CNCon / KubeCon NA Seattle 2018

Fluentd Project Intro

CLOUD NATIVE
COMPUTING FOUNDATION

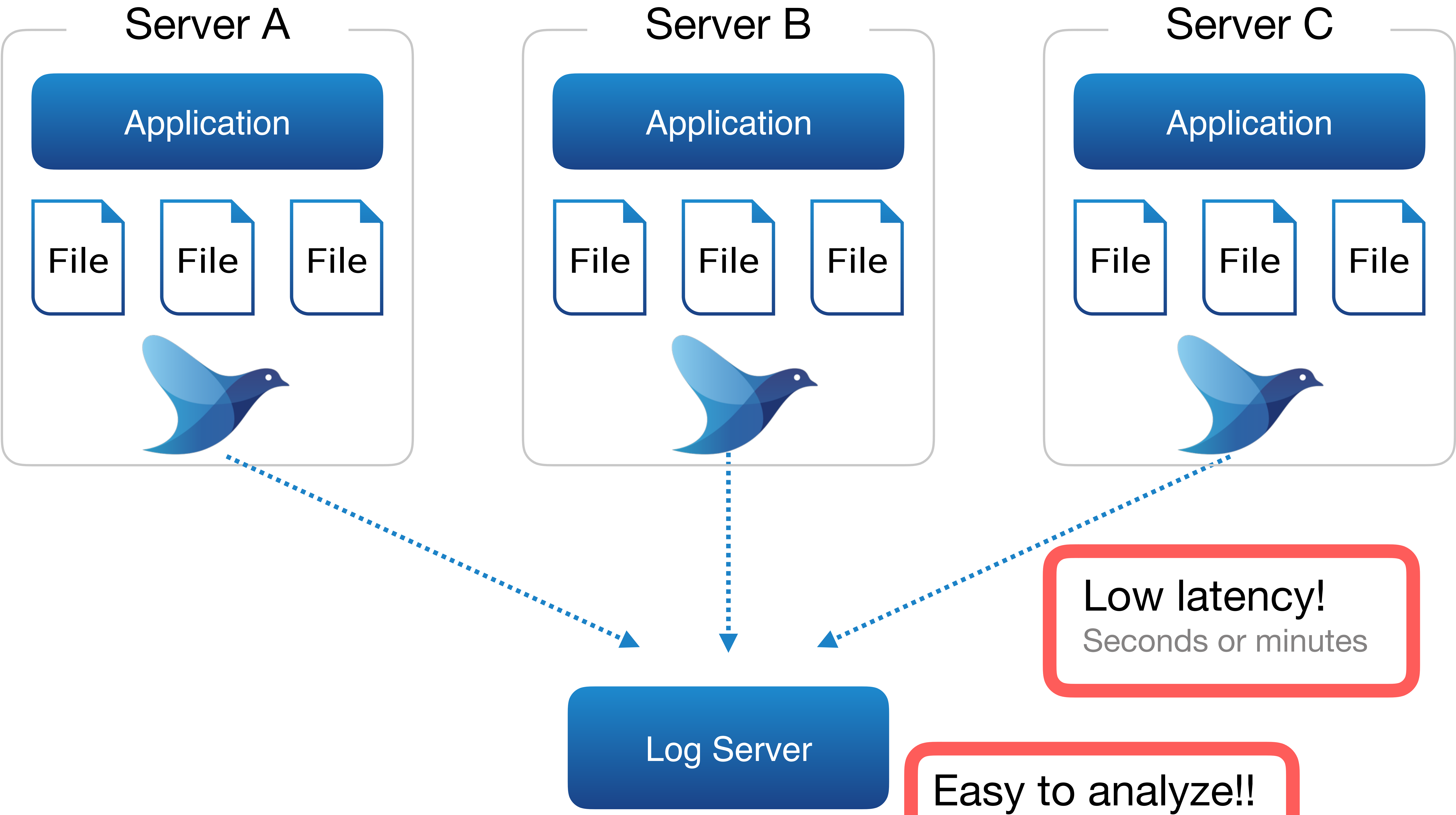
Masahiro Nakagawa
Senior Software Engineer
arm
TREASURE DATA

Fluentd overview

What’s Fluentd

- **Streaming data collector for unified logging**
  - Simple core + plugins
- **RubyGems based various plugins**
  - Follow Ruby’s standard way
- **Several setup ways**
  - https://docs.fluentd.org/v1.0/categories/installation
- **Latest version**: v1.3.2
- **Logging part in CNCF**

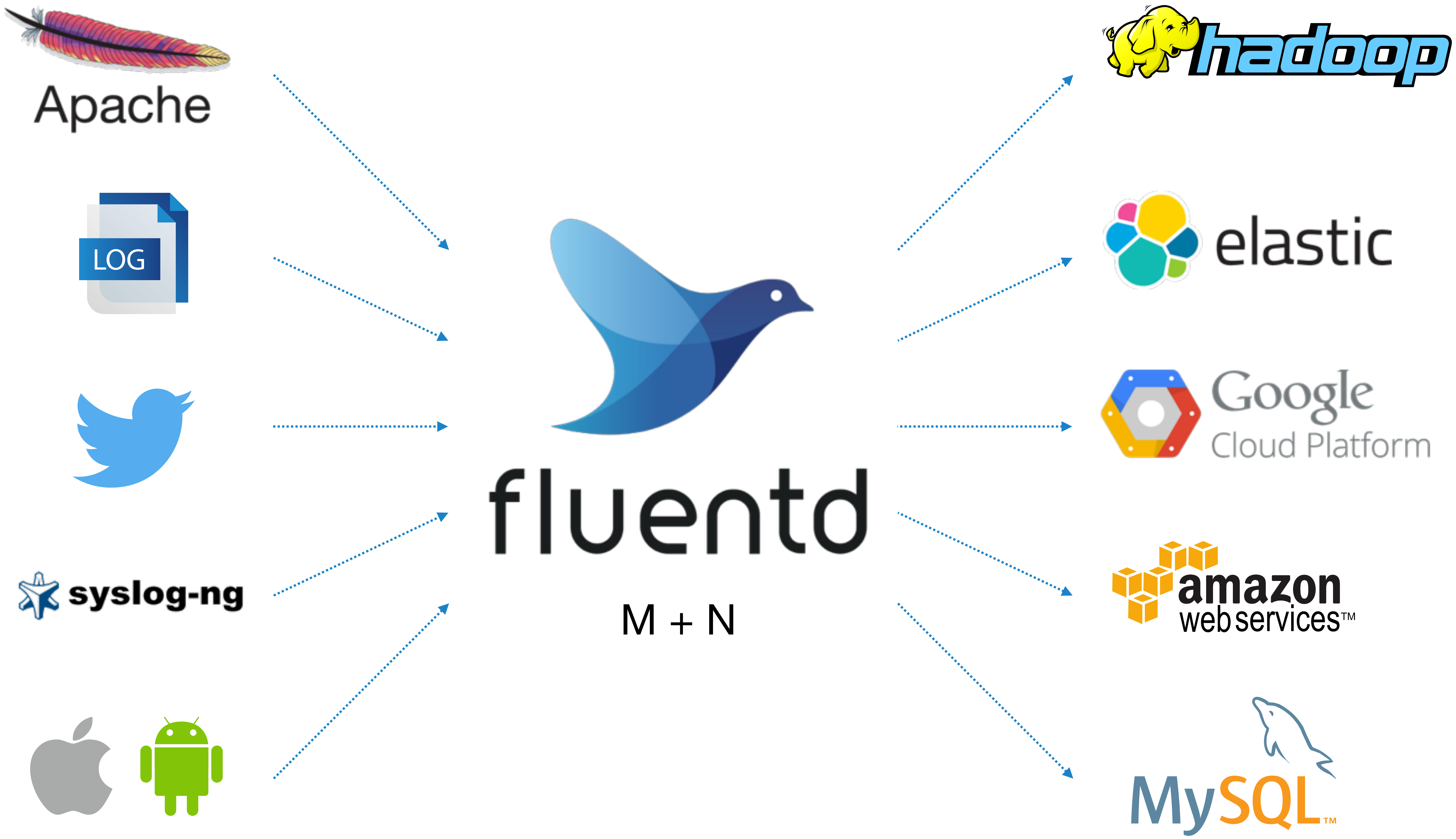
Streaming way with Fluentd



Low latency!
Seconds or minutes

Easy to analyze!!
Parsed and formatted

Unified logging layer



Fluentd Architecture

Design

Core

- Buffering & Retrying
- Error handling
- Event routing
- Parallelism

Plugins

- Read / receive data
- Parse data
- Filter / enrich data
- Buffer data
- Format data
- Write / send data

