CNCon / KubeCon at North America

# Fluentd and Distributed Logging

CLOUD NATIVE
COMPUTING FOUNDATION

Masahiro Nakagawa
Senior Software Engineer

TREASURE
DATA

# Logging and Containers

# Logging on production

- Service Logs
  - Web access logs
  - Ad logs
  - Transcation logs (Game, EC, etc…)

Distributed tracing

- System Logs
  - Syslog, systemd and other logs
  - Audit logs
  - Metrics (CPU, memory, etc…)

**Logs for Bussiness**

KPI
Machine Learning
…

**Logs for Service**

System monitoring
Root cause check
…

# The Container Era

| | Server Era | Container Era |
|---|---|---|
| **Service Architecture** | Monolithic | Microservices |
| **System Image** | Mutable | Immutable |
| **Local Data** | Persistent | Ephemeral |
| **Network** | Physical addresses | No fixed addresses |
| **Log Collection** | syslogd / rsync | **?** |

# Logging challenges with Containers

- No permanent storage

  → Transfer logs to anywhere ASAP

- No fixed physical addresses

  → Push logs from containers

- No fixed mappings between servers and roles

  → Label logs with Service/Tags

- Lots of application types

  → Need to handle various logs

# Fluentd overview

# What's Fluentd?

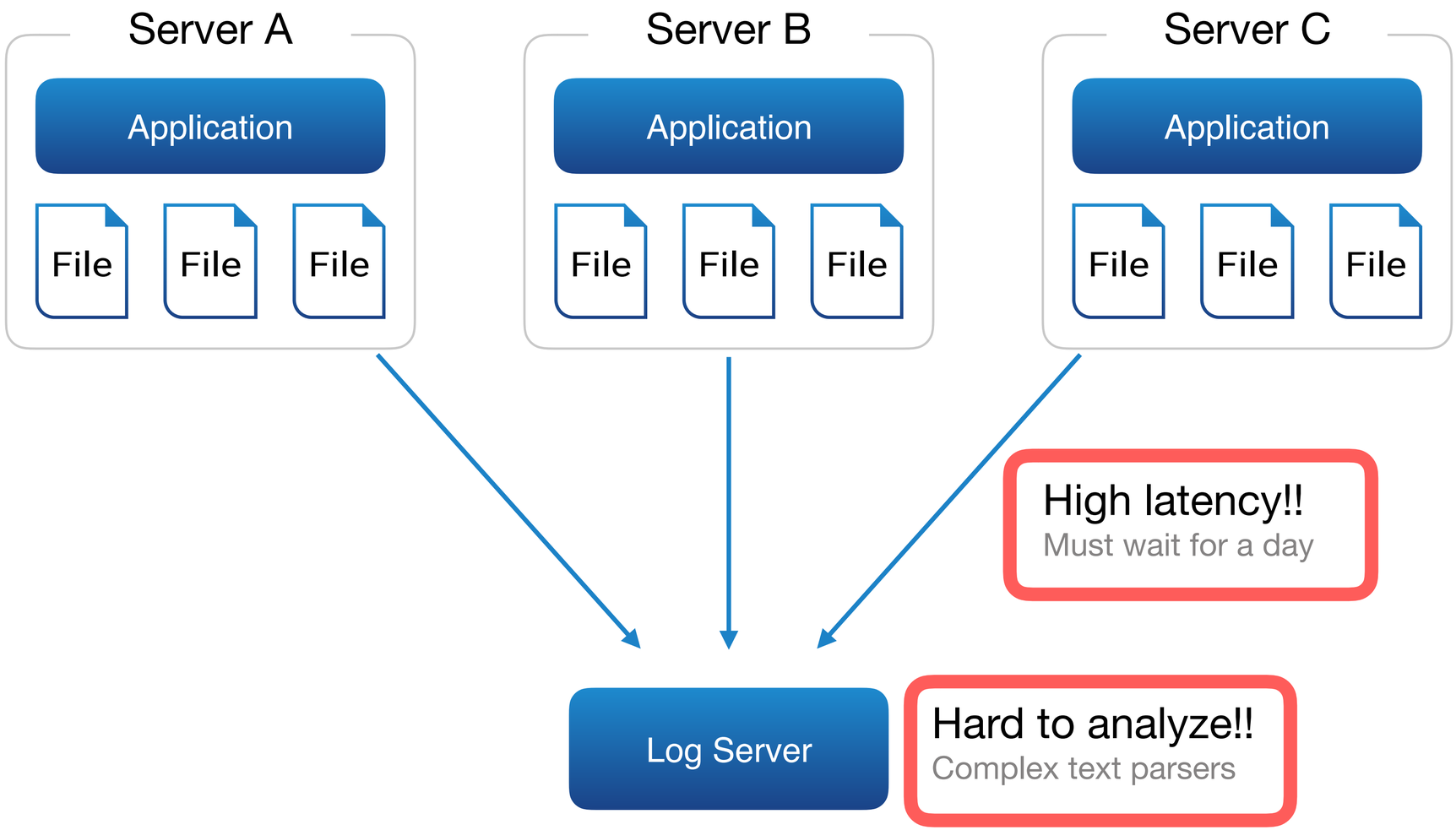
Buffering, HA (failover),
Secondary output, etc.

