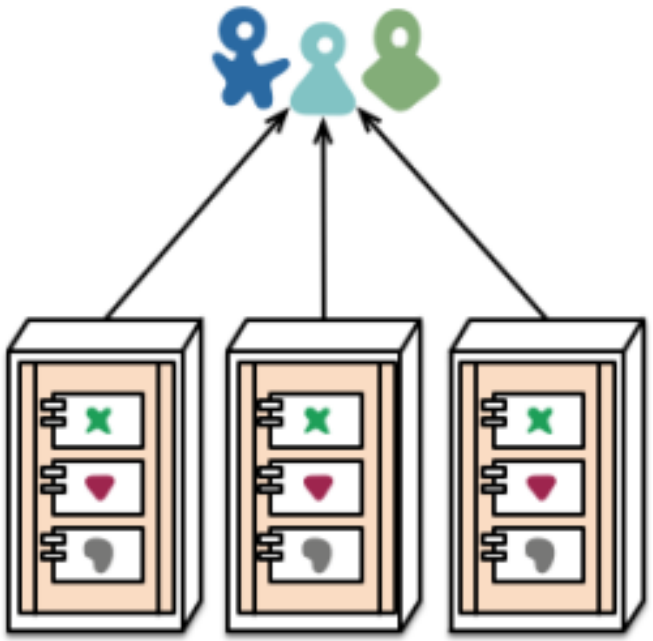
Microservices in Production:
Logging Patterns

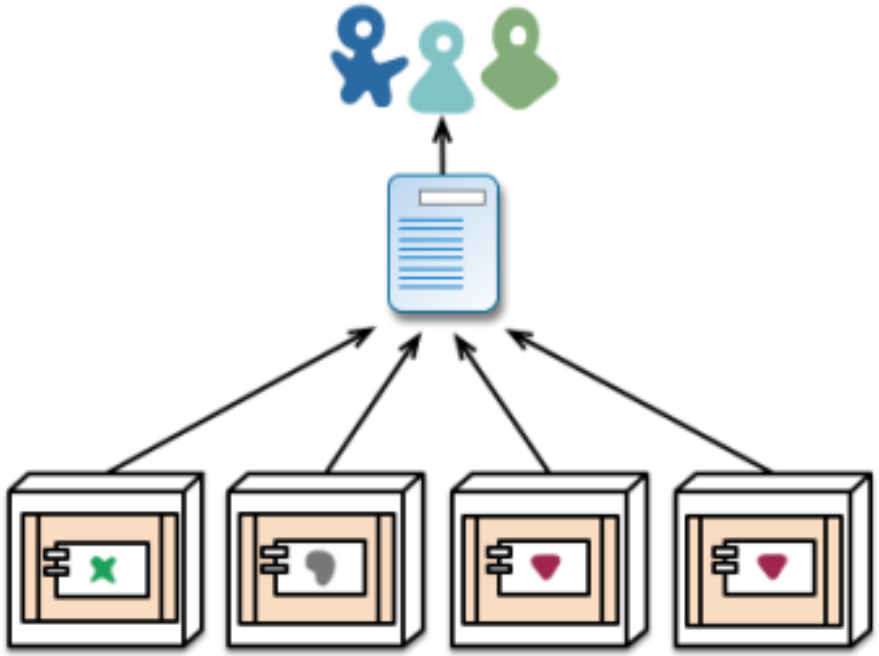
Christian Melendez
Cloud and DevOps Consultant

ASPE
cprime COMPANY

What is a Microservice?



monolith - multiple modules in the same process



microservices - modules running in different processes

Microservice: Advantages

- Simple services that are focused on doing one thing well
- Each service can be built using the best tool for the job
- Systems built this way are inherently loosely coupled
- Teams can deliver and deploy independently from each other
- Enable continuous delivery but allowing frequent releases while the rest of the system continues to be stable

Microservices = Optimize for Speed

"Organizations with these types of service-oriented architectures, such as Google and Amazon, have incredible flexibility and scalability. These organizations have tens of thousands of developers where small teams can still be incredibly productive."

