CNCon / KubeCon Europe 2018

Fluentd Project Intro

Masahiro Nakagawa
Senior Software Engineer
TREASURE DATA

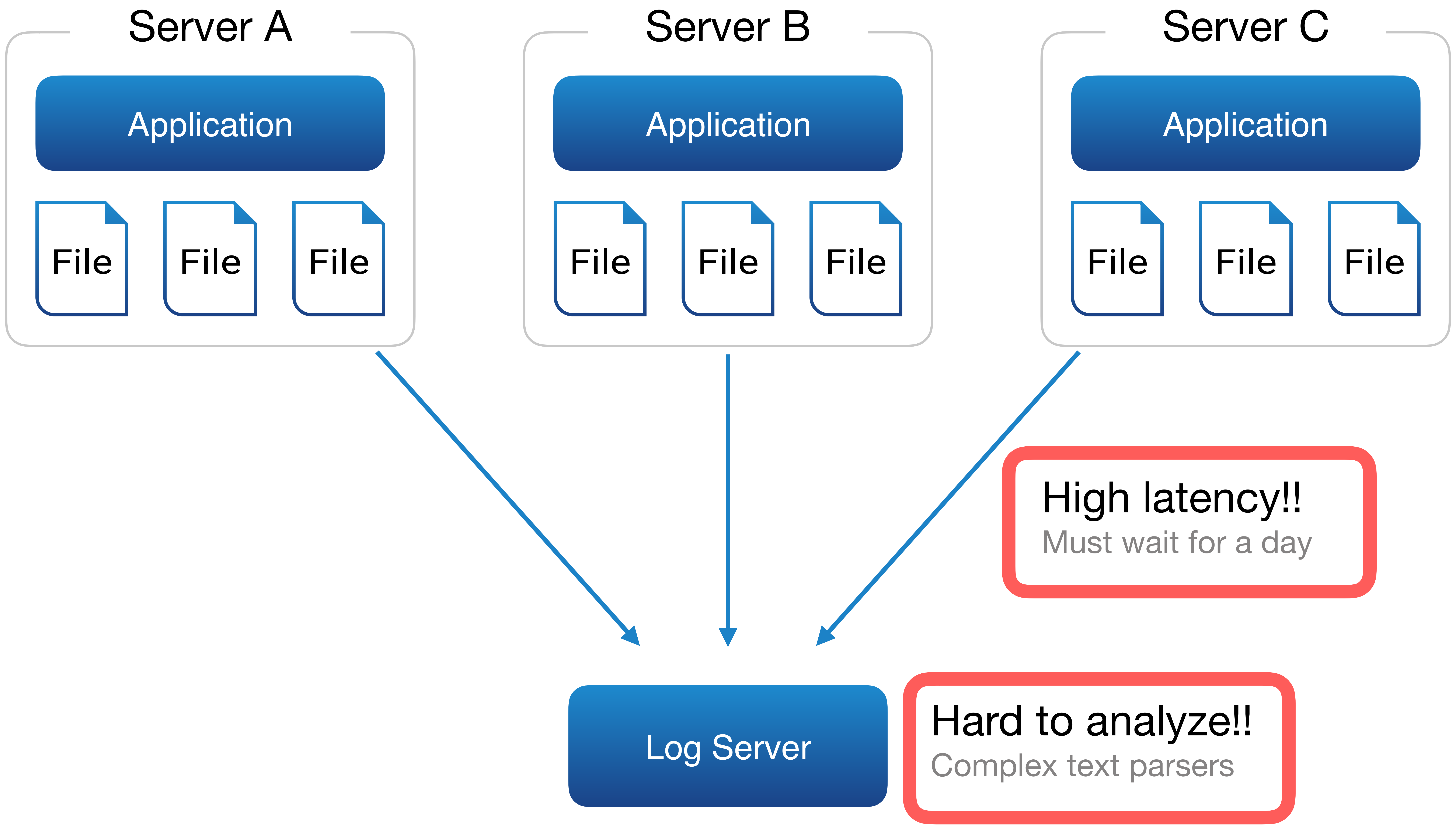
CLOUD NATIVE COMPUTING FOUNDATION

Fluentd overview

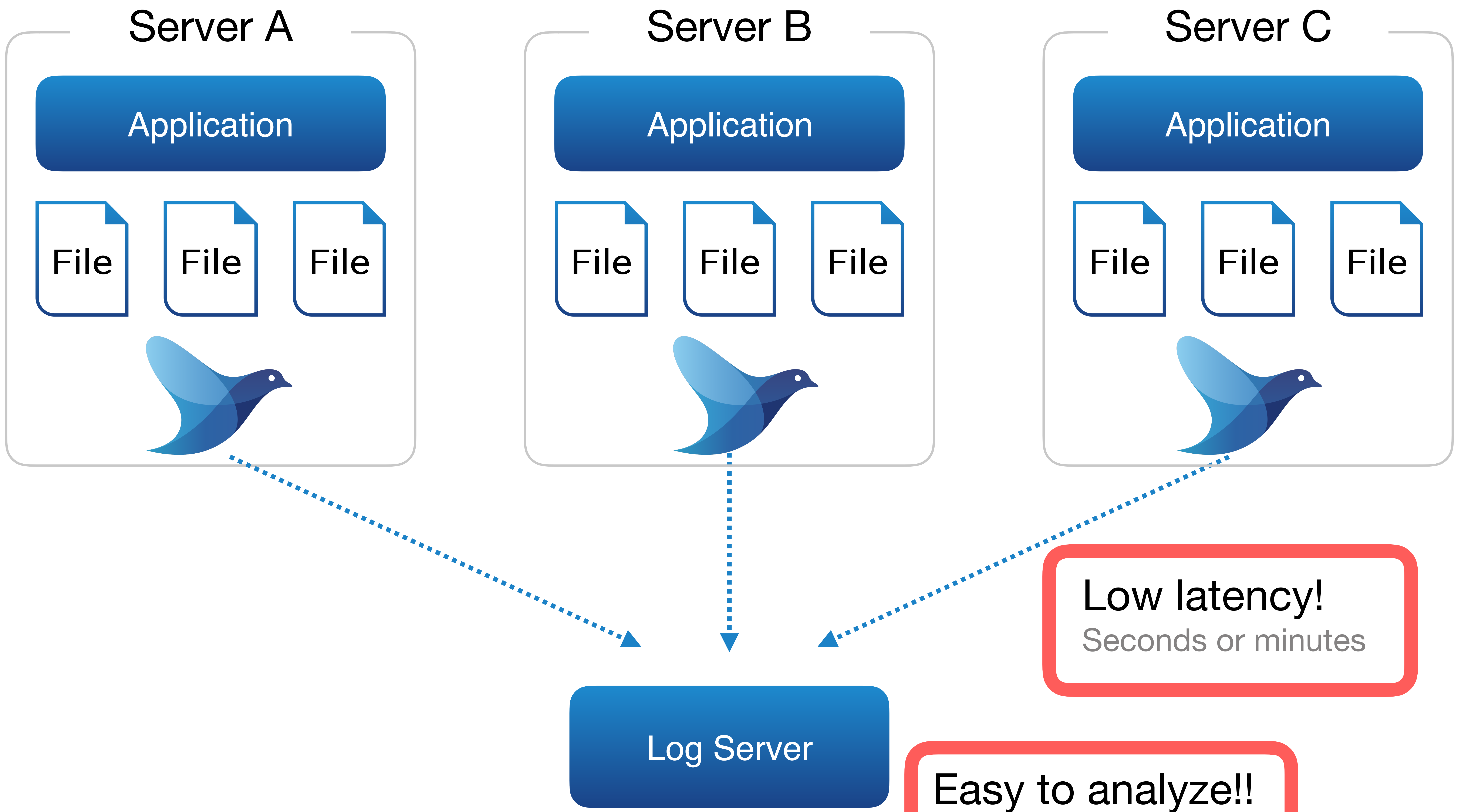
What’s Fluentd

- **Streaming data collector for unified logging**
    - Simple core + plugins
- **Gem based various plugins**
    - Follow Ruby’s standard way
- **Several installation ways**
    - https://docs.fluentd.org/v1.0/categories/installation
- **Latest version**: v1.2.0, released yesterday.
- **Logging part in CNCF**