Event structure

Time

- Nano-second unit
- from logs

Tag

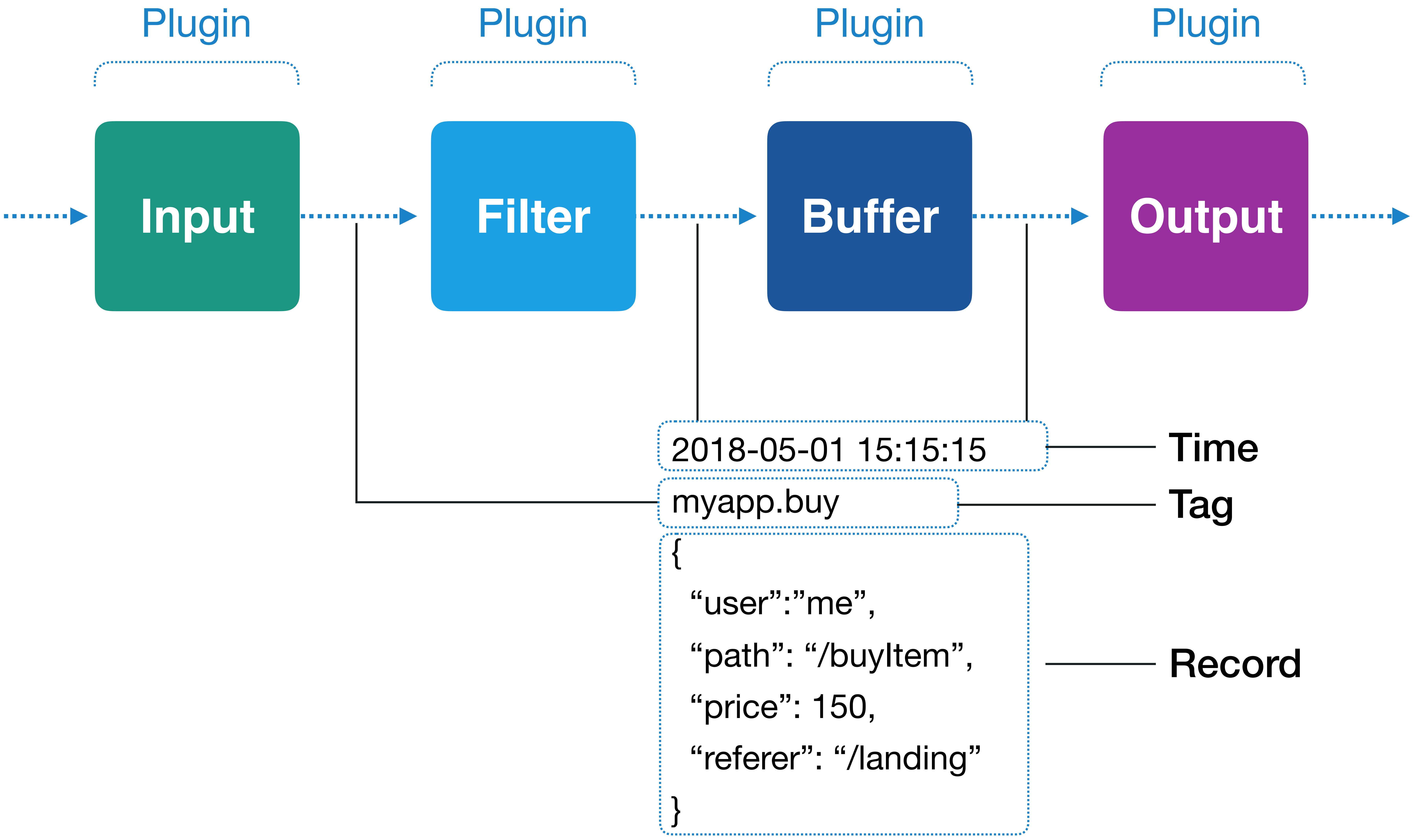
- for event routing
- Identify data source

Record

- JSON object, not raw string

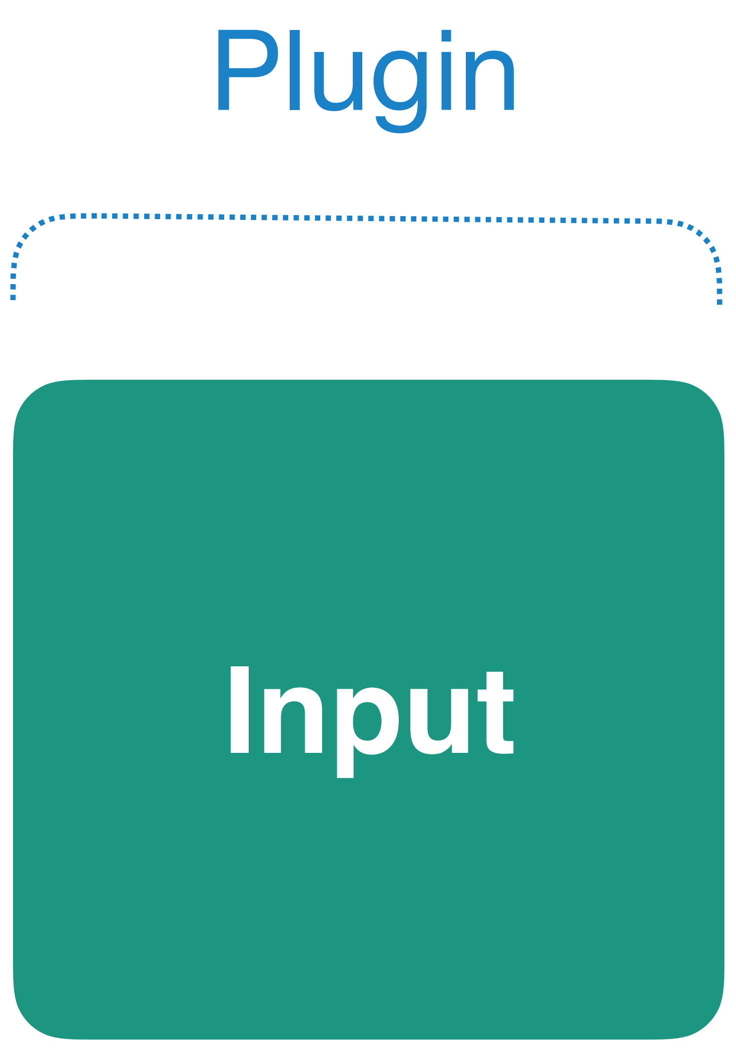
```
{
  “str_field”:”hey”,
  “num_field”: 100,
  “bool_field”: true,
  “array_field”: [“elem1”, “elem2”]
}
```

Data pipeline (simplified)



```
2018-05-01 15:15:15
myapp.buy
{
  “user”:”me”,
  “path”: “/buyItem”,
  “price”: 150,
  “referer”: “/landing”
}
```

Architecture: Input Plugins



Plugin

Input

- Receive or pull logs from data sources
- Emit logs to data pipeline
- Parse incoming logs for structured logging

HTTP+JSON (in_http)
Local files (in_tail)
Syslog (in_syslog)
…

Architecture: Filter Plugins

Plugin

Filter

- ✓ Transform logs
- ✓ Filter out unnecessary logs
- ✓ Enrich logs

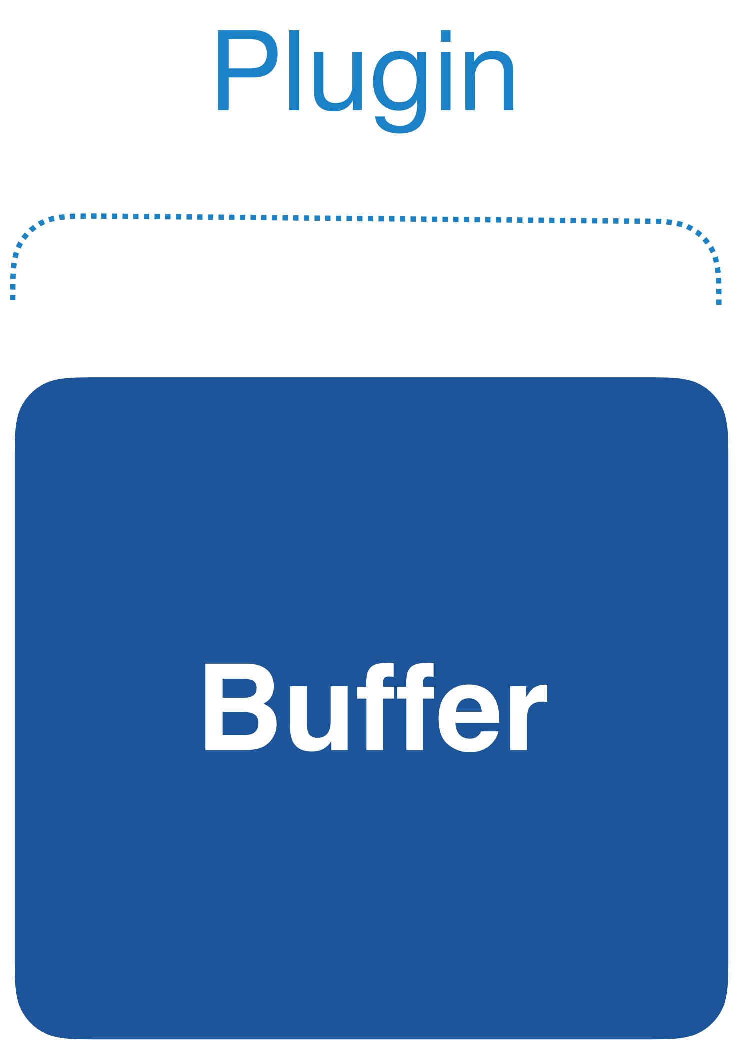
Modify logs (record_transformer)

Filter out logs (grep)

Parse field (parser)