Randy Shoup, former Engineering Director for Google App Engine

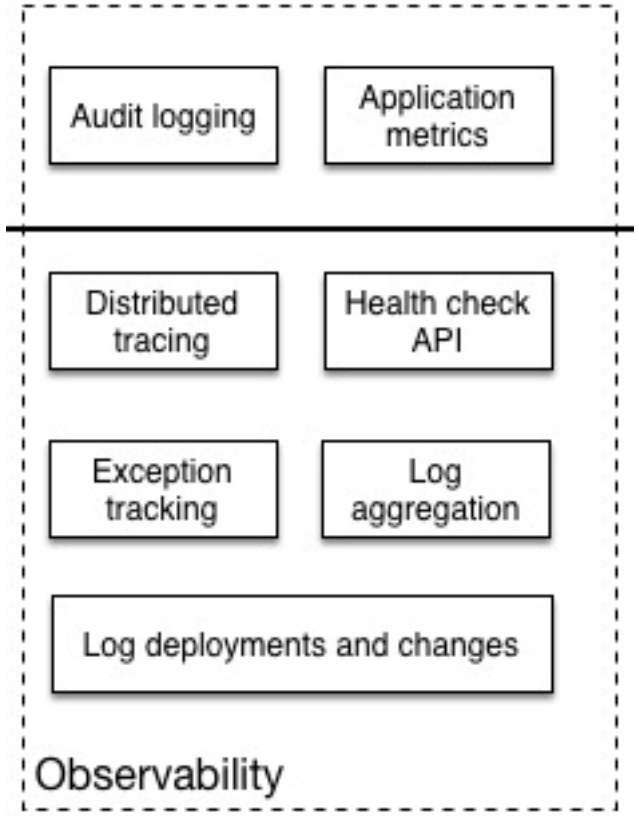
How do you know everything is working?

Observability Patterns

Health Check API
A health check client - a monitoring service - periodically invokes the endpoint to check the health of the service instance

Ideally, you have a service that can ensure your service is active and accepting requests, a 'Health Check'

Health check examples included Consul, Kubernetes, and various libraries such as Sprint Boot, Gokit, etc.



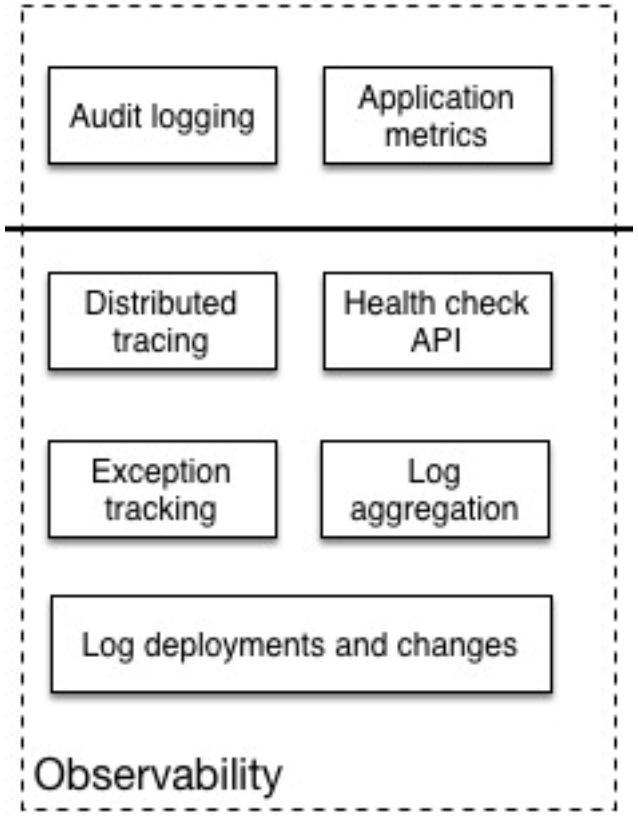
Application Metrics
Need to understand the behavior of an application so performance and other problems can be proactively managed

Typically come in two models:

- **Push – service pushes metrics (typically an agent) from the service**
- Pull – Service pulls metrics from the service (typically a dashboard)