AN EXTENSIBLE & RELIABLE DATA COLLECTION TOOL

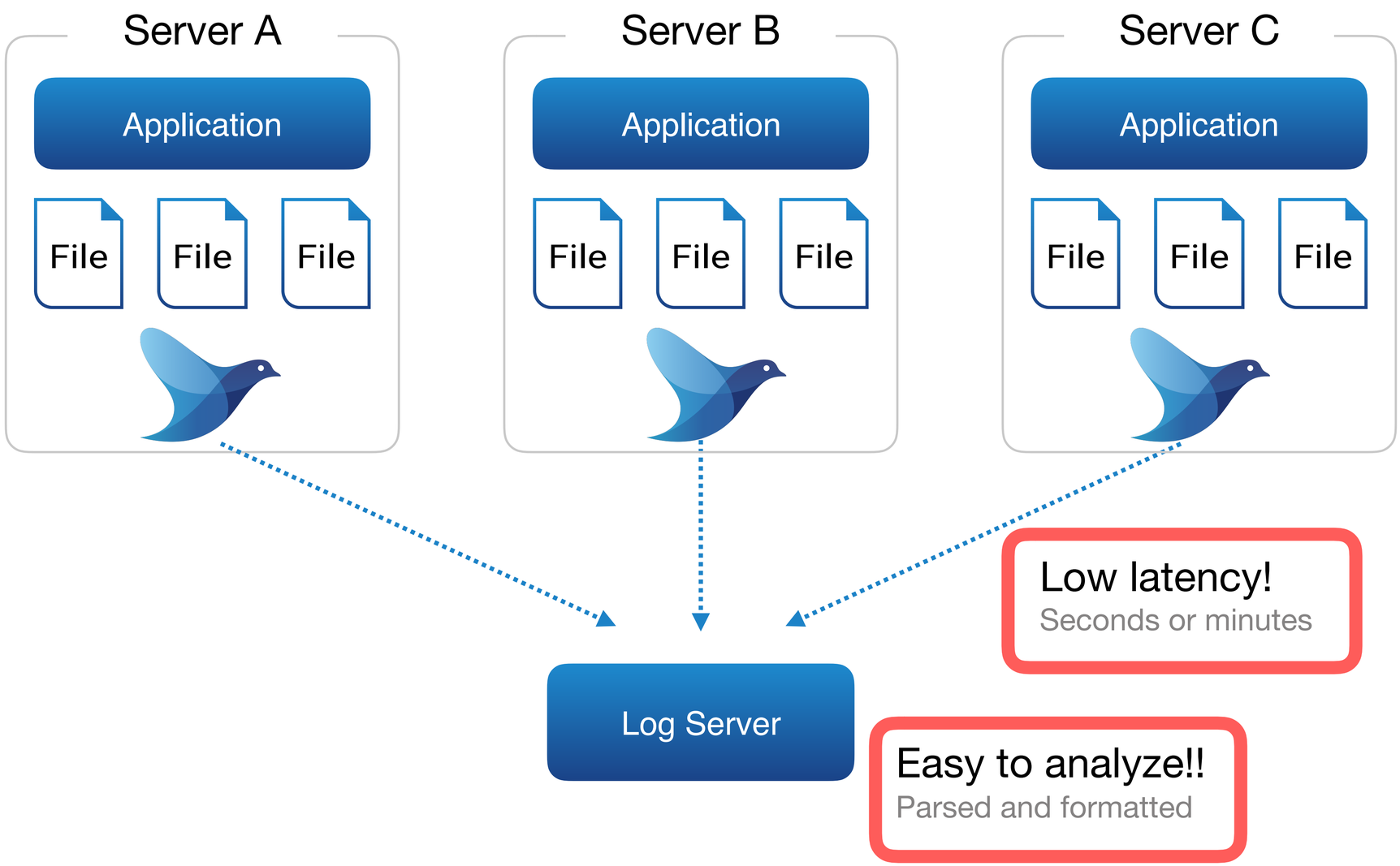
Simple core
+ Variety of plugins