Log collection with traditional sync batch model



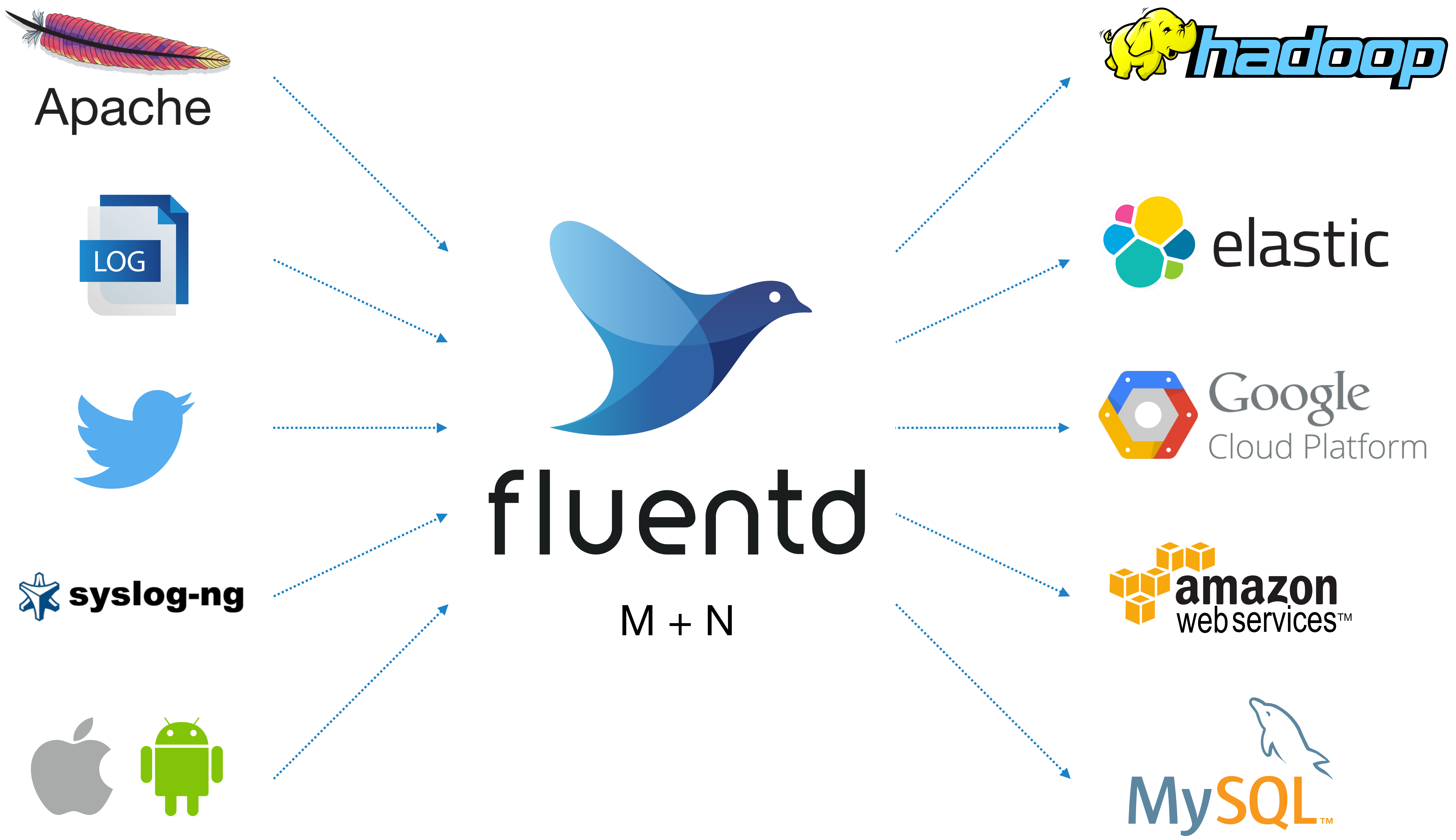
Streaming way with Fluentd



Low latency!
Seconds or minutes

Easy to analyze!!
Parsed and formatted

Unified logging layer



Fluentd Architecture

Design

Core

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

Plugins

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

Event structure

Time

- Nano-second unit
- from logs

Tag

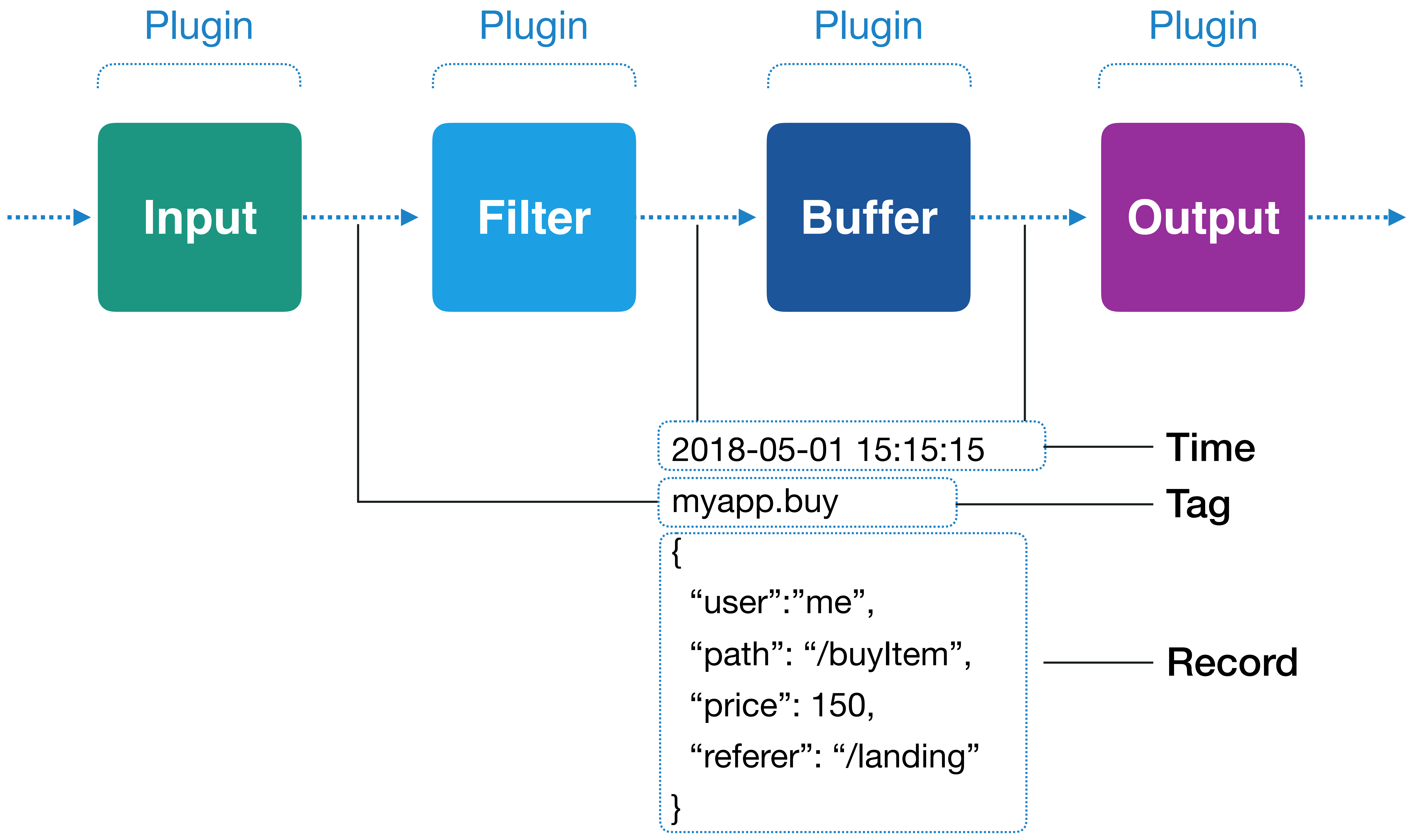
- for message routing
- Identify data source

Record

- JSON object,
  not raw string

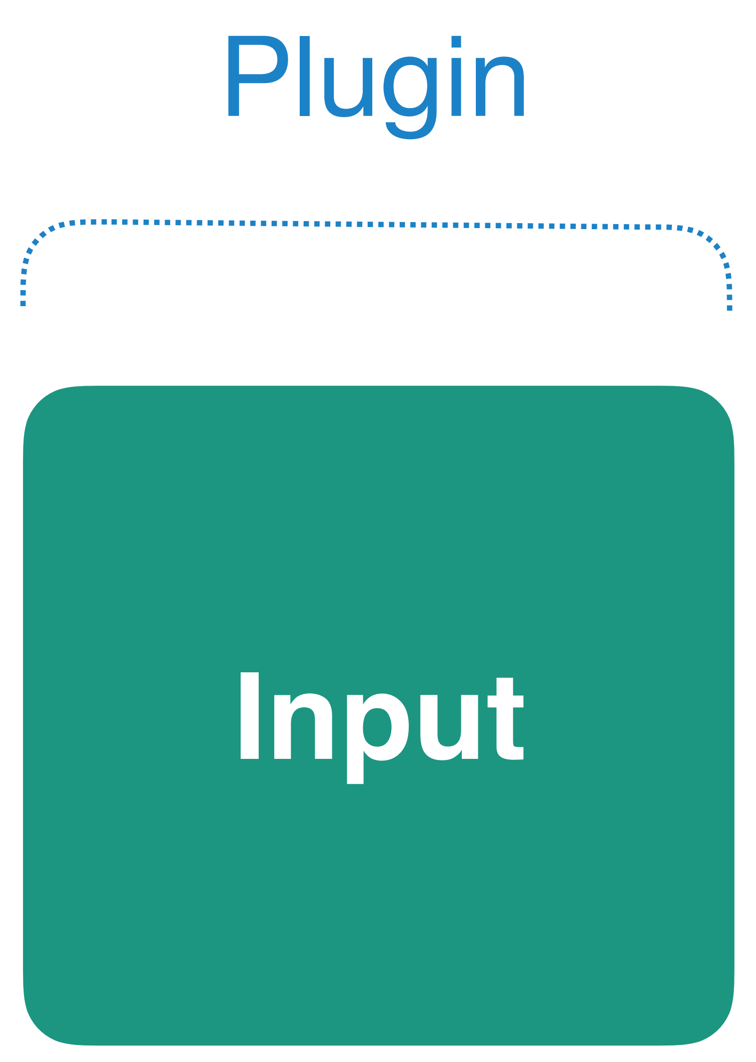
```
{
  “str_field”:”hey”,
  “num_field”: 100,
  “bool_field”: true,
  “array_field”: [“elem1”, “elem2”]
}
```

