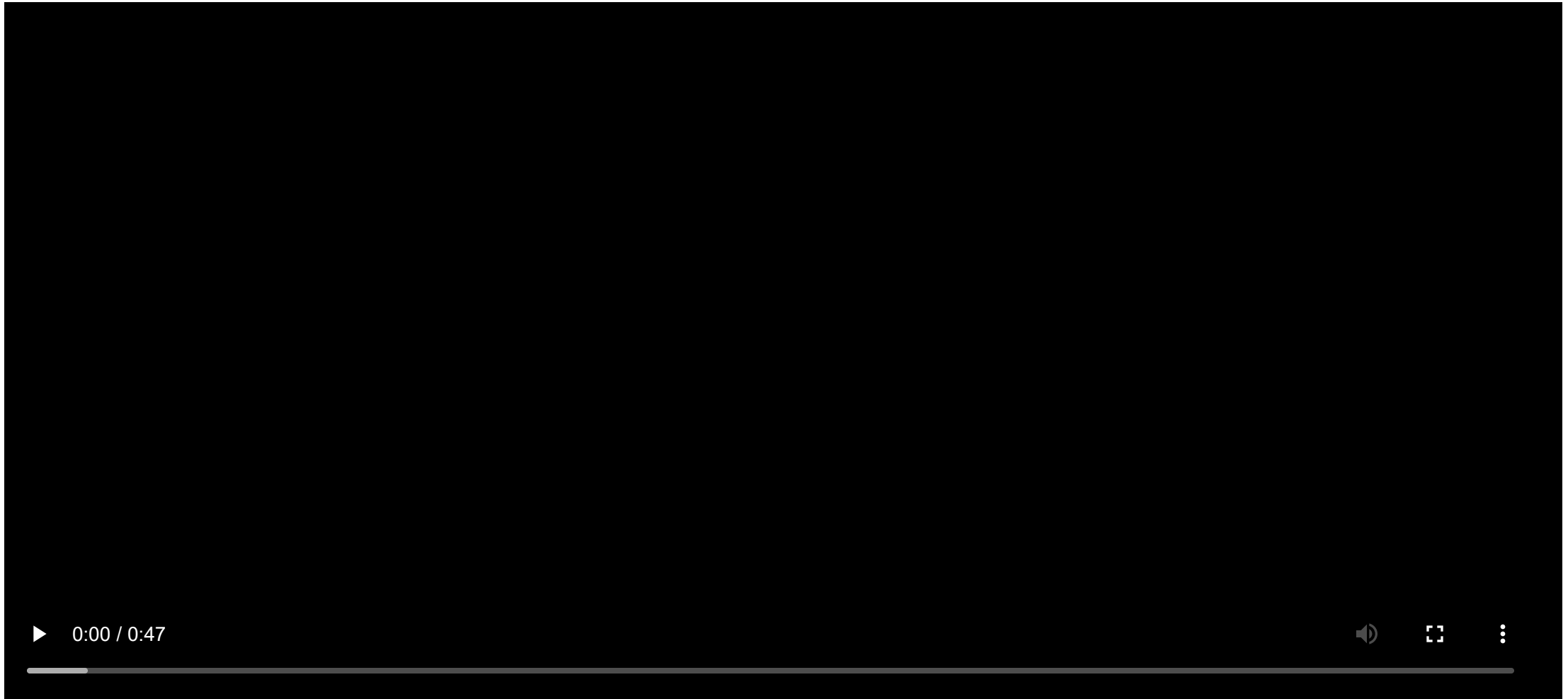
- Configure log behavior without code change.
- Multiple sink with different logging level filtering.
- Reasonable Performance.
- Log data pipeline and forward capability support.
- Enabling ROS 2 logging system with Cloud-Native Log Management and Observability.