Examples include Prometheus, AWS Cloud Watch, Sysdig, etc

Observability Patterns



Distributed Tracing

- It becomes quite important to understand what broke and why

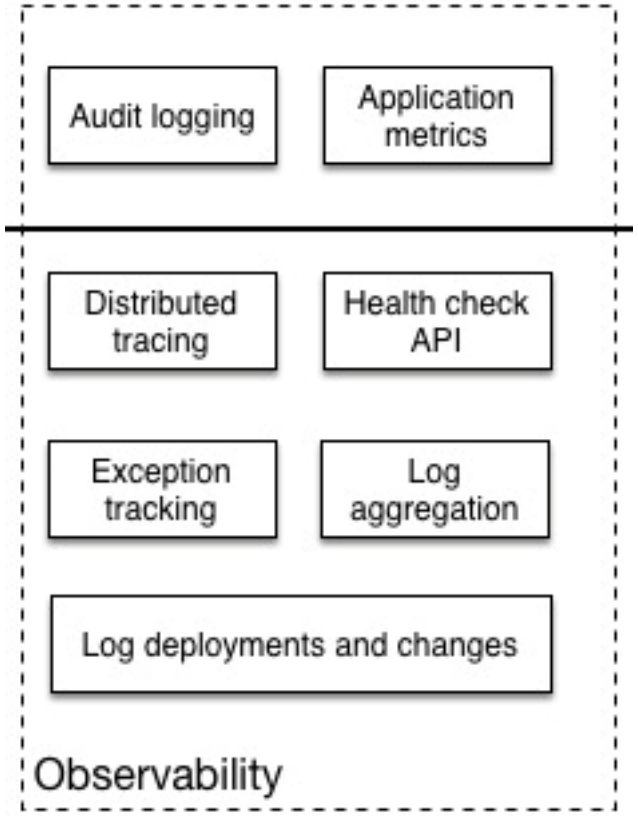
Common Distributed Tracing functionality

- Assign each external request a unique ID
- Pass ID to all services involved with handling the request
- Include id in all log messages
- Record information (start/end time) about requests and operations performed

Observability Patterns

Exception Tracking

- It is crucial to save errors and the corresponding stack trace
- Ideally, exceptions are de-duplicated and recorded for further aggregation and investigation



Log deployments and changes

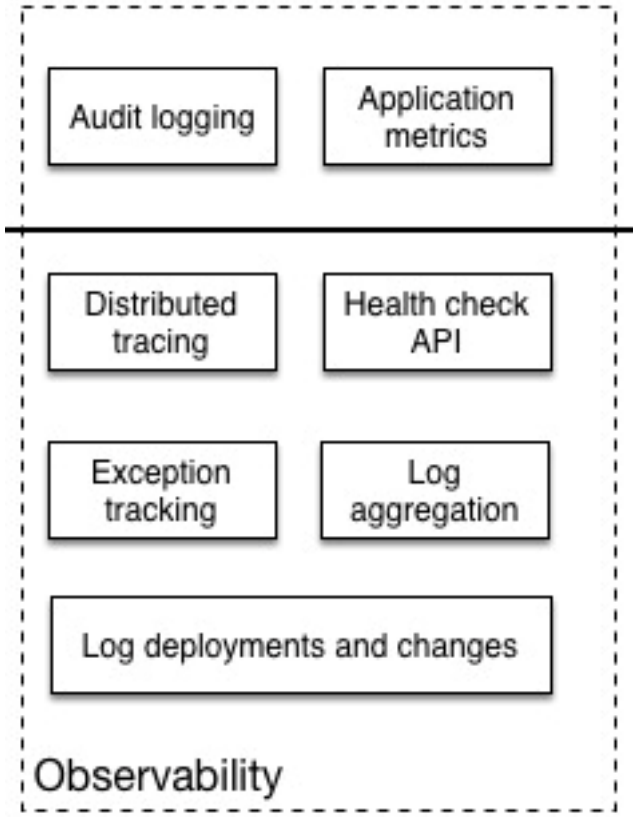
- Track changes as they are opportunities for failure
- For example, tracking deployments and changes so they can be easily correlated with issues later for faster resolution

*Etsy graphing error rates against new code deployments

Observability Patterns

Log Aggregation

- Requests span multiple instances over multiple machines. It is important to gather information in a standardized format
- Log files contain errors, warnings, and debug information
- It is important to aggregate, search and analyze such logs
- Popular solutions for this include the ELK stack (Elastic Search, Logstash and Kibana) and AWS Cloud Watch