Data pipeline (simplified)



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

Architecture: Input Plugins



- ✓ Receive logs
- ✓ Or pull logs from data sources
- ✓ Parse incoming logs for structured logging

HTTP+JSON (in_http)
File tail (in_tail)
Syslog (in_syslog)
…

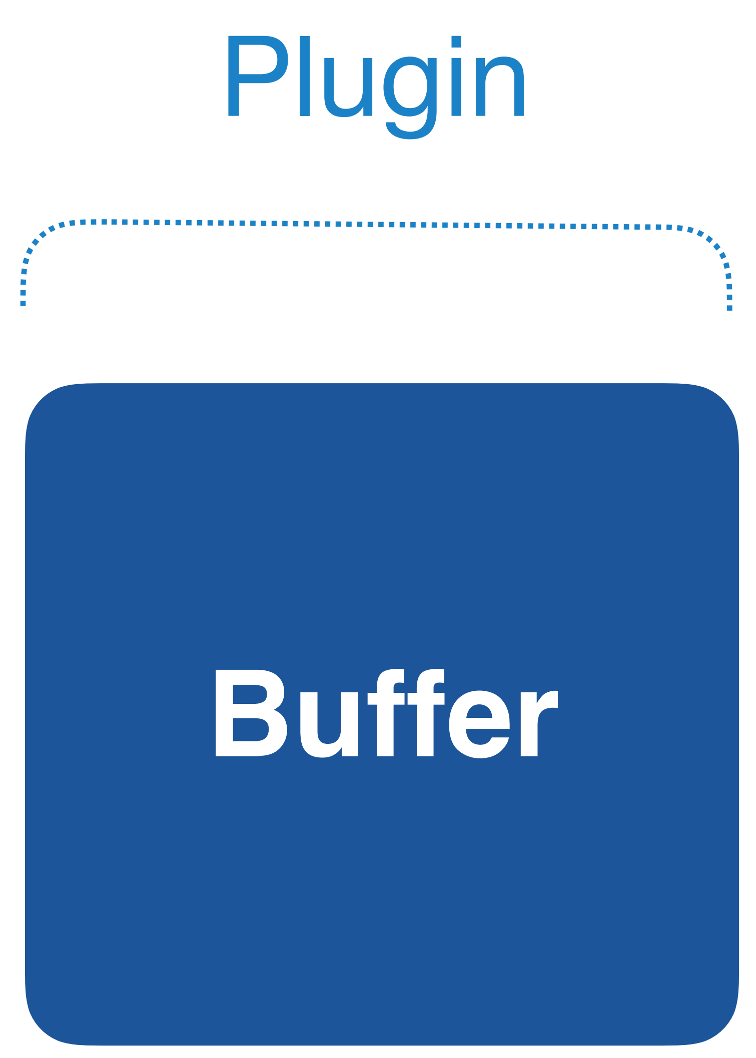
Architecture: Filter Plugins

Plugin
Filter

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

Modify logs (record_transformer)
Filter out logs (grep)
Parse field (parser)
…

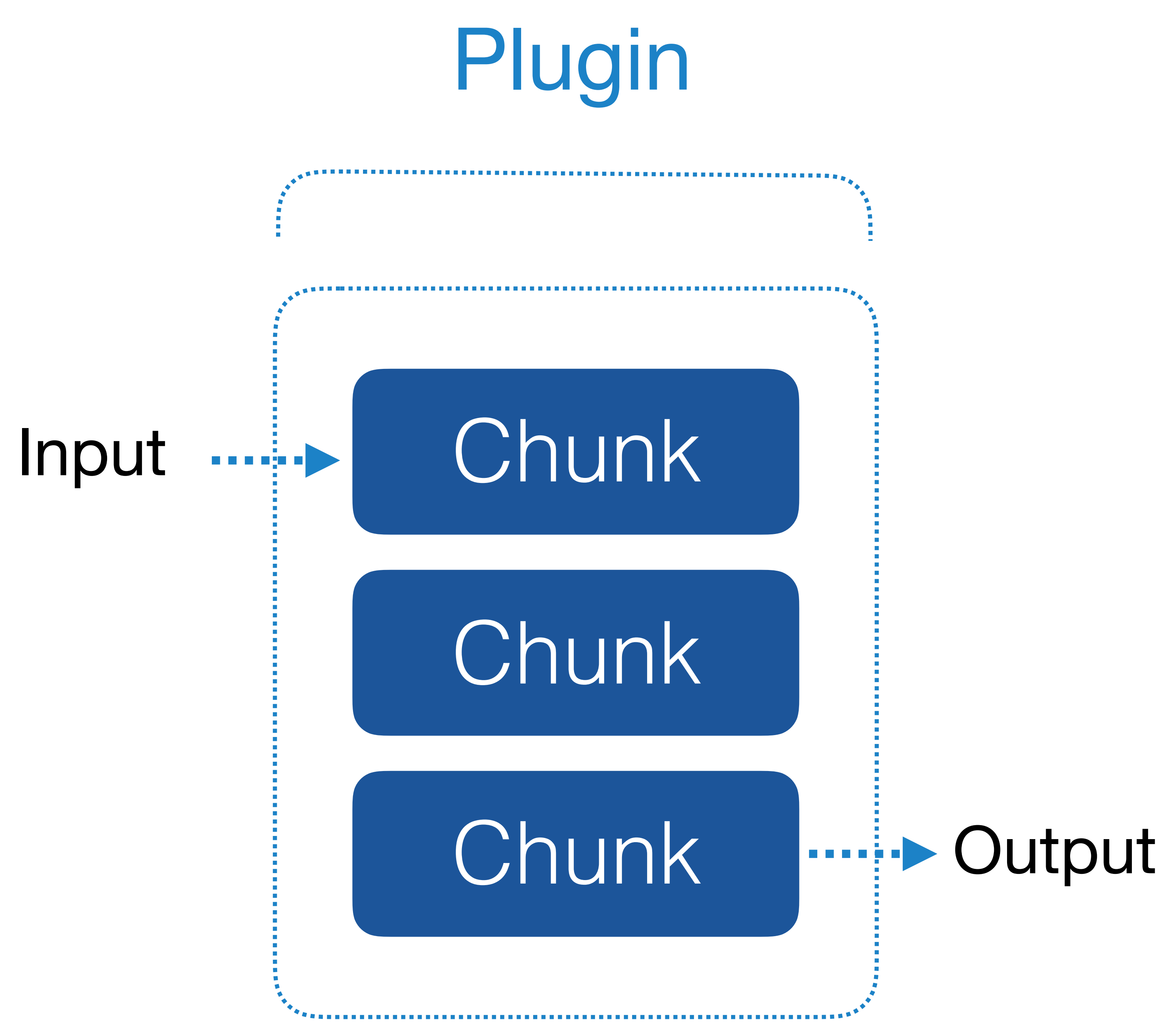
Architecture: Buffer Plugins



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

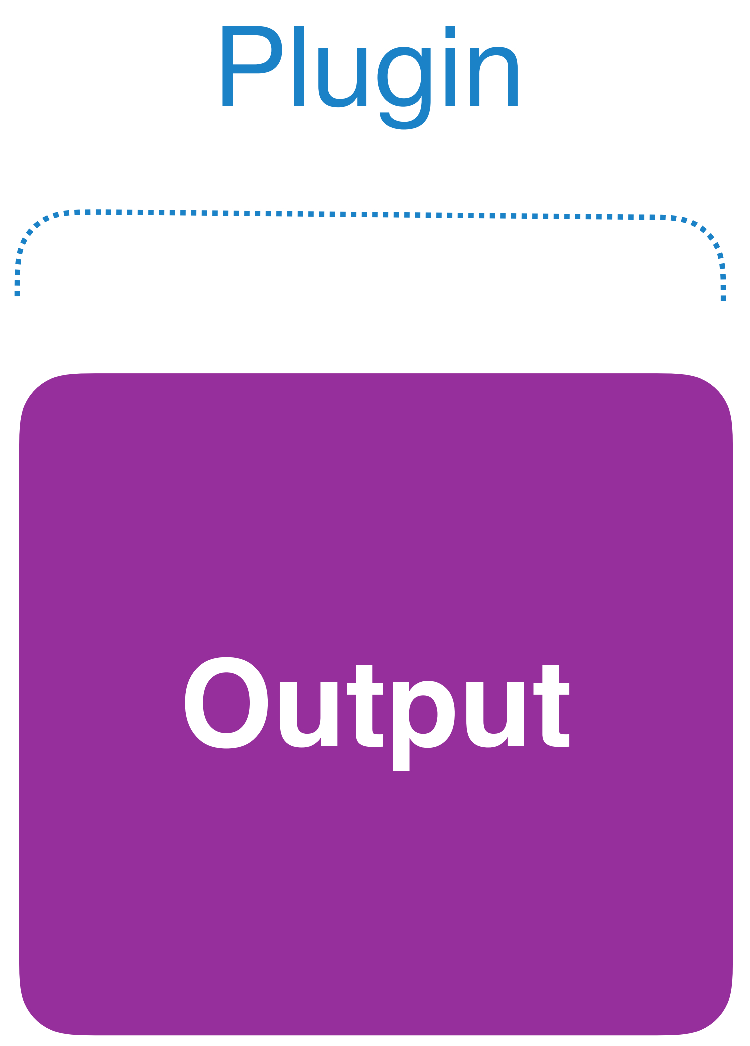
Memory (buf_memory)
File (buf_file)