…

Architecture: Buffer Plugins

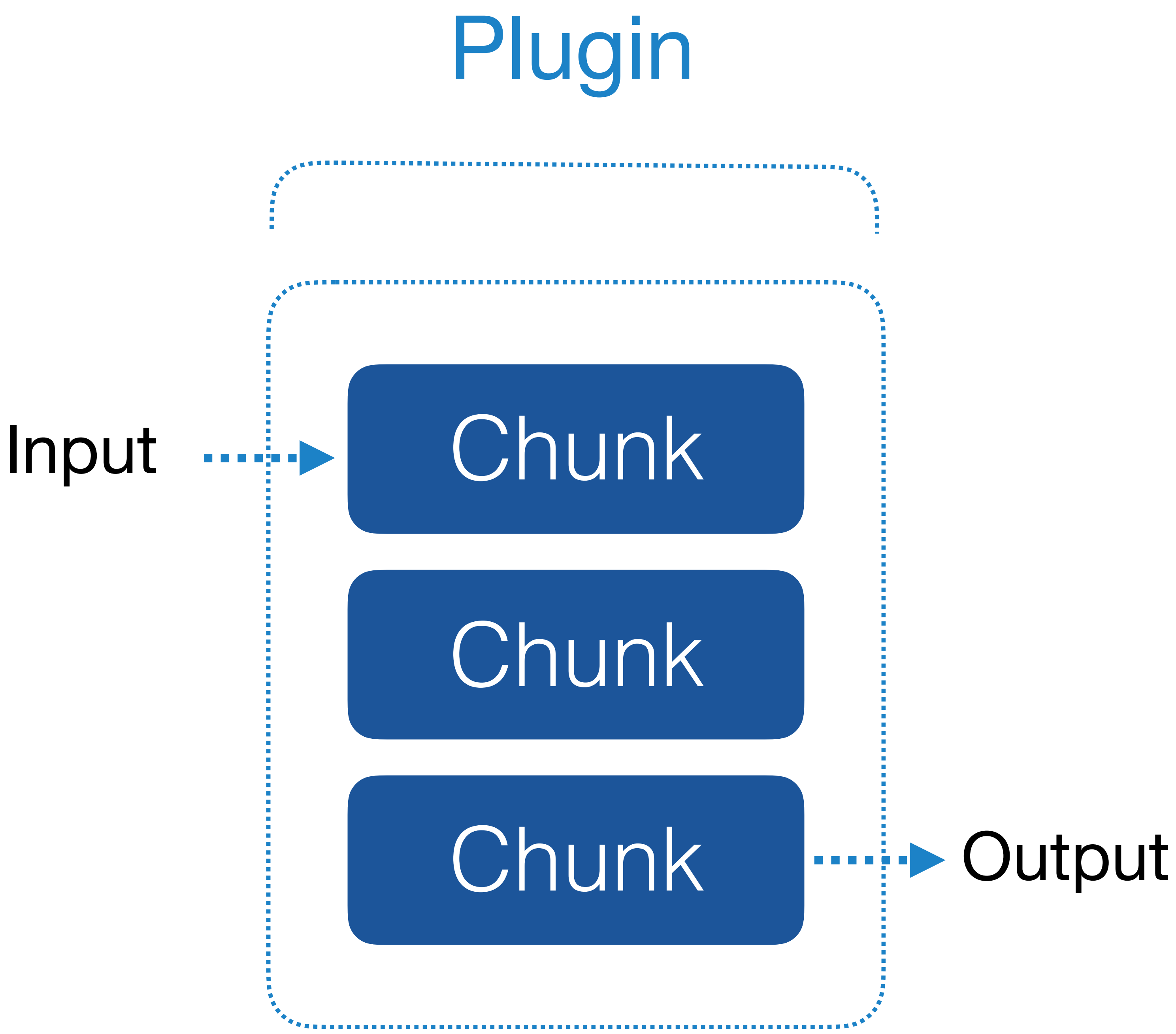


- ✓ Improve performance
- ✓ Provide reliability
- ✓ Provide thread-safety

Memory (buf_memory)

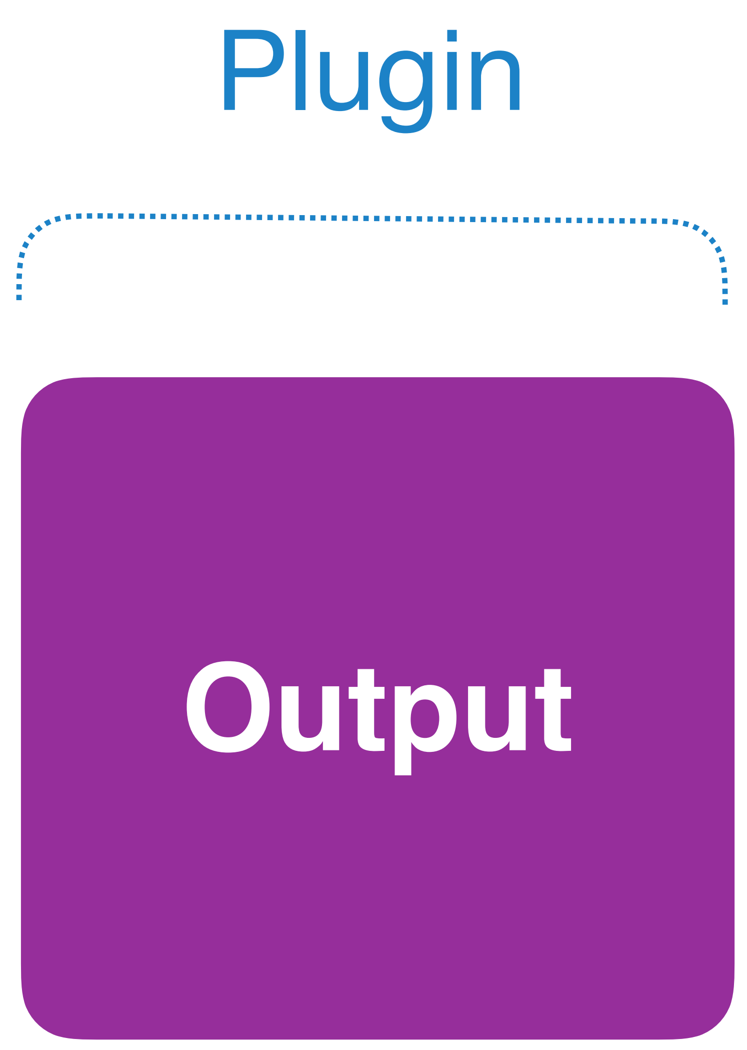
File (buf_file)

Architecture: Buffer Plugins



- ✓ Improve performance
- ✓ Provide reliability
- ✓ Provide thread-safety

Architecture: Output Plugins



- ✓ Write or send event logs
- ✓ Sync or Async

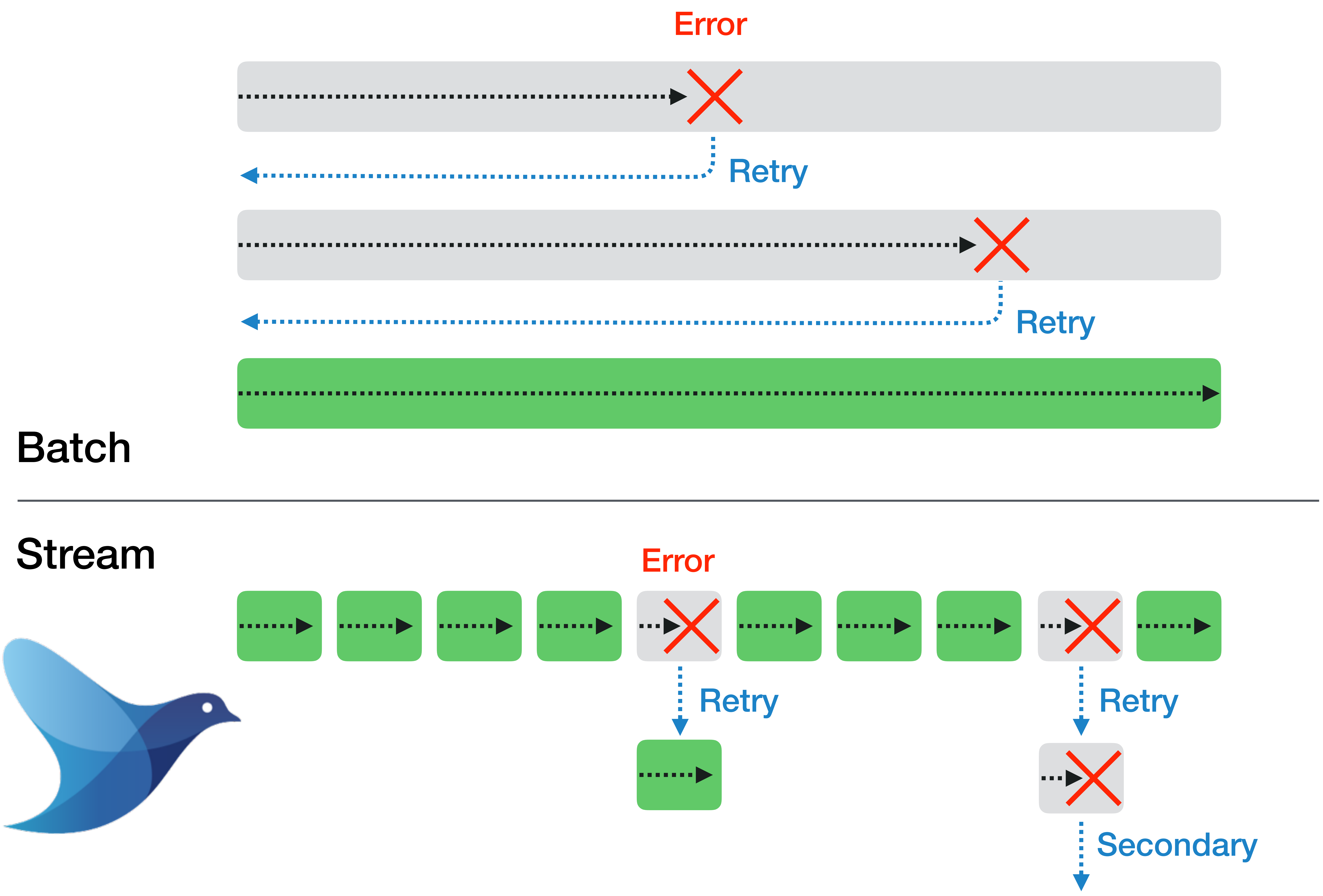
File (out_file)

Amazon S3 (out_s3)

Forward to other fluentd (out_forward)