Audit Logging

- It is vital to know actions a user has performed.
- For security and compliance, this is often a must have
- Average time to detect intrusions is 98 days for financial services, 197 days for retail, it should/can be much lower

Logging Best Practices

- Correlate service calls with a unique ID

```
X-Correlation-Id: 123e4567-e89b-12d3-a456-426655440000
```

- Include a unique ID in the response payload

```
{
  "data": {

  },
  "correlationid": "123e4567-e89b-12d3-a456-426655440000 "
}
```

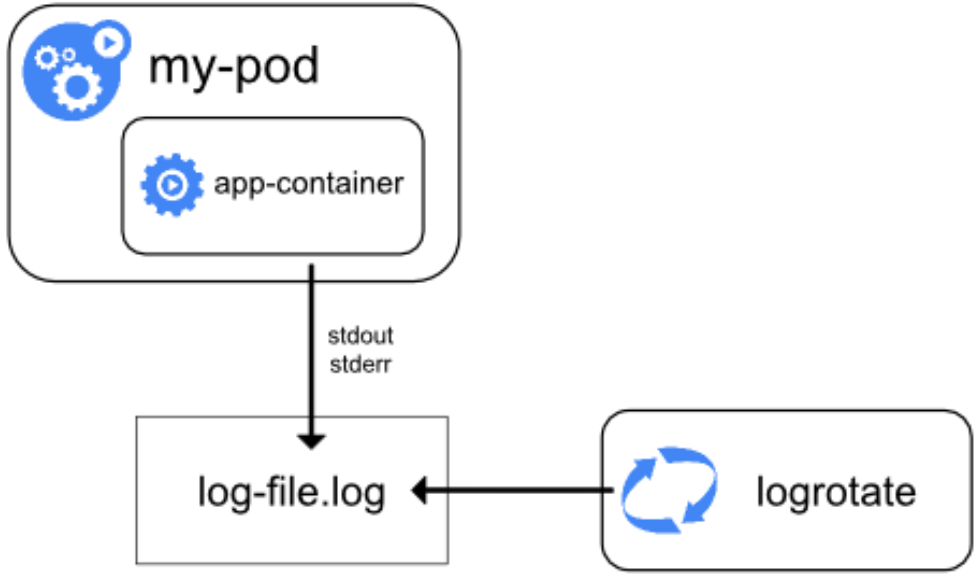
Logging Best Practices

- Centralize logs for better troubleshooting
- Structure your log data, typically a JSON structure
- Include as many context as you can like query string parameters
- Write logs to a local storage, typically a set of files
- Include traces in every service call or to any external dependency

How all of this look like in practice?