Architecture: Buffer Plugins



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

Architecture: Output Plugins



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

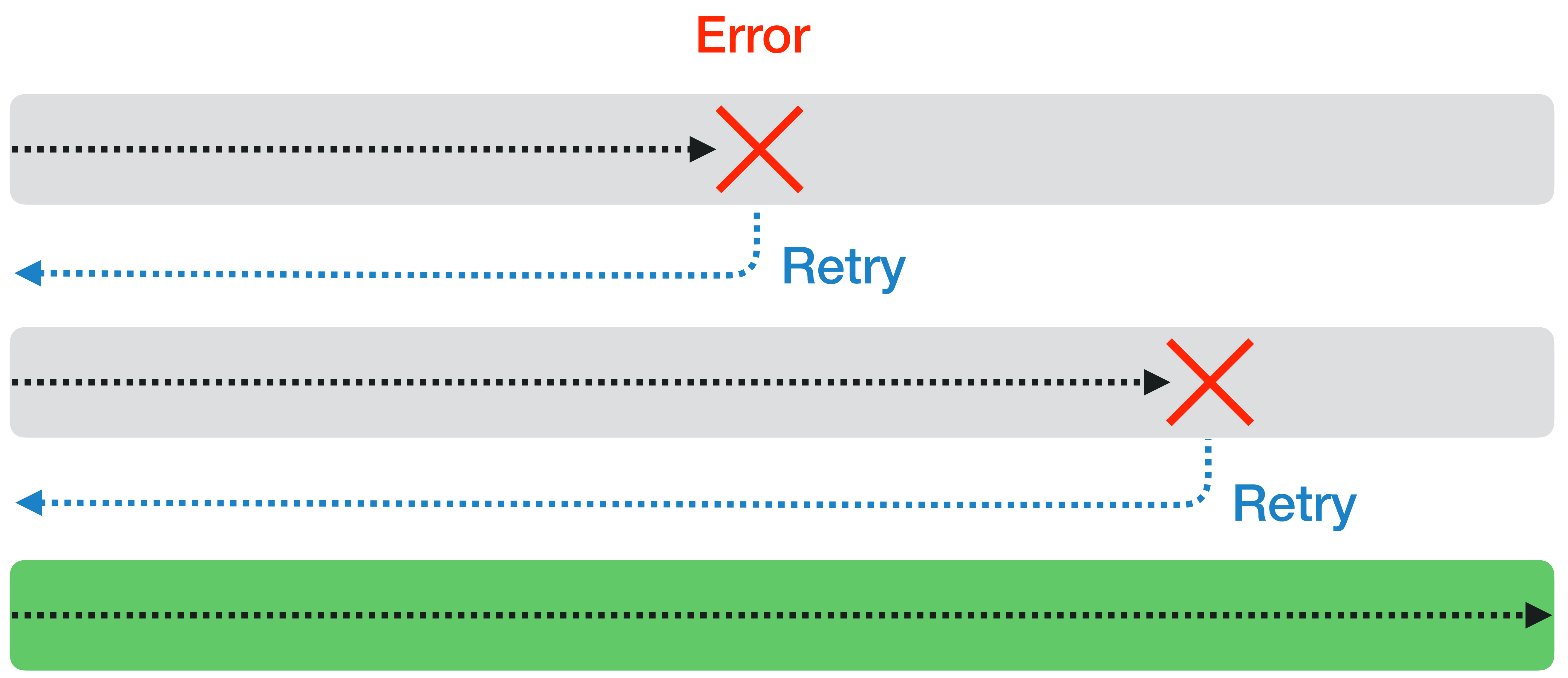
File (out_file)

Amazon S3 (out_s3)

Forward to other fluentd (out_forward)

