Cluster-Level Logging with Kubernetes

Jonathan Hancock

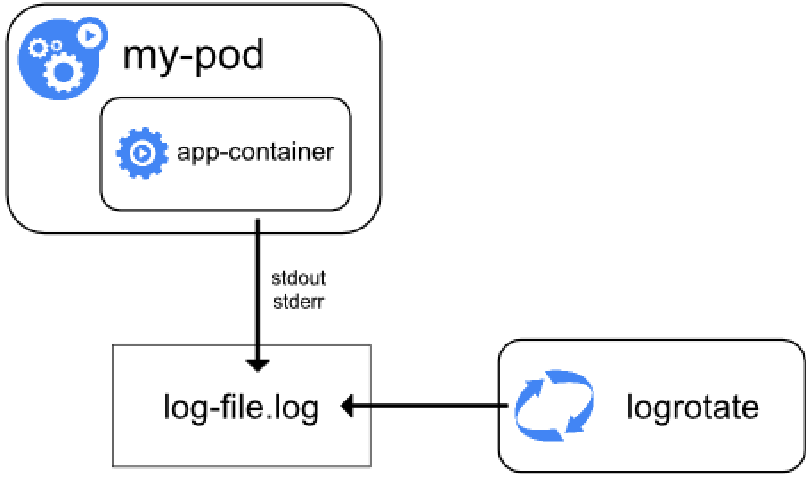
Motivation

- Logging is an important part of any CCM system, allowing for easier debugging of problems, and monitoring of applications.
- It makes sense for logging to be separate from the lifecycle of pods and nodes: if a node dies suddenly, we wouldn’t want its logs to be lost with it.
- Kubernetes does not natively support external logging, but it can be implemented without excessive effort.

UNIVERSITY OF BIRMINGHAM

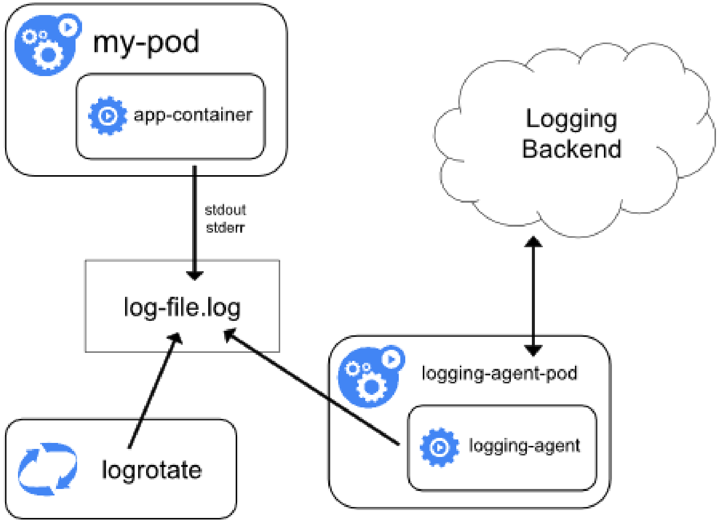
The default setup

- When the container writes to stdout or stderr, the container engine redirects the output.
- For example, docker will write it to a log file in the JSON format.
- Kubernetes clusters use the logrotate tool to delete the logs every hour.
- All the features mentioned can of course be customised.



Node-level logging agent

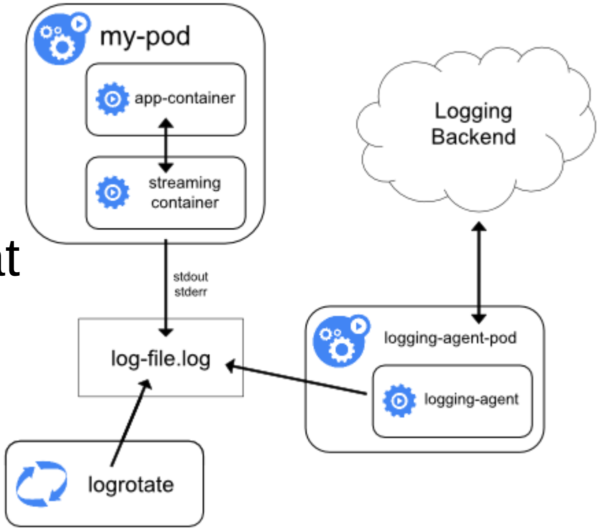
- We can upgrade to cluster-level logging by adding a logging agent pod to each node.
- A logging agent is an application that can read all logs in the node, and push or expose them to a backend.
- A DaemonSet is used to ensure all nodes run a copy of this pod.



UNIVERSITY OF BIRMINGHAM

Sidecar application

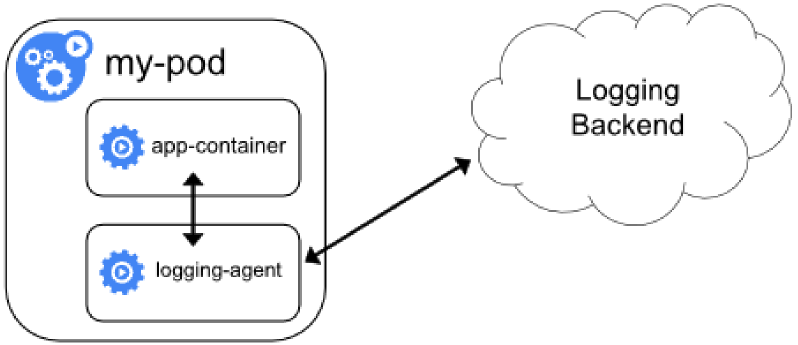
- The system can be improved by adding a “sidecar” container to pods that produce logs.
- This container reads the logs from the application, then writes them to stdout/stderror itself.
- This allows for more processing of the application logs, such as separating logs from different parts of it, or using formats that cannot be written directly to stdout.



UNIVERSITY OF BIRMINGHAM

Sidecar application (cont.)

- Sometimes the node-level logging agent is not flexible enough.
- In this case, we can make sidecar containers that function as logging agents, and communicate directly with the backend.
- This can be resource intensive, so shouldn't be the default implementation.



UNIVERSITY OF BIRMINGHAM

Citations

- All diagrams from https://kubernetes.io

UNIVERSITY OF BIRMINGHAM