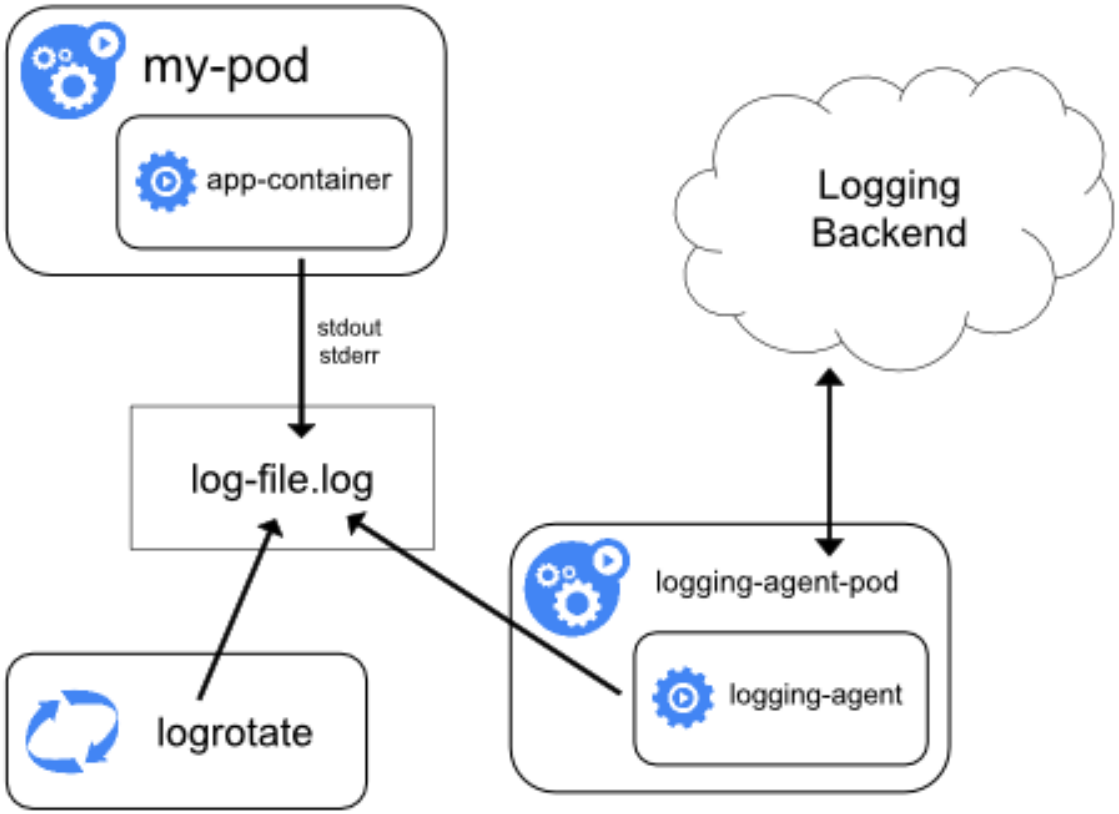
Logging Architecture in Kubernetes

- If you're not using Kubernetes, the following are good ideas
- You could replace "pod" with a VM
- Typically, you have something like this