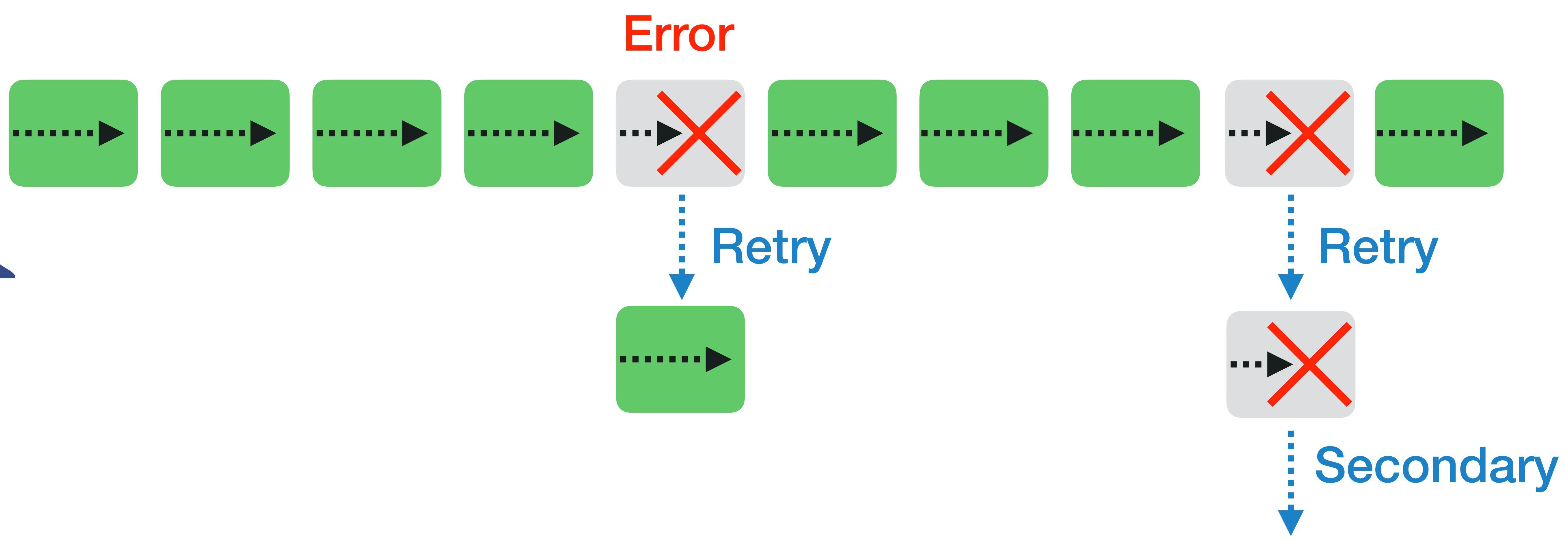
…

Divide & Conquer for retry

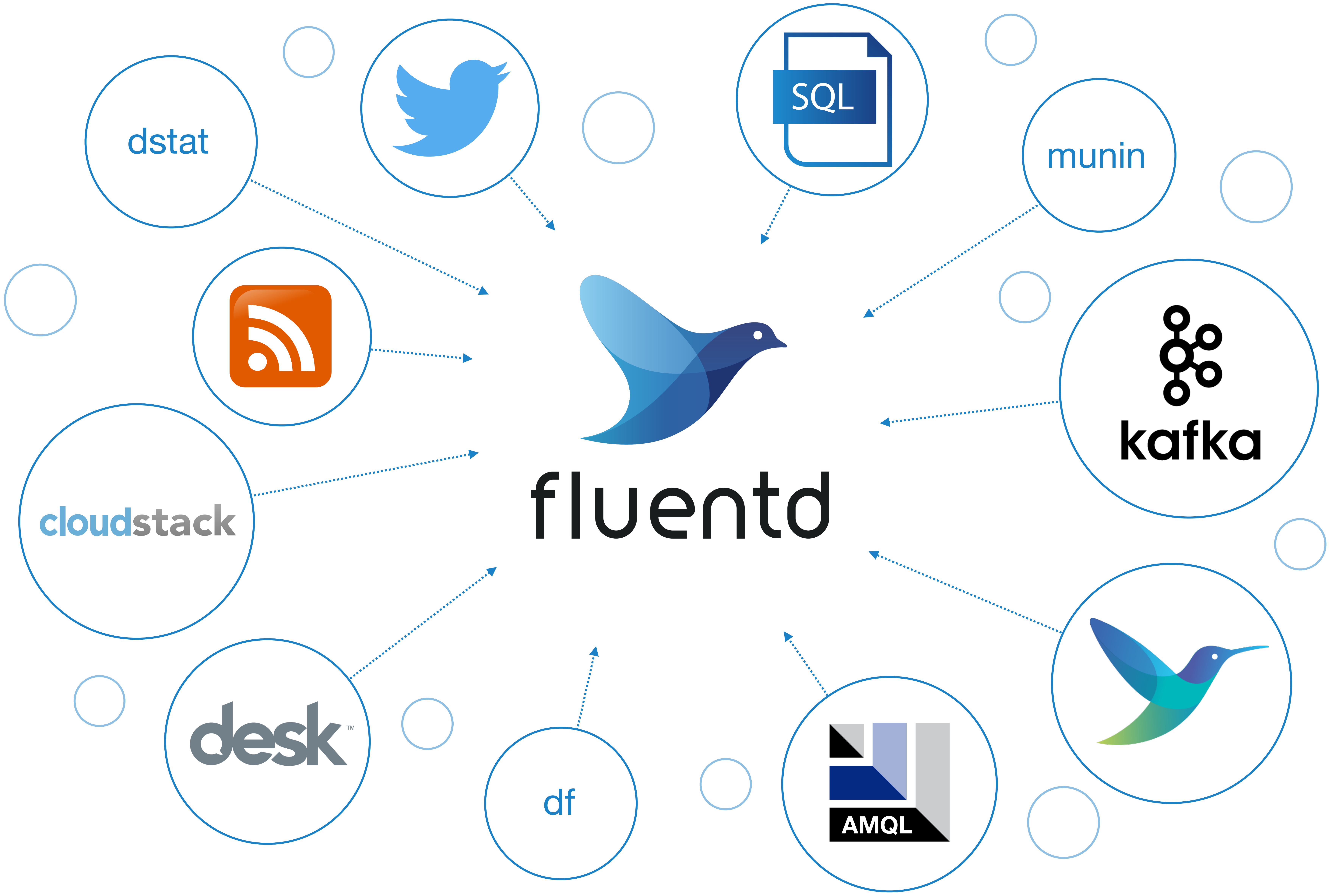


Batch

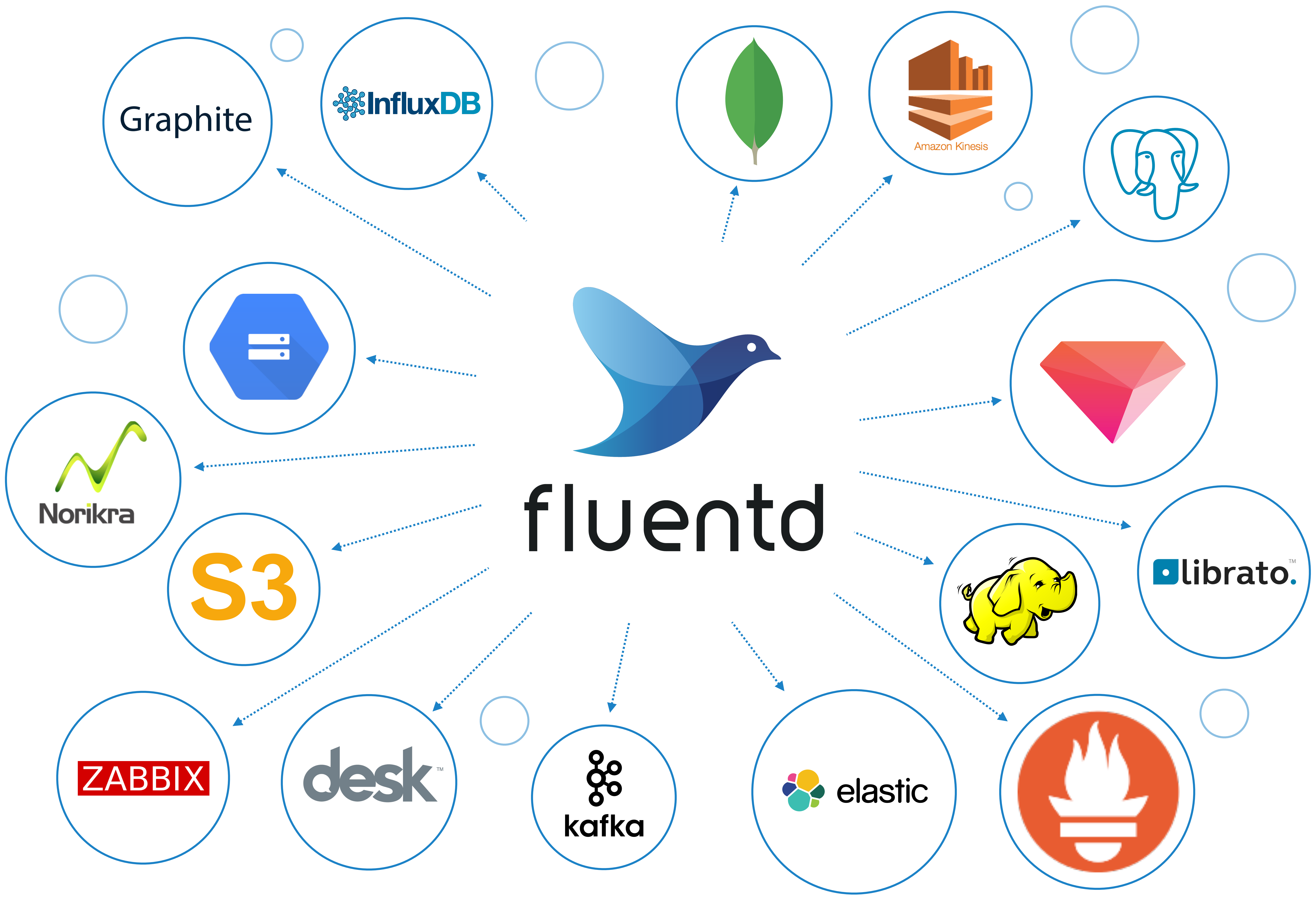
Stream



3rd party input plugins

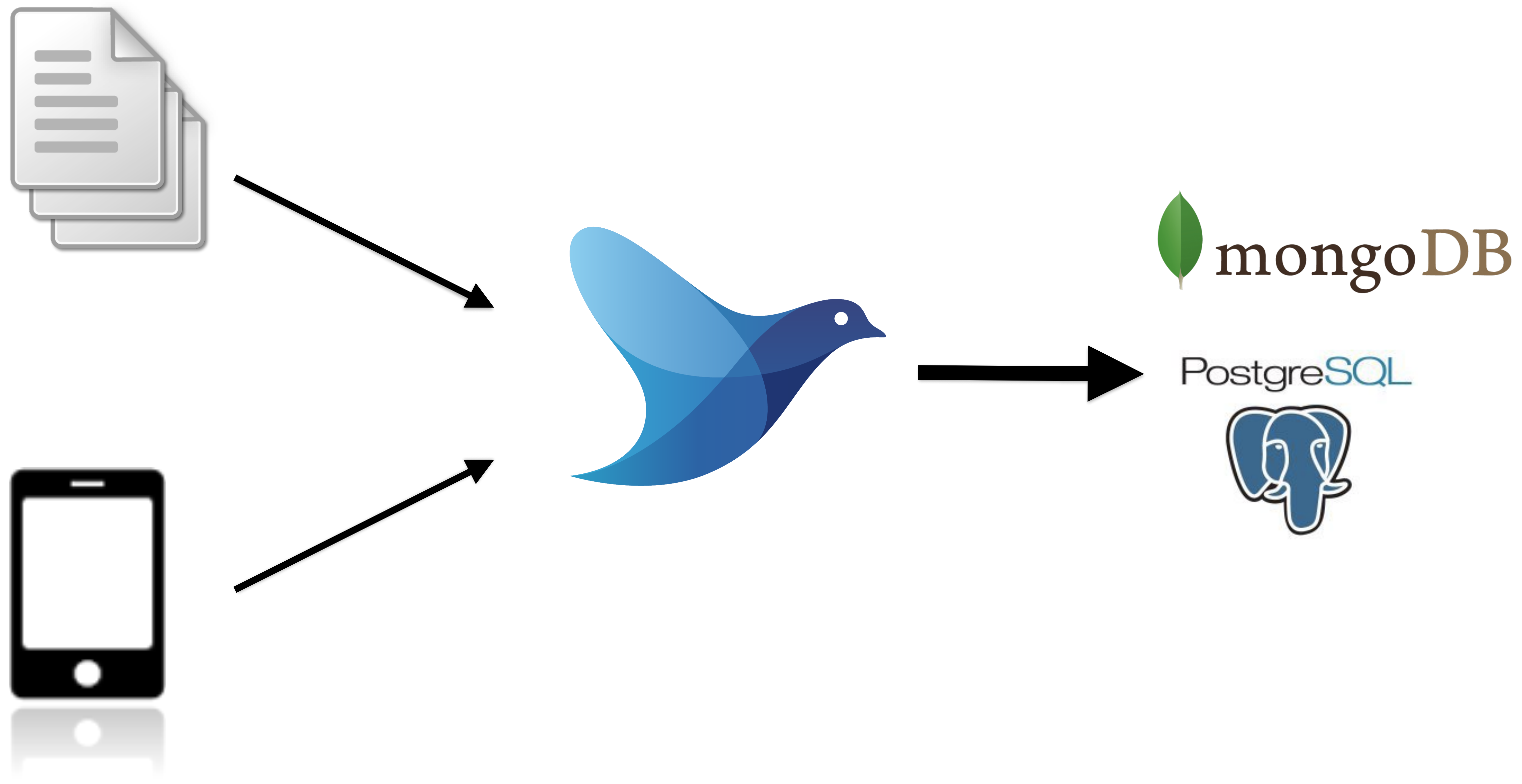


3rd party output plugins



Use cases

Simple forwarding

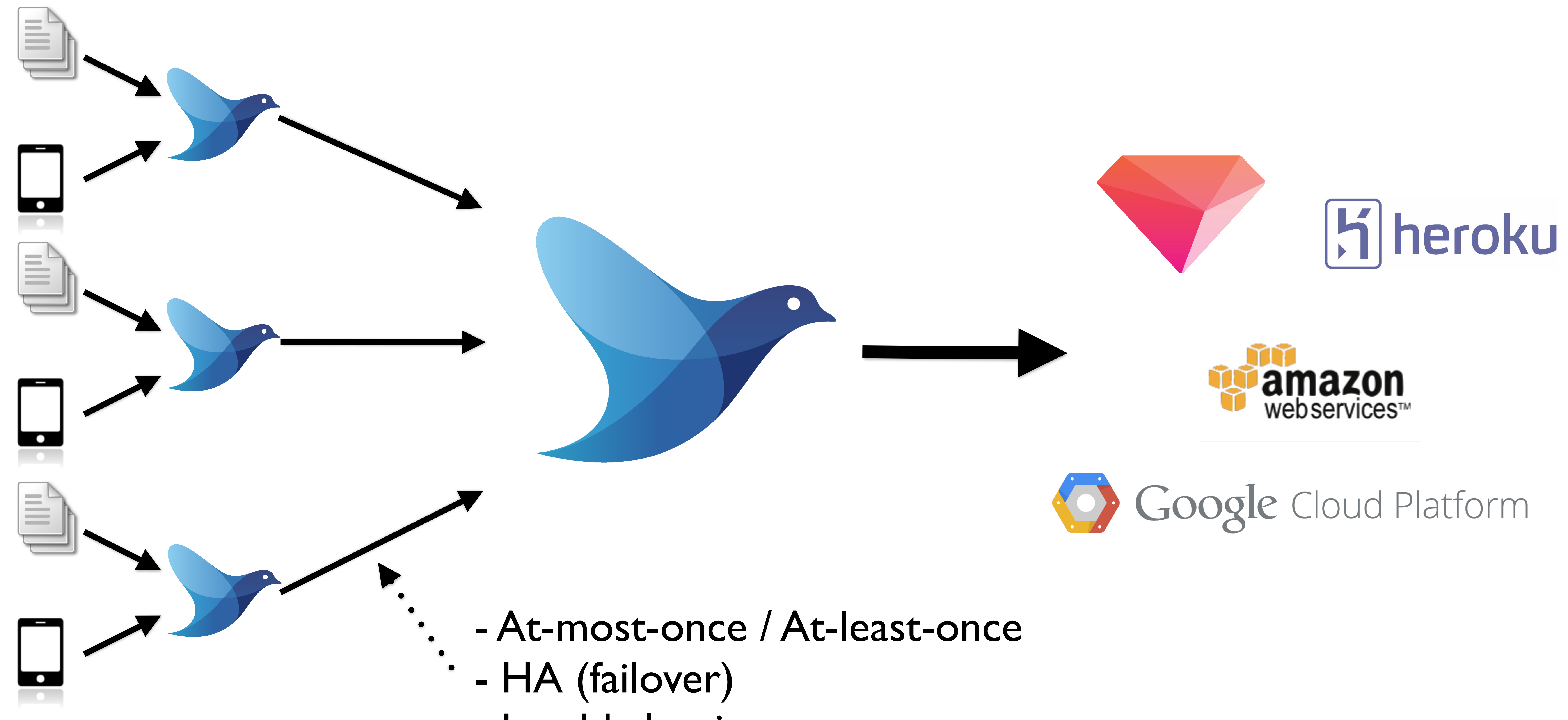


```
# logs from a file
<source>
  @type tail
  path /var/log/httpd.log
  pos_file /tmp/pos_file
  format apache2
  tag backend.apache