…

Divide & Conquer for retry



Error

Retry

Retry

Batch

Stream

Error

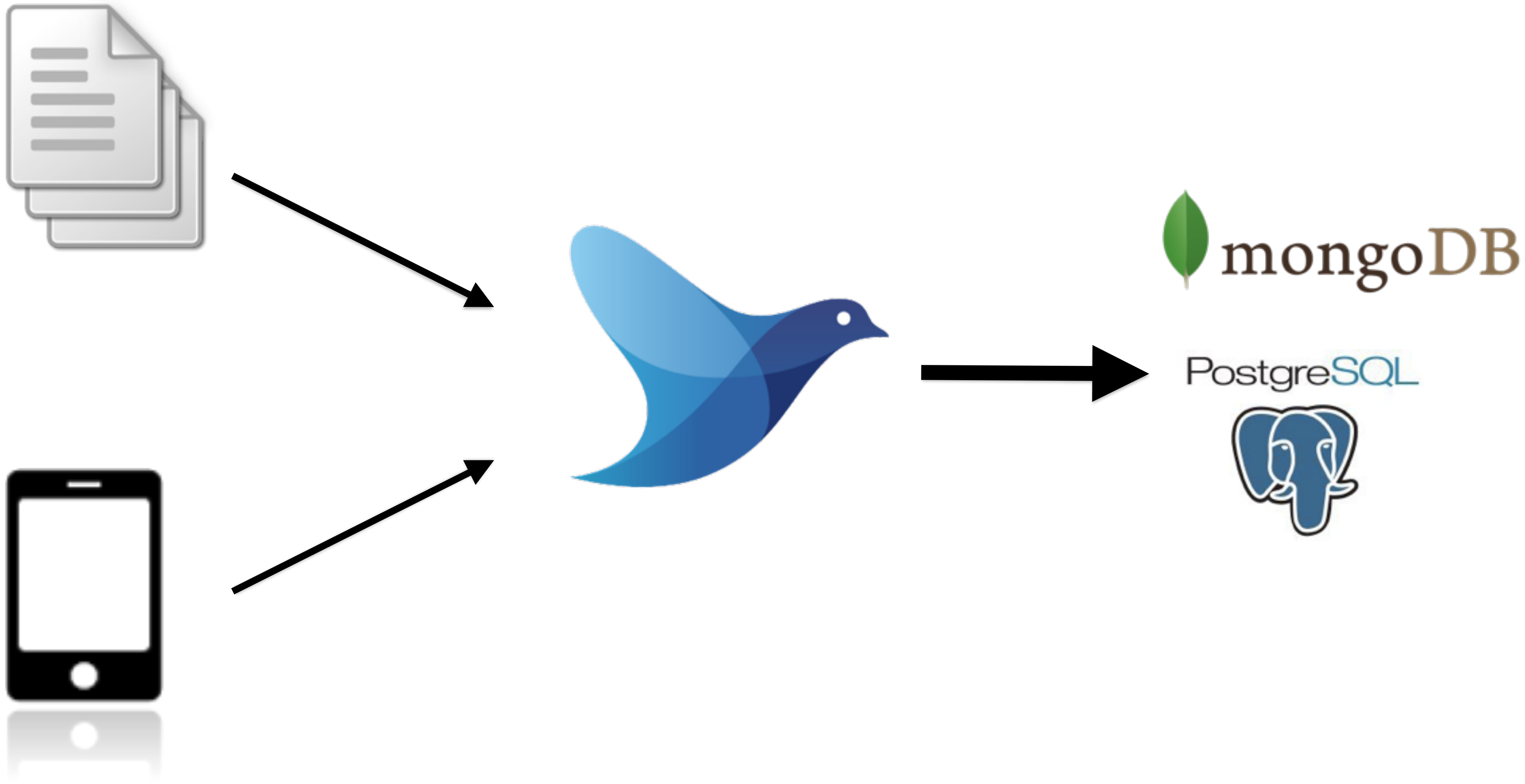
Retry

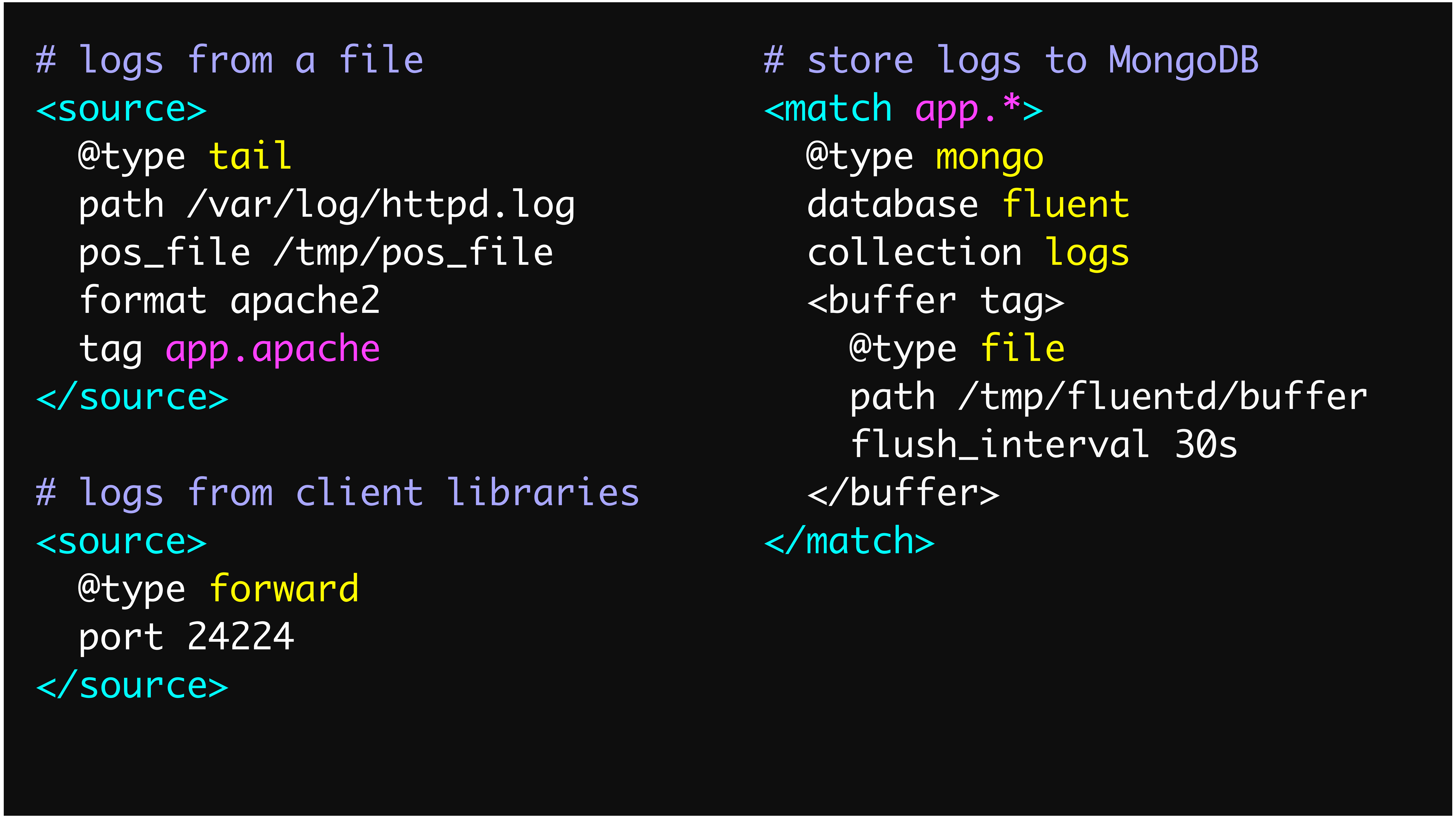
Retry

Secondary

Use-cases with Configuration Example

Simple forwarding



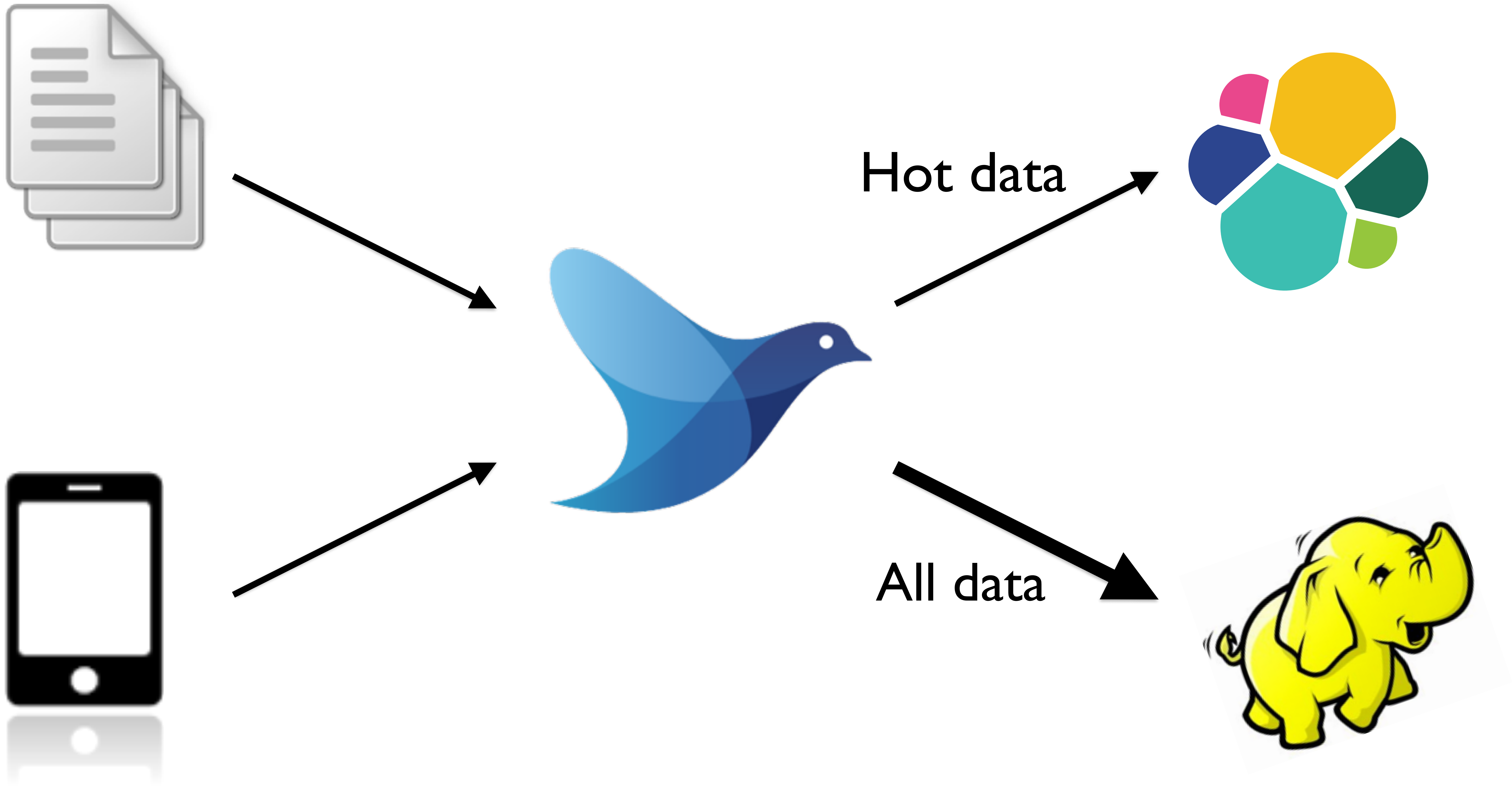


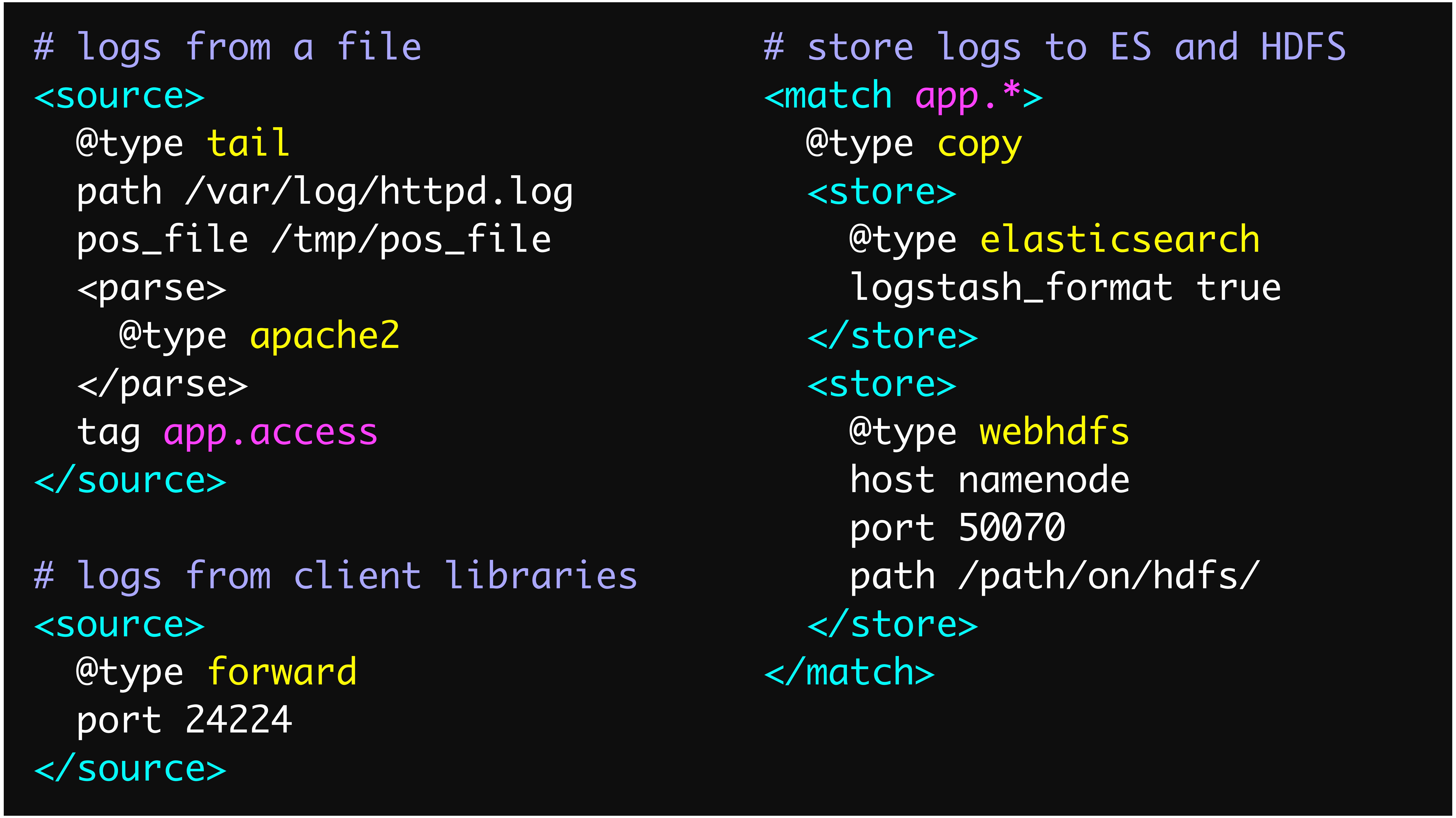
```
# logs from a file
<source>
  @type tail
  path /var/log/httpd.log
  pos_file /tmp/pos_file
  format apache2
  tag app.apache
</source>

# logs from client libraries
<source>
  @type forward
  port 24224
</source>
```

```
# store logs to MongoDB
<match app.*>
  @type mongo
  database fluent
  collection logs
  <buffer tag>