Demo FluentBit



Demo Fluentd/Loki/Grafana

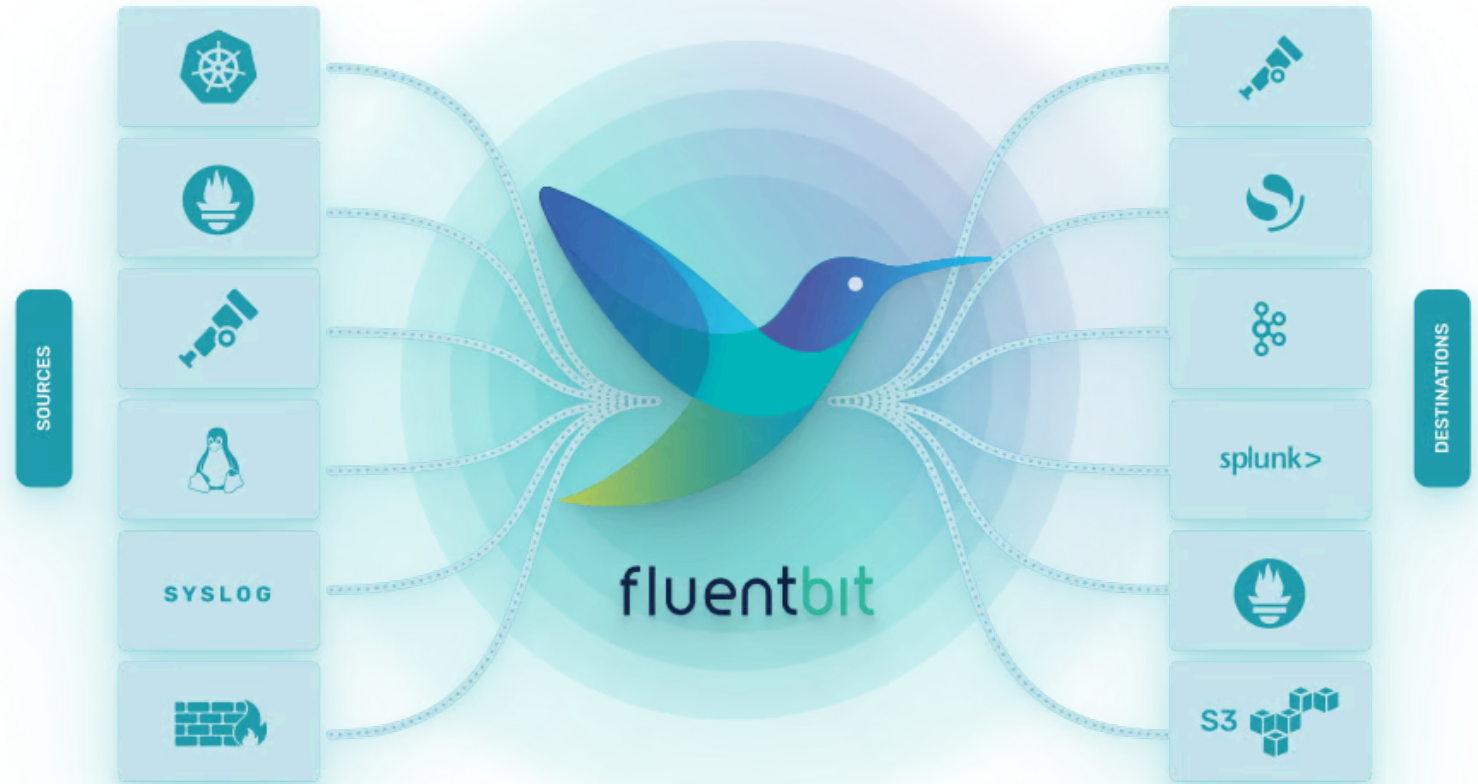


rsyslog

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](#).

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
```

- The pain is we always need to build for now... we cannot choose the external logger at runtime...
- [Feature Request: Select the logger without rebuilding](#), hopefully i could fix this in next release 🤔🤔🤔
- If you use docker, either binding `-v /dev/log:/dev/log` or enable `rsyslogd` in the container.

Issues and PRs always welcome 🚀

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

https://github.com/fujitatomoya/rcl_logging_syslog