</source>

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

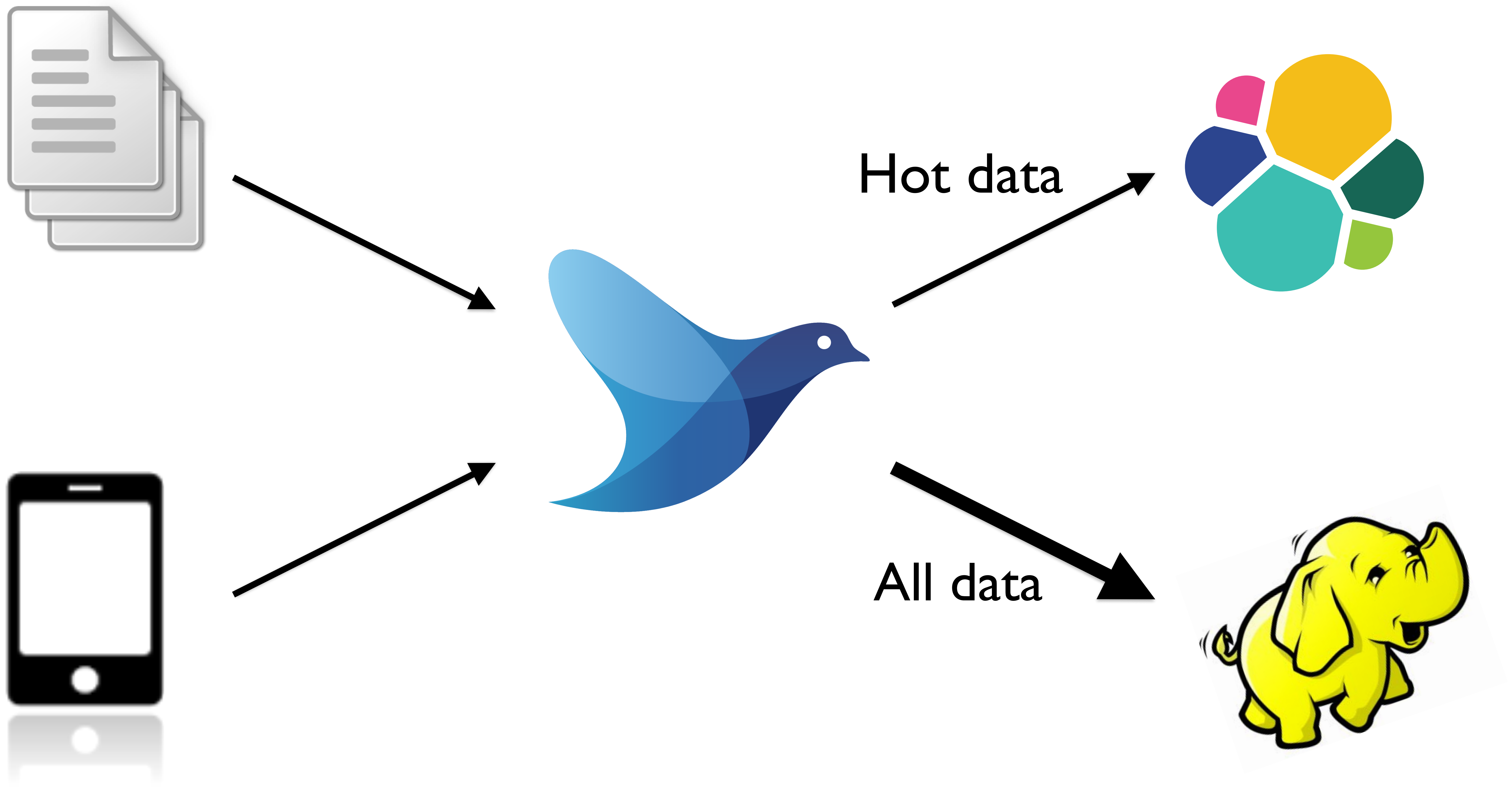
```
# store logs to MongoDB
<match backend.*>
  @type mongo
  database fluent
  collection logs
  <buffer tag>
    @type file
    path /tmp/fluentd/buffer
    flush_interval 30s
  </buffer>
</match>
```

Multi-tier Forwarding



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

Multiple destinations

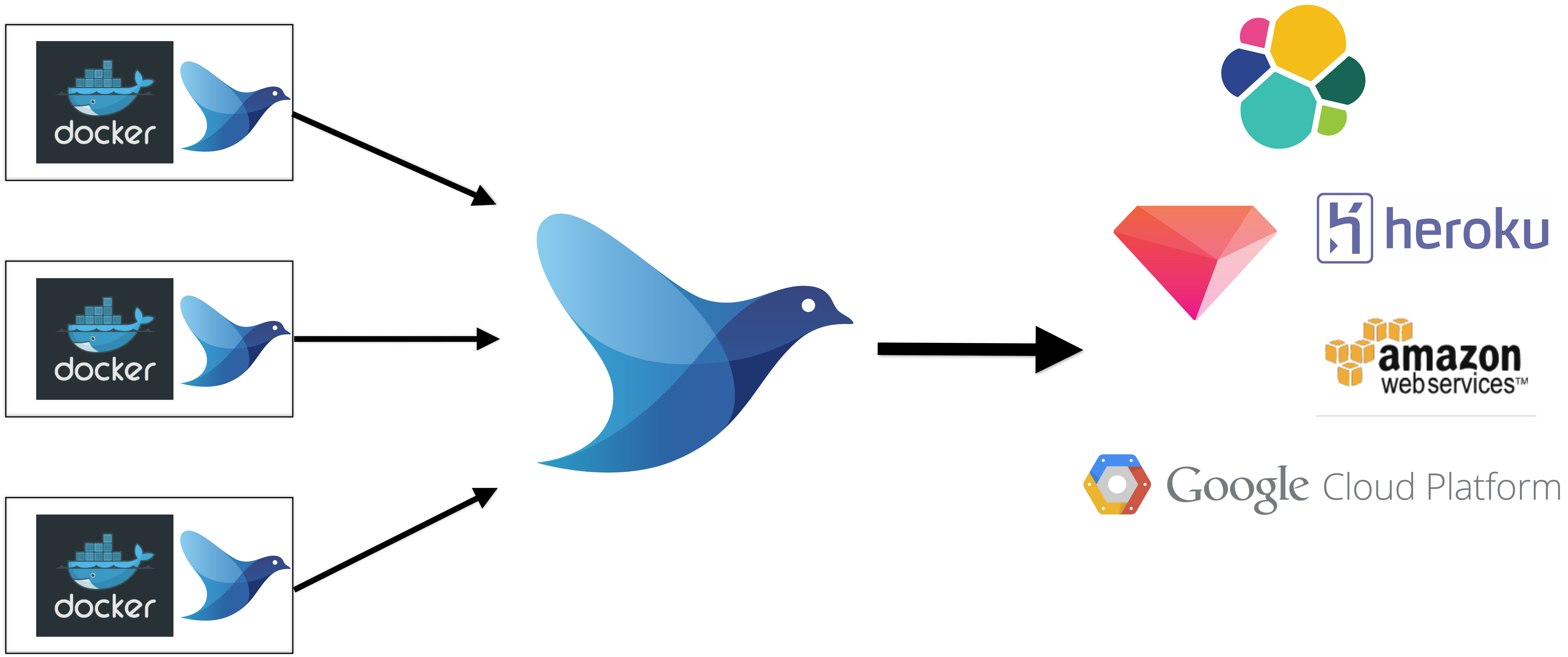