    @type file
    path /tmp/fluentd/buffer
    flush_interval 30s
  </buffer>
</match>
```

Multiple destinations



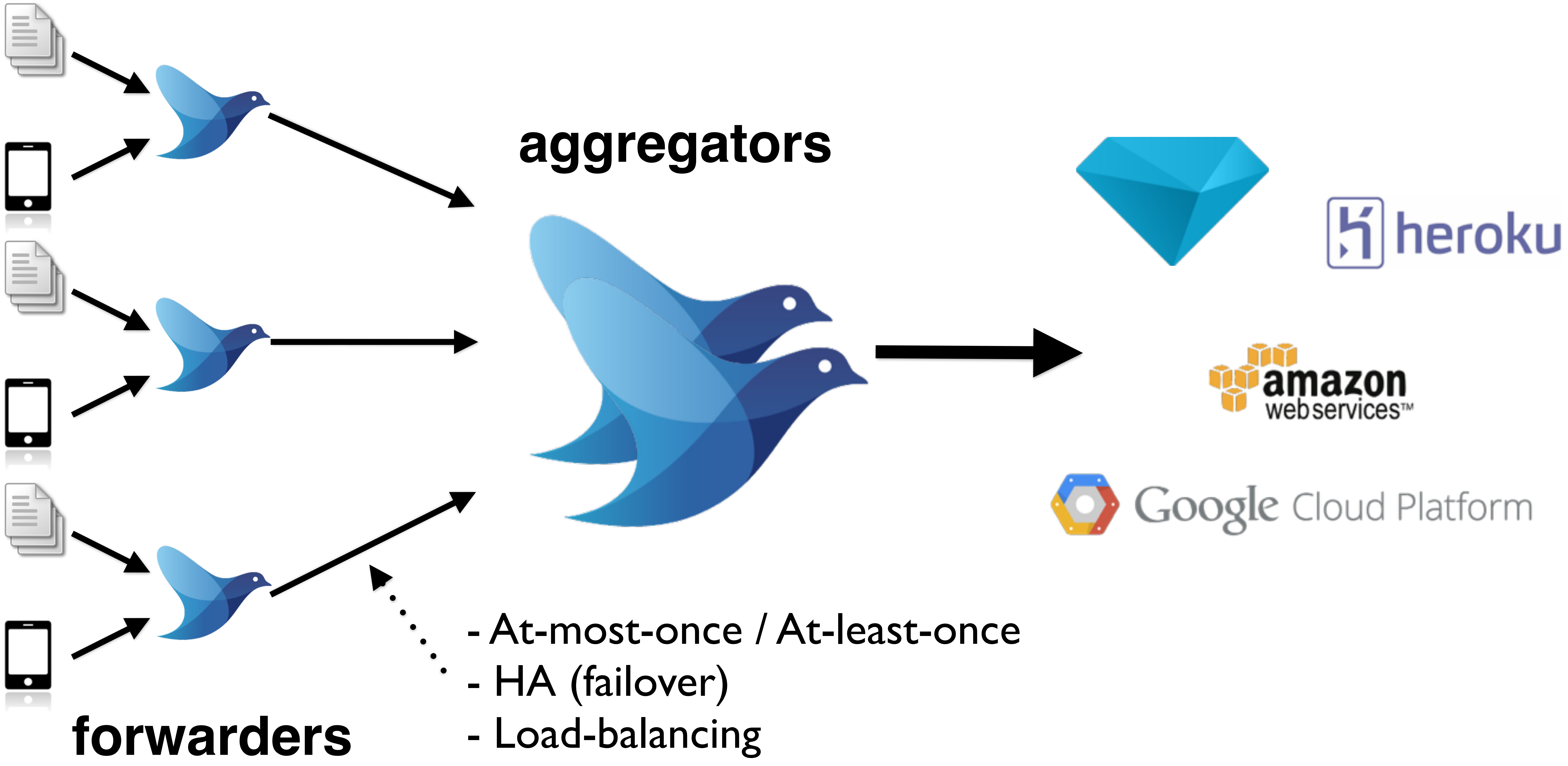


```
# logs from a file
<source>
  @type tail
  path /var/log/httpd.log
  pos_file /tmp/pos_file
  <parse>
    @type apache2
  </parse>
  tag app.access
</source>

# logs from client libraries
<source>
  @type forward
  port 24224
</source>
```

```
# store logs to ES and HDFS
<match app.*>
  @type copy
  <store>
    @type elasticsearch
    logstash_format true
  </store>
  <store>
    @type webhdfs
    host namenode
    port 50070
    path /path/on/hdfs/
  </store>
</match>
```