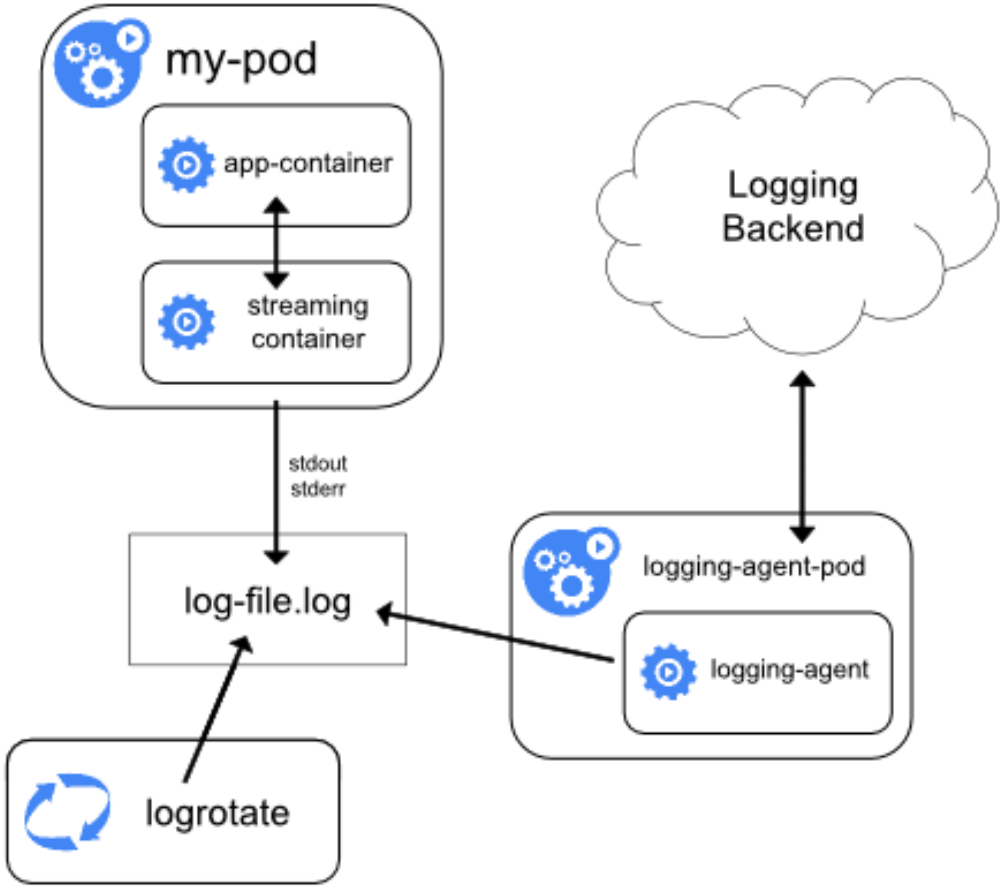


Node Logging Agent

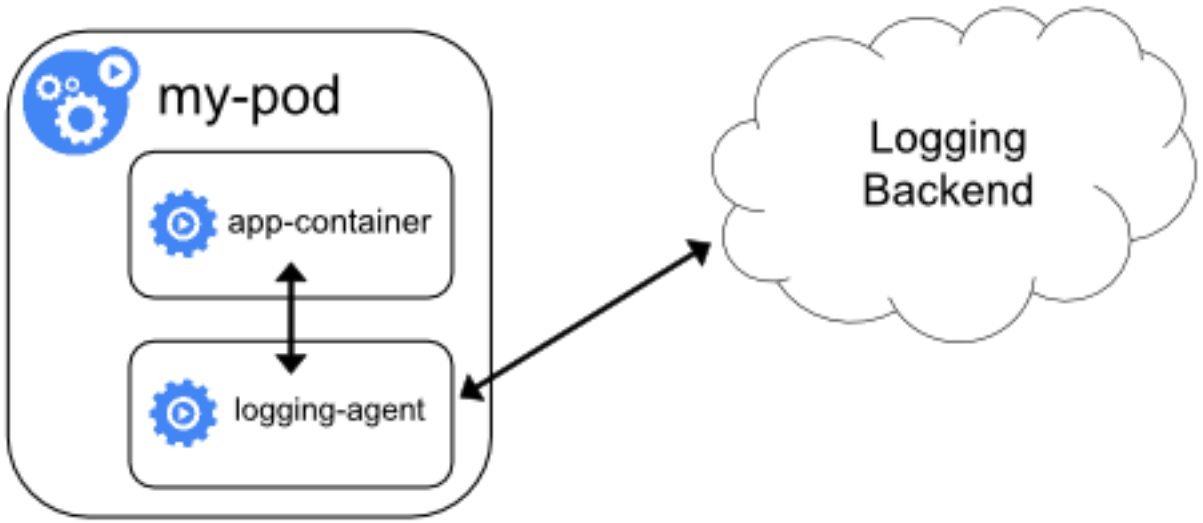
- Then, you need to move the logs somewhere else



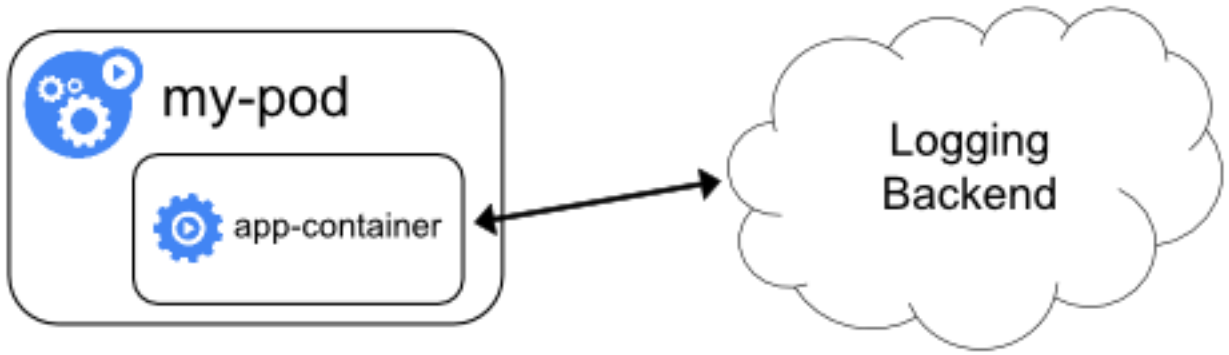
Sidecar Logging Container



Sidecar + Node Logging Container



Exposing Logs from the Application



EFK in Kubernetes

Let’s deploy Elasticsearch + Fluentd + Kibana in Kubernetes

Elasticsearch is for key/value storage for structured logs
Fluentd moves logs from files in a server to Elasticsearch
Kibana is the tool we’ll use to explore logs with a UI

Let me show you …

Questions? 

- Ask in the Chat
- Ask me later @christianmldz
- Read me later cmelendeztech.com

ASPE