```
# logs from a file
<source>
  @type tail
  path /var/log/httpd.log
  pos_file /tmp/pos_file
  <parse>
    @type apache2
  </parse>
  tag web.access
</source>

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

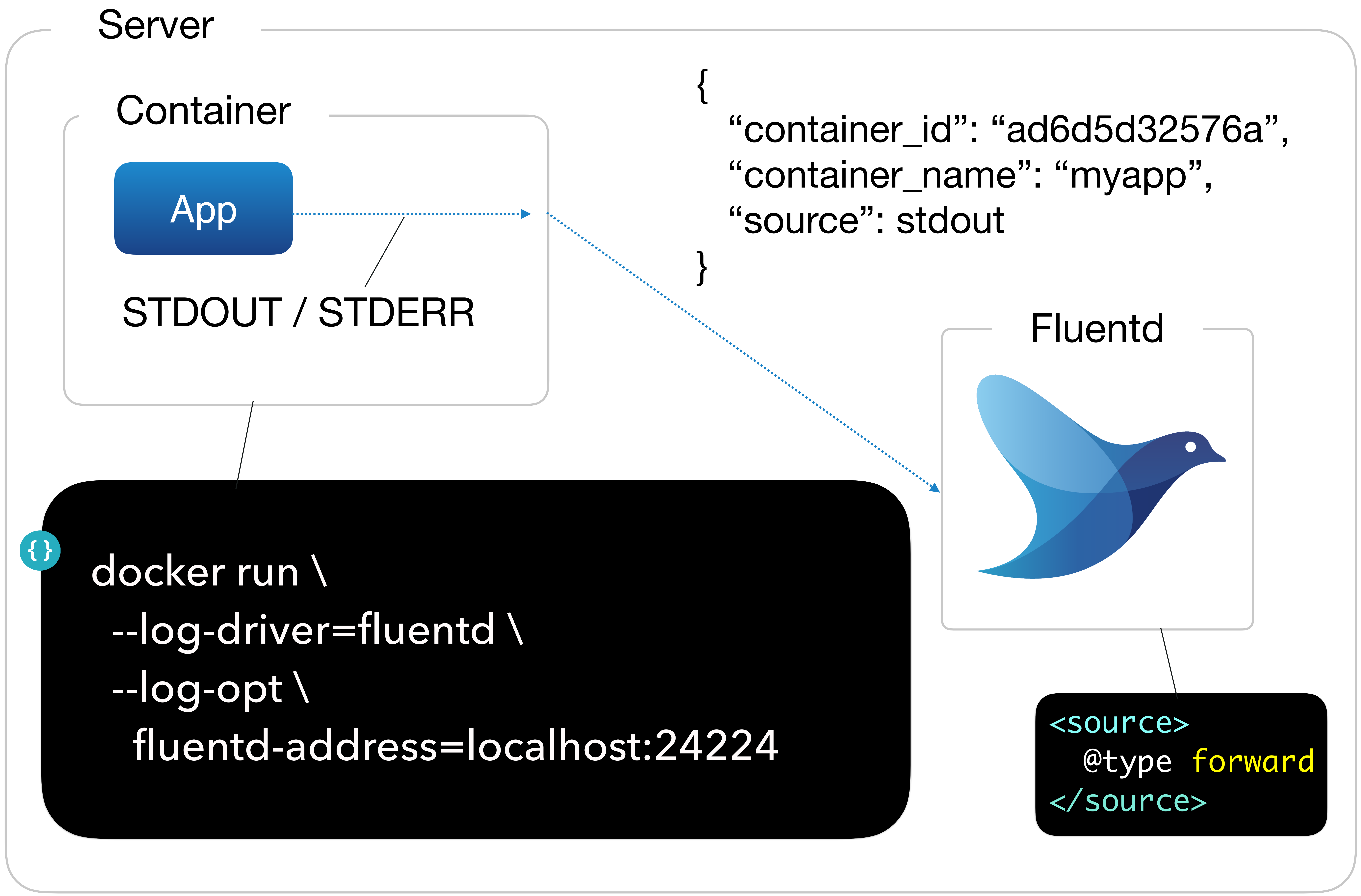
```
# store logs to ES and HDFS
<match web.*>
  @type copy
  <store>
    @type elasticsearch
    logstash_format true
  </store>
  <store>
    @type webhdfs
    host namenode
    port 50070
    path /path/on/hdfs/
  </store>
</match>
```

Container Logging



Collect logs from containers

Text logging with --log-driver=fluentd

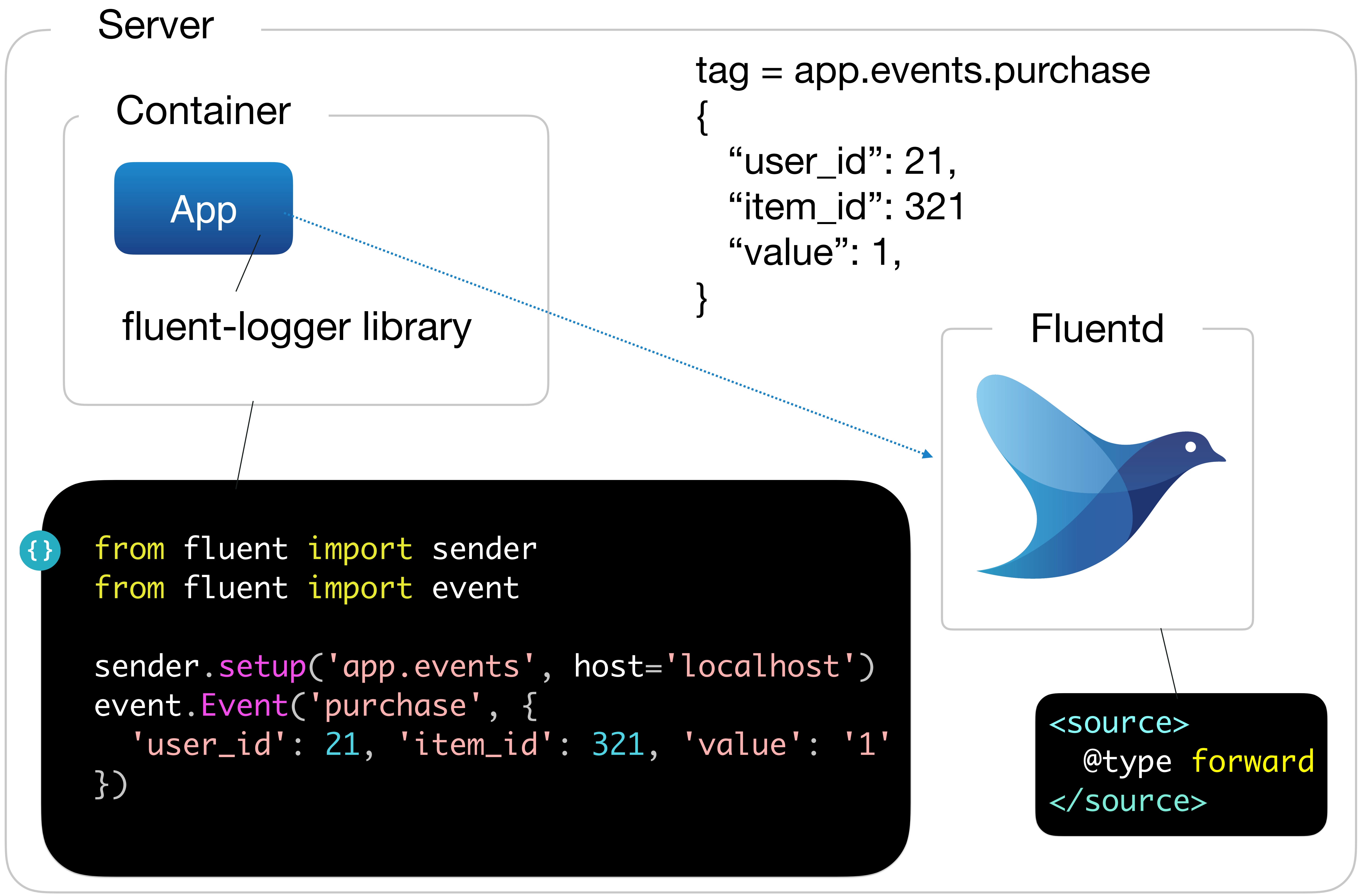


```
{
  "container_id": "ad6d5d32576a",
  "container_name": "myapp",
  "source": stdout
}
```