Multi-tier Forwarding



aggregators

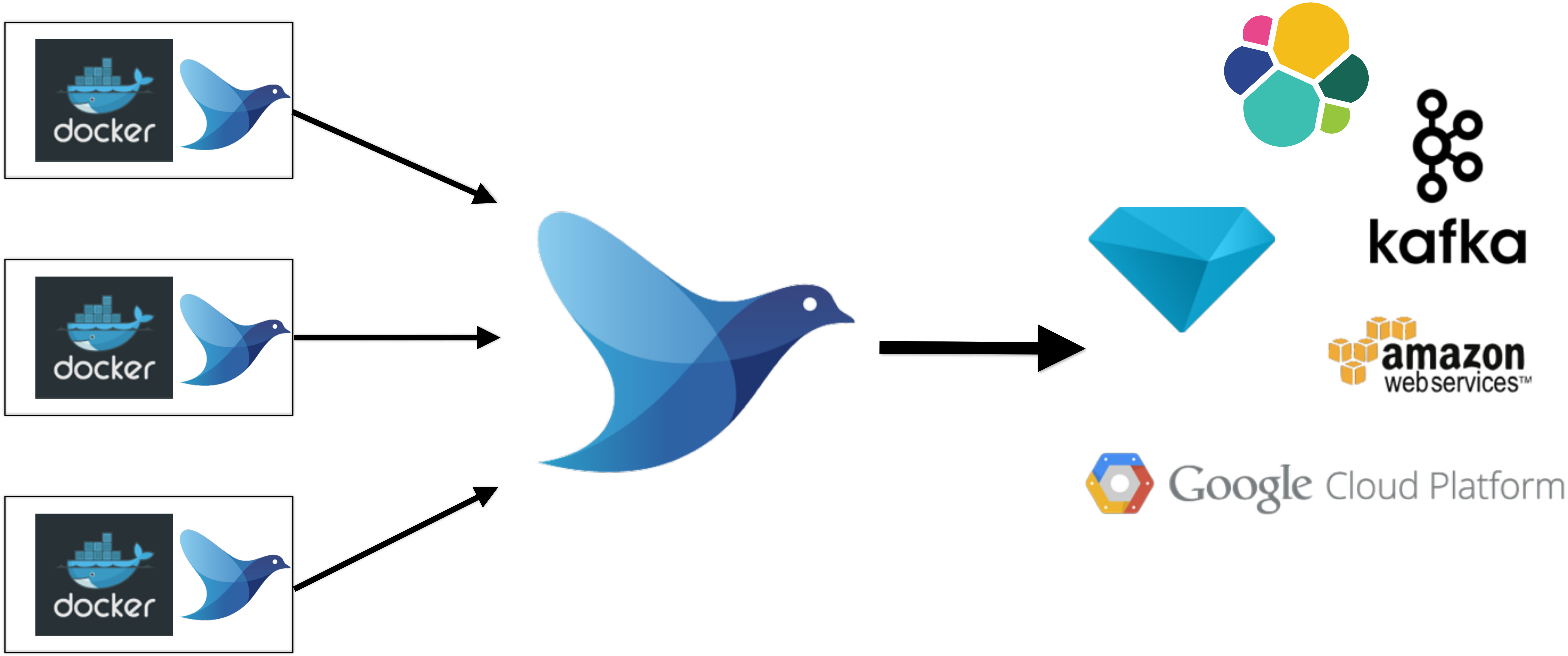
forwarders

- At-most-once / At-least-once
- HA (failover)
- Load-balancing

https://www.slideshare.net/repeatedly/fluentd-and-distributed-logging-at-kubecon

Container and Kubernetes

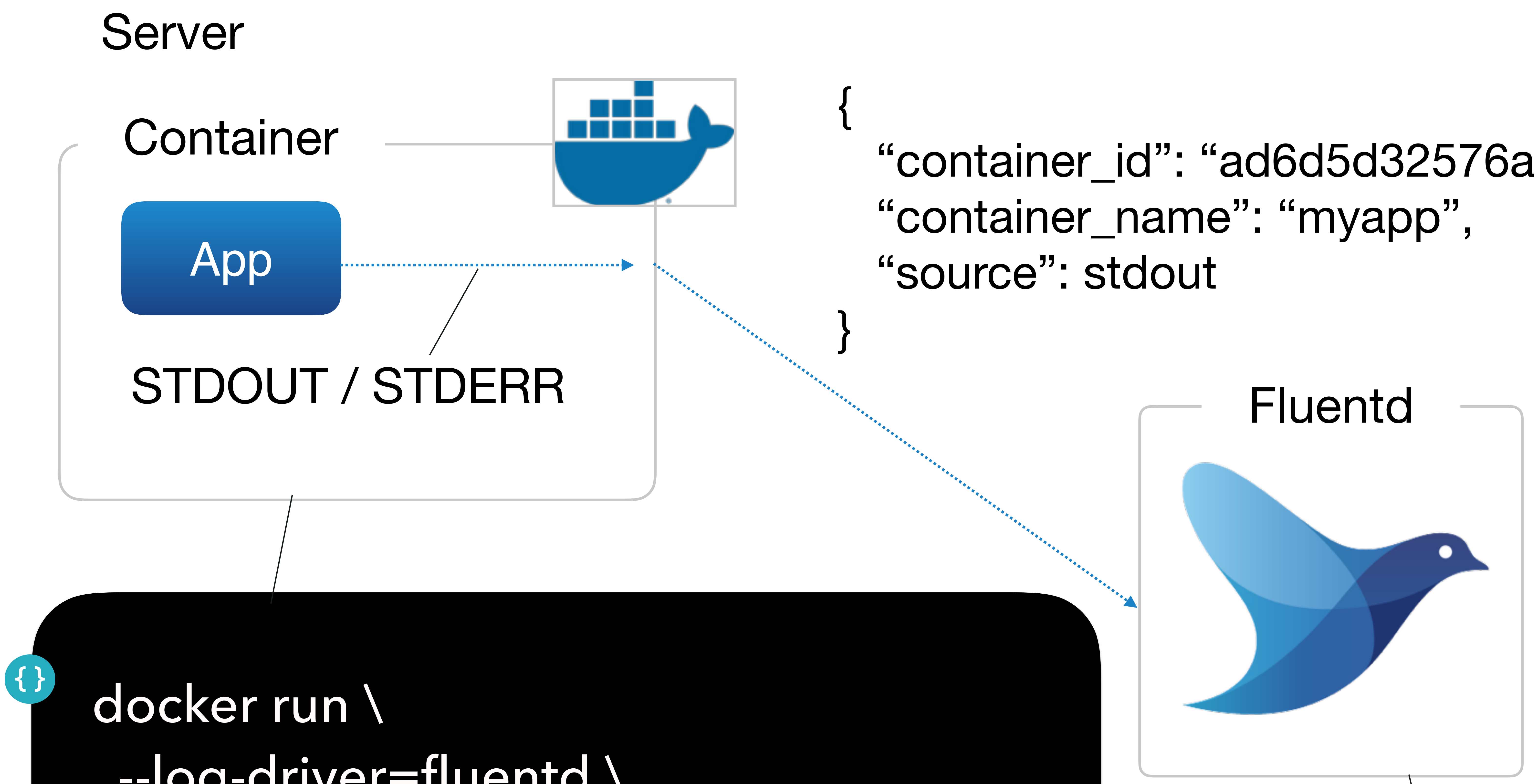
Container Logging



Resources

- **Docker : fluentd-docker-image**
  - Alpine / Debian images
- **Kubernetes : fluentd-kubernetes-daemonset**
  - Debian images
  - Some built-in destinations, es, kafka, graylog, etc…
- **Helm chart**
  - https://github.com/helm/charts/tree/master/stable/fluentd