Like syslogd in streaming manner

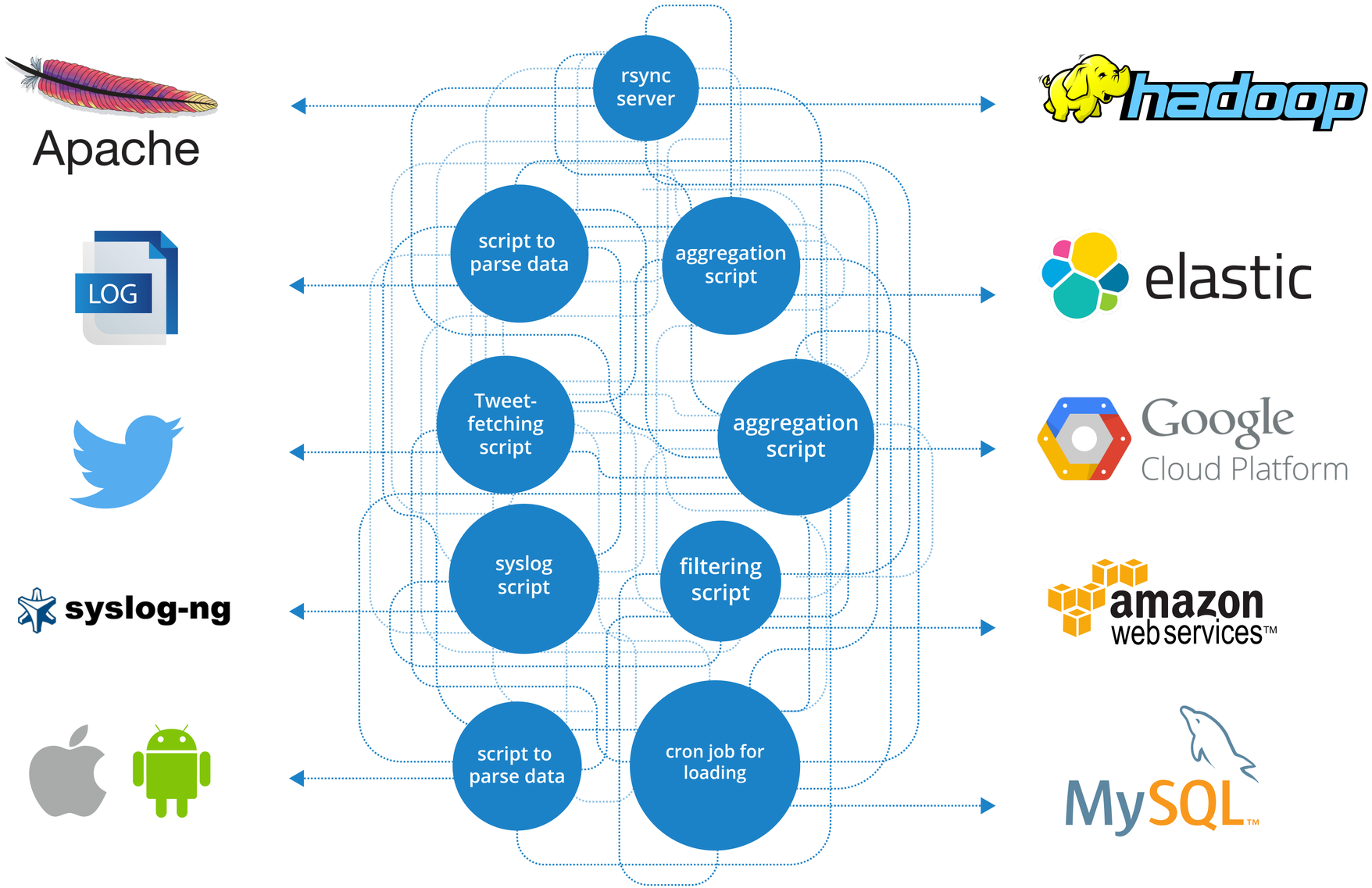
# Log collection with traditional logrotate + rsync