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

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

Metrics collection with fluent-logger



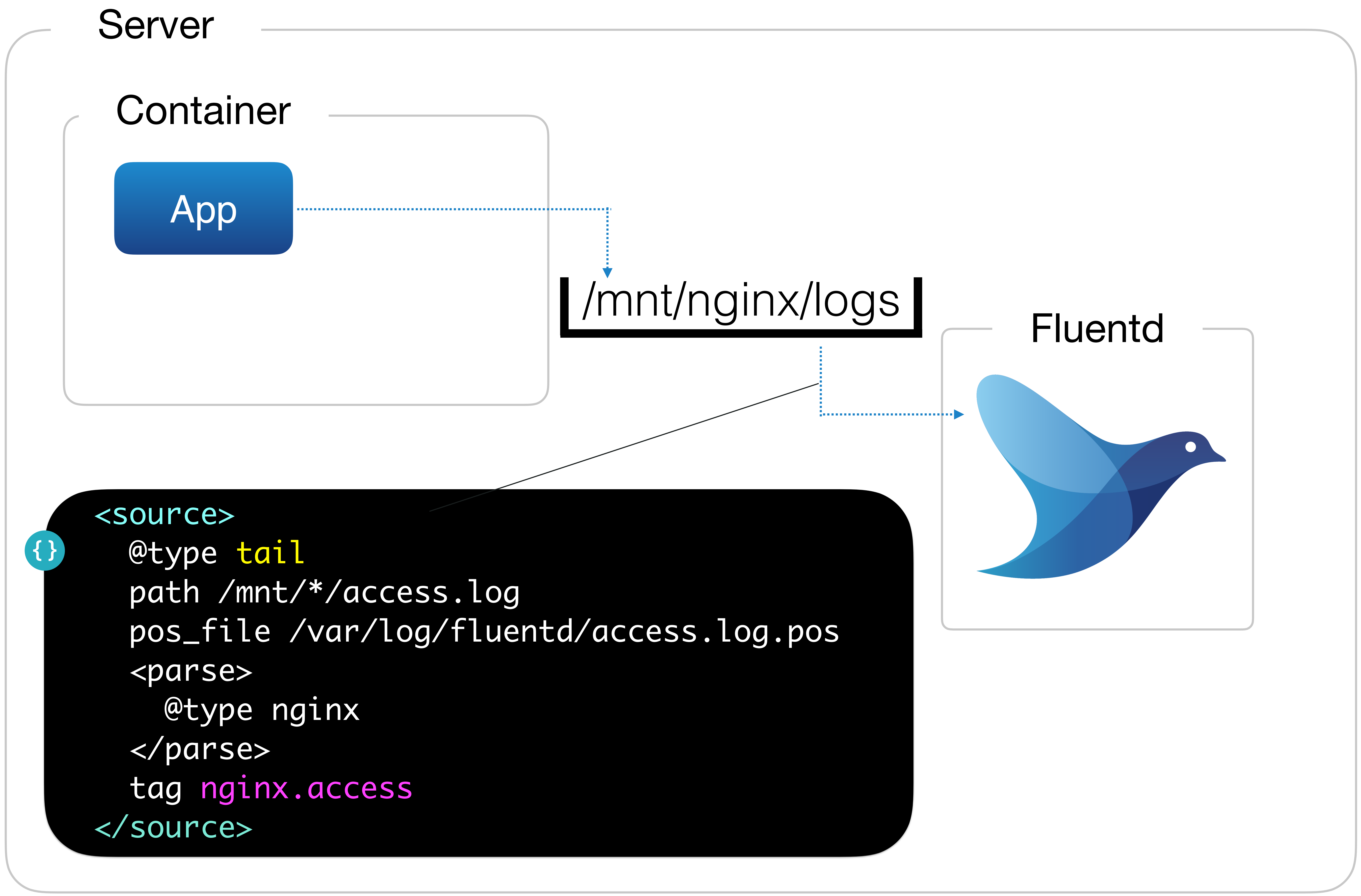
```
tag = app.events.purchase
{
  "user_id": 21,
  "item_id": 321
  "value": 1,
}
```

```python
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>
```

Logging approach summary

- Collecting log messages
  - --log-driver=fluentd
- Application metrics
  - fluent-logger
- Access logs, logs from middleware
  - Shared data volume
- System metrics (CPU usage, Disk capacity, etc.)
  - Fluentd's input plugins (Fluentd pulls metrics periodically)
  - Prometheus or other monitoring agent

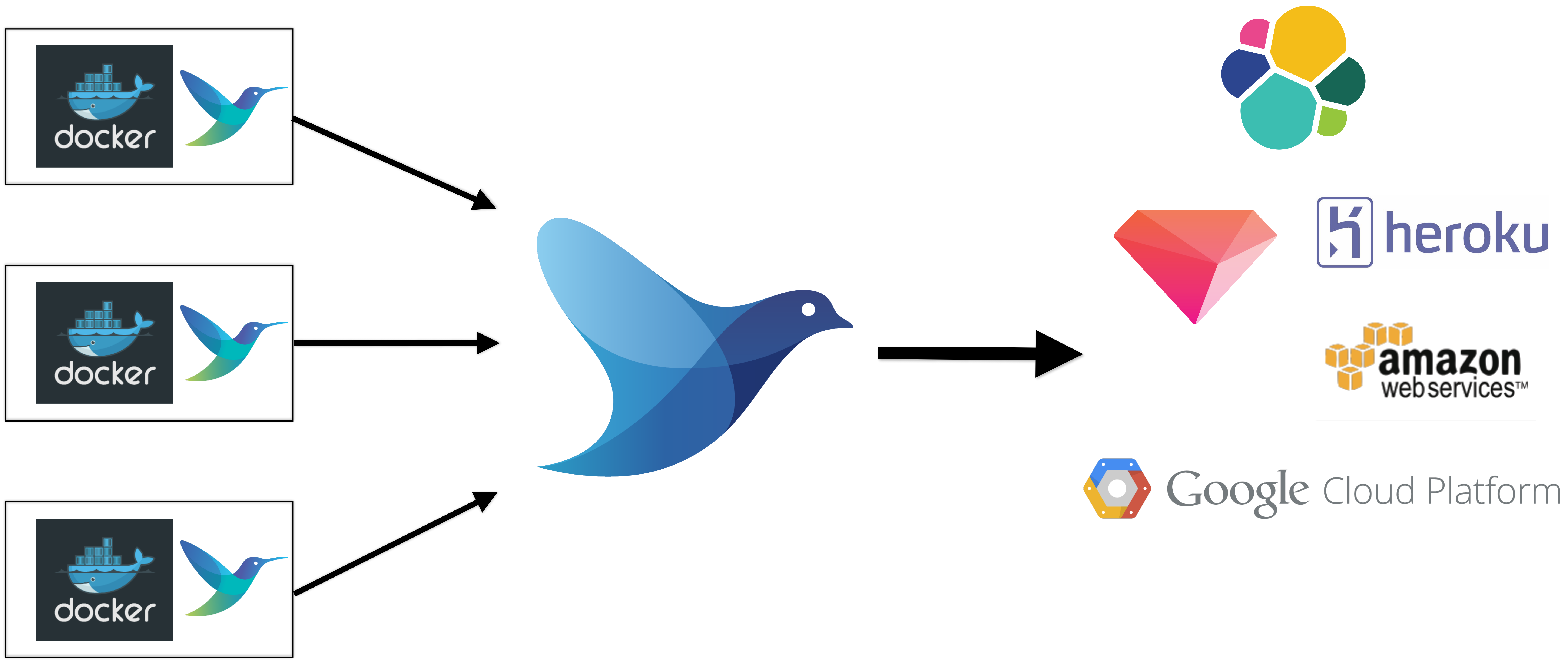
Fluent-bit

Fluent and Fluent-bit

	Fluentd	Fluent-bit
Implementation	Ruby + C	C
Focus	Flexiblility and Robustness	Performance and footprint
Design	Plaggable	Plaggable
Target	Forwarder / Aggregator	Forwarder / Embbeded environment

Forward logs from fluent-bit to fluentd is typical pattern

Container Logging with fluent-bit



Enjoy logging!