Docker logging with --log-driver=fluentd

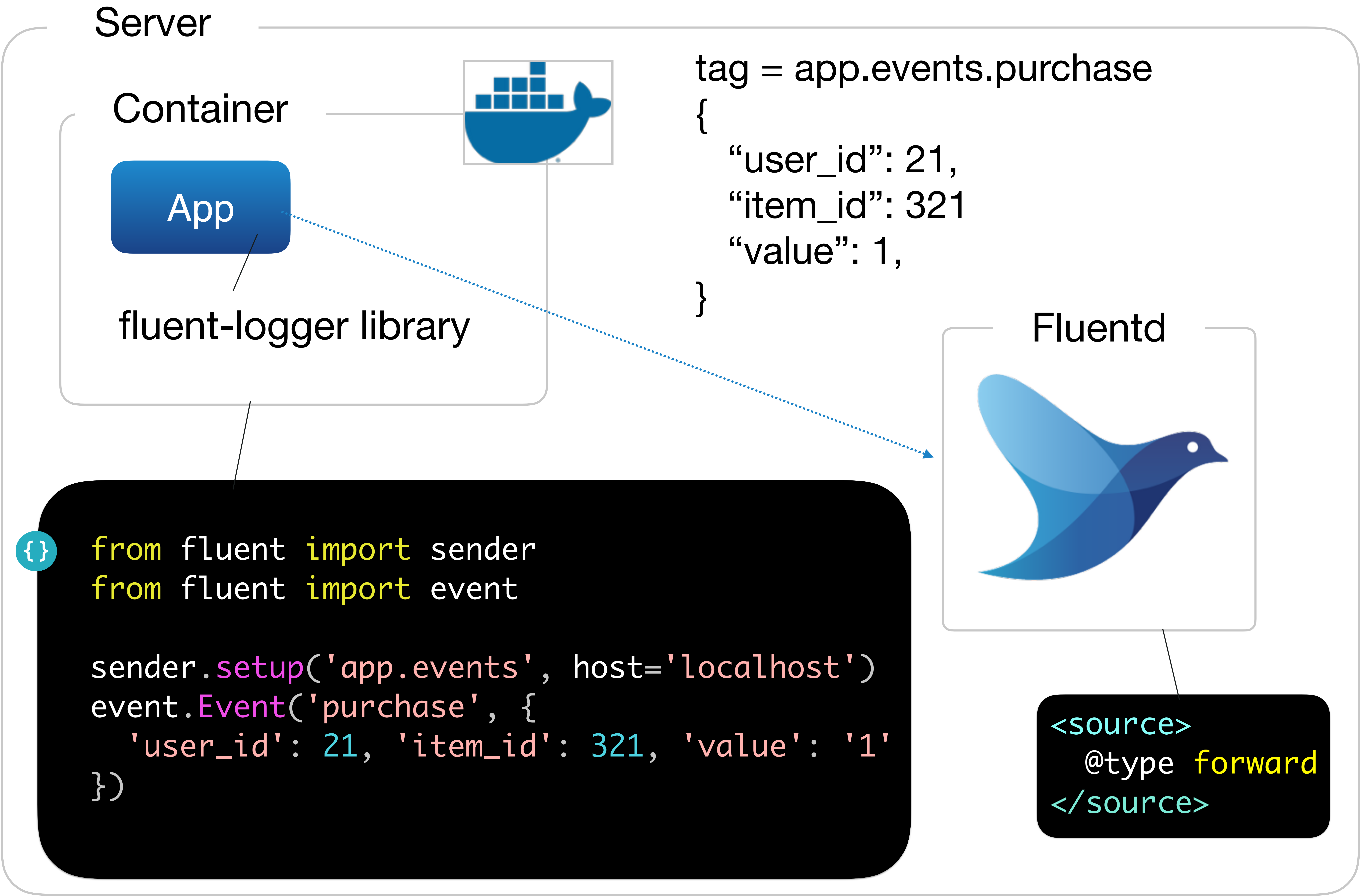


```
{
   “container_id”: “ad6d5d32576a”,
   “container_name”: “myapp”,
   “source”: stdout
}
```

```
docker run \
  --log-driver=fluentd \
  --log-opt \
    fluentd-address=localhost:24224
```

```
<source>
  @type forward
</source>
```

Data collection with fluent-logger



Server

Container

App

fluent-logger library

tag = app.events.purchase
{
  “user_id”: 21,
  “item_id”: 321
  “value”: 1,
}

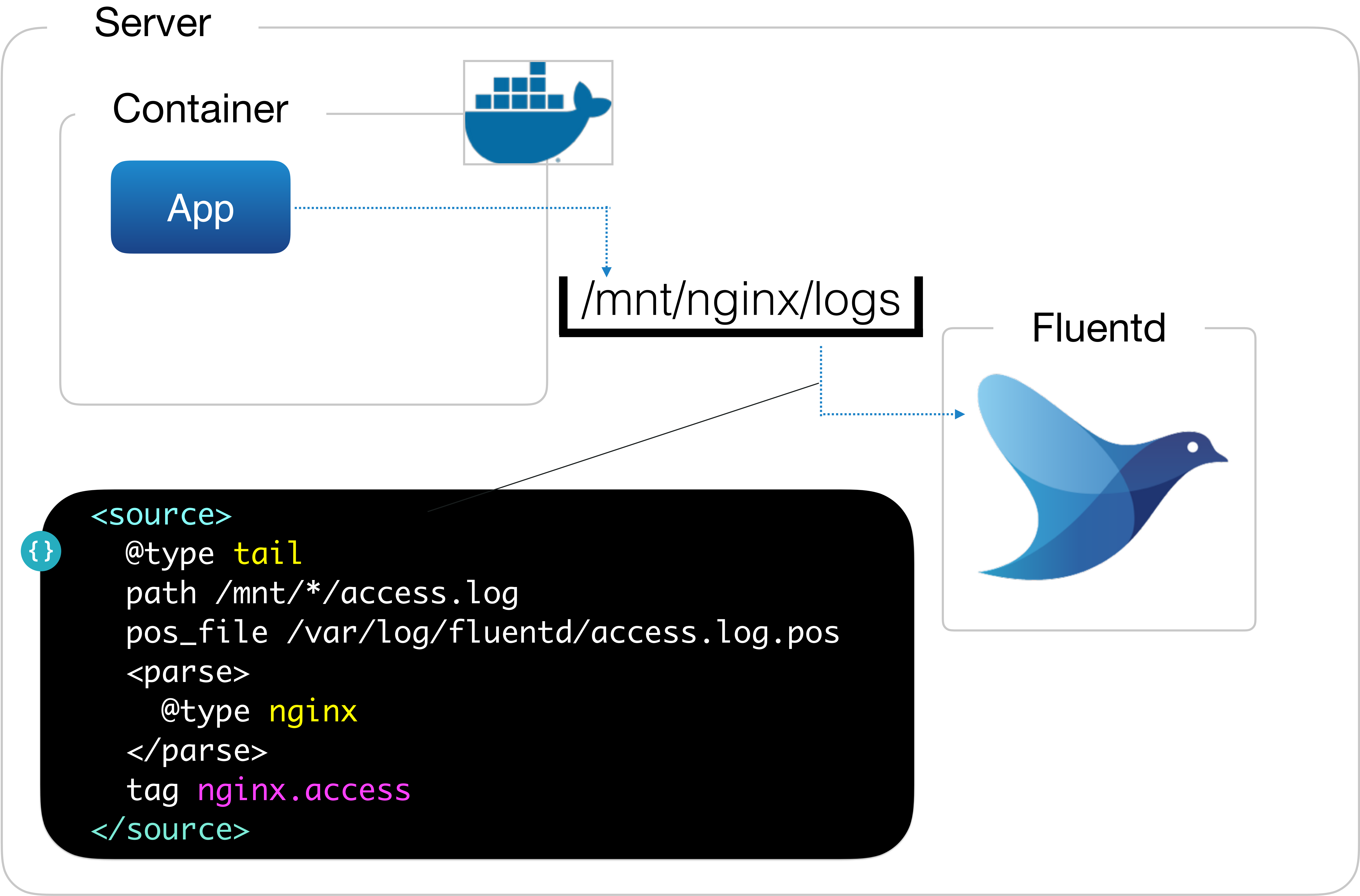
Fluentd

```
from fluent import sender
from fluent import event

sender.setup('app.events', host='localhost')
event.Event('purchase', {
  'user_id': 21, 'item_id': 321, 'value': '1'
})
```

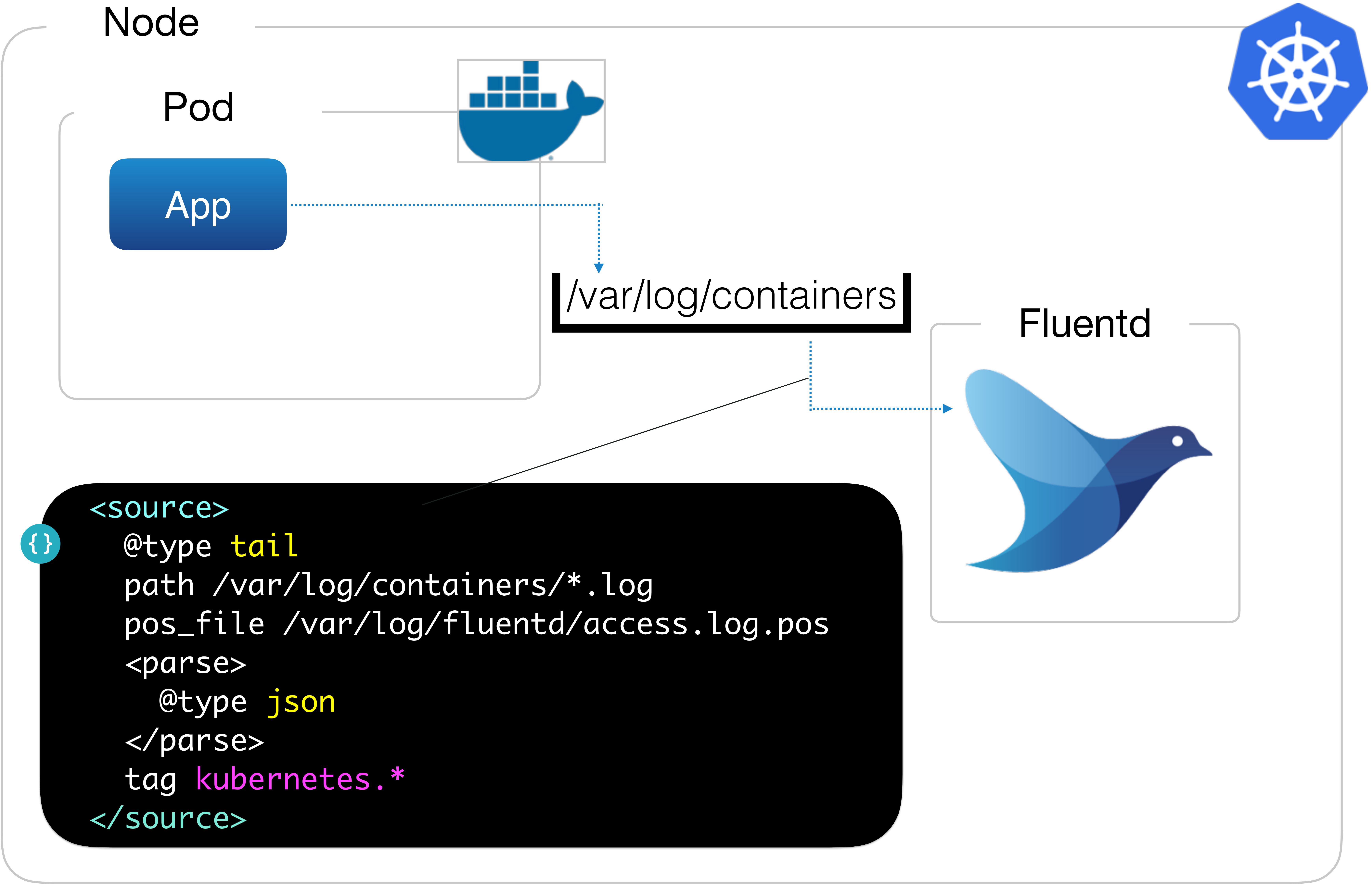
```
<source>
  @type forward
</source>
```

Shared data volume and tailing



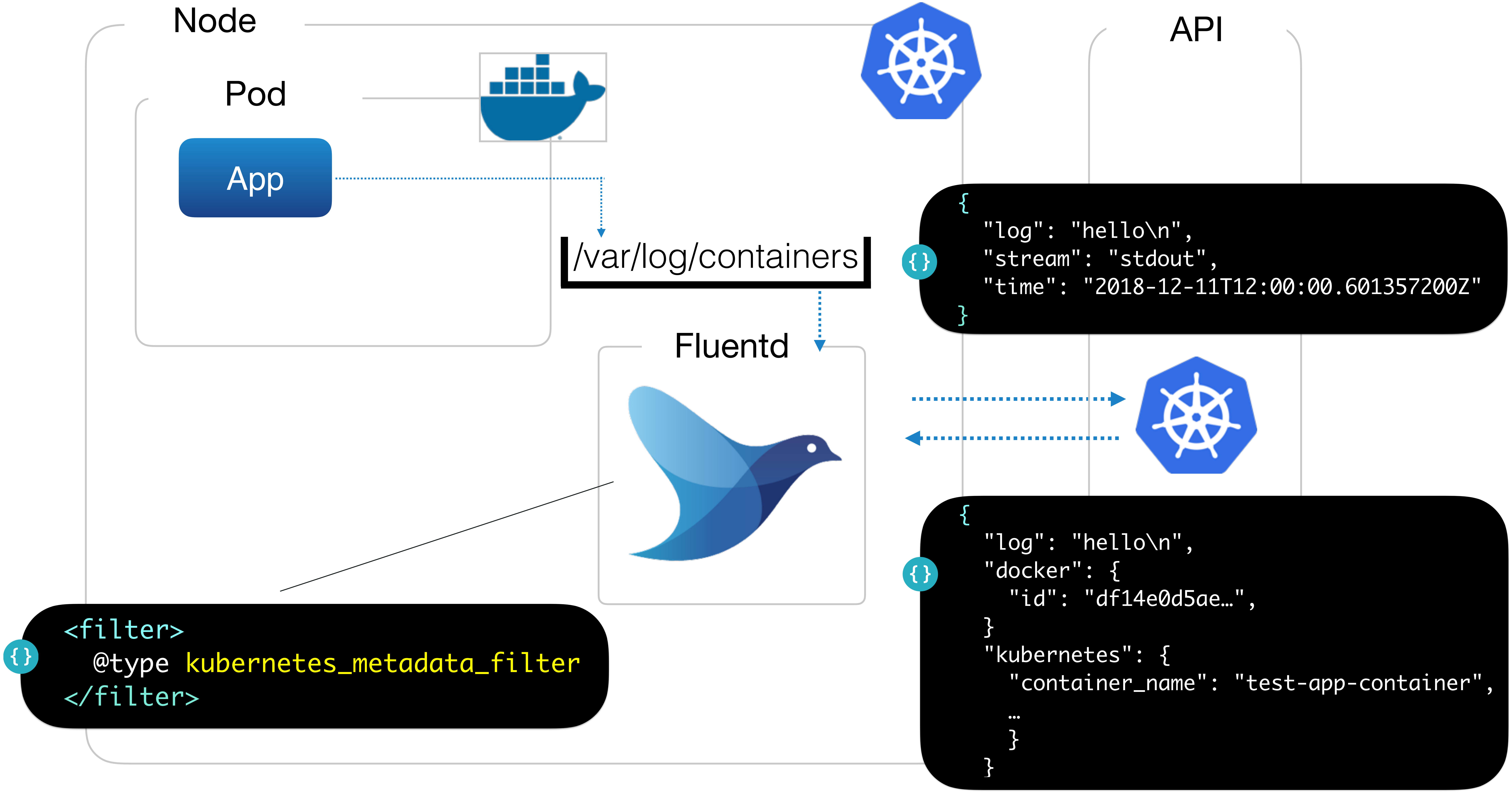
```
<source>
  @type tail
  path /mnt/*/access.log
  pos_file /var/log/fluentd/access.log.pos
  <parse>
    @type nginx
  </parse>
  tag nginx.access
</source>
```

Kubernetes Daemonset



```
<source>
  @type tail
  path /var/log/containers/*.log
  pos_file /var/log/fluentd/access.log.pos
  <parse>
    @type json
  </parse>
  tag kubernetes.*
</source>
```

Kubernetes Daemonset & metadata



```
{
  "log": "hello\n",
  "stream": "stdout",
  "time": "2018-12-11T12:00:00.601357200Z"
}
```

```
{
  "log": "hello\n",
  "docker": {
    "id": "df14e0d5ae…",
  }
  "kubernetes": {
    "container_name": "test-app-container",
    …
    }
  }
```

```
<filter>
  @type kubernetes_metadata_filter
</filter>
```

Container Logging approach summary

- Collect log messages with docker
  - --log-driver=fluentd
- Application data/metrics
  - fluent-logger
- Access logs, logs from middleware
  - Shared data volume with in_tail
- Kubernetes Daemonset
  - Collect container logs from /var/log/containers/*
  - Add kubernetes metadata to logs

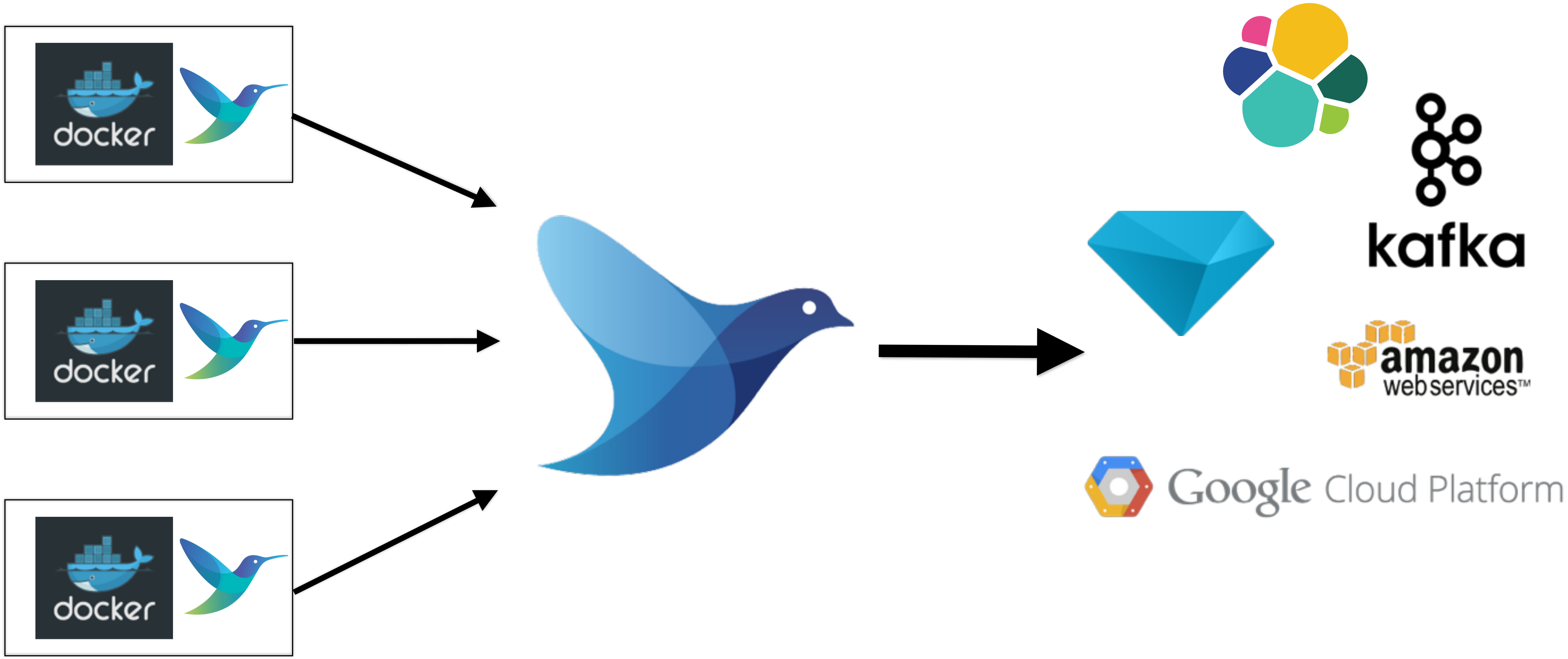
Fluent-bit

Fluentd and Fluent-bit

	Fluentd	Fluent-bit
Implementation	Ruby + C	C
Focus	Flexibility and Robustness	Performance and footprint
Design	Pluggable	Pluggable
Target	Forwarder / Aggregator	Forwarder / Device

Forward logs from fluent-bit to fluentd is popular pattern

Container Logging with fluent-bit



Enjoy logging!