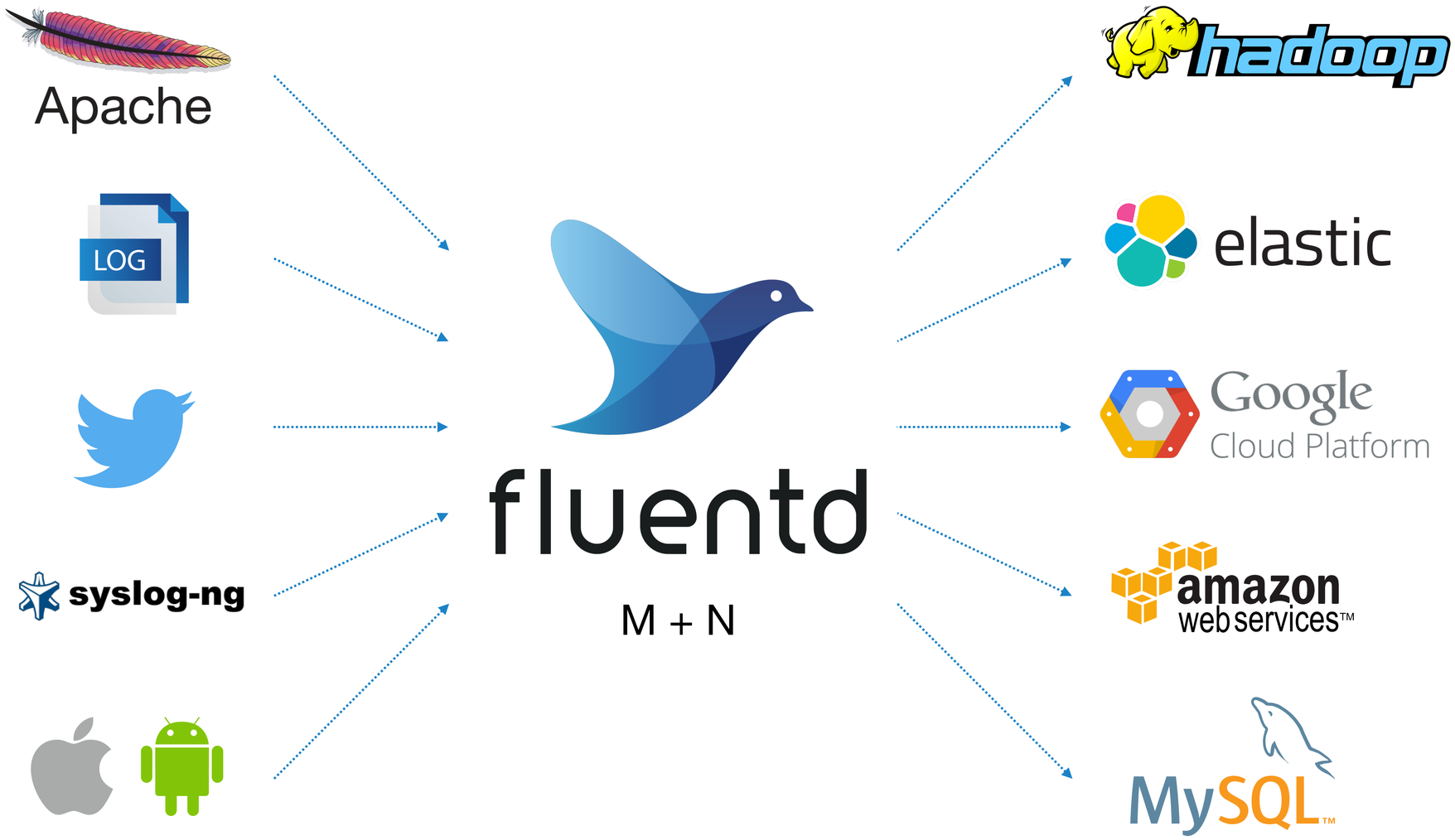
# Streaming way with Fluentd



# M x N problem for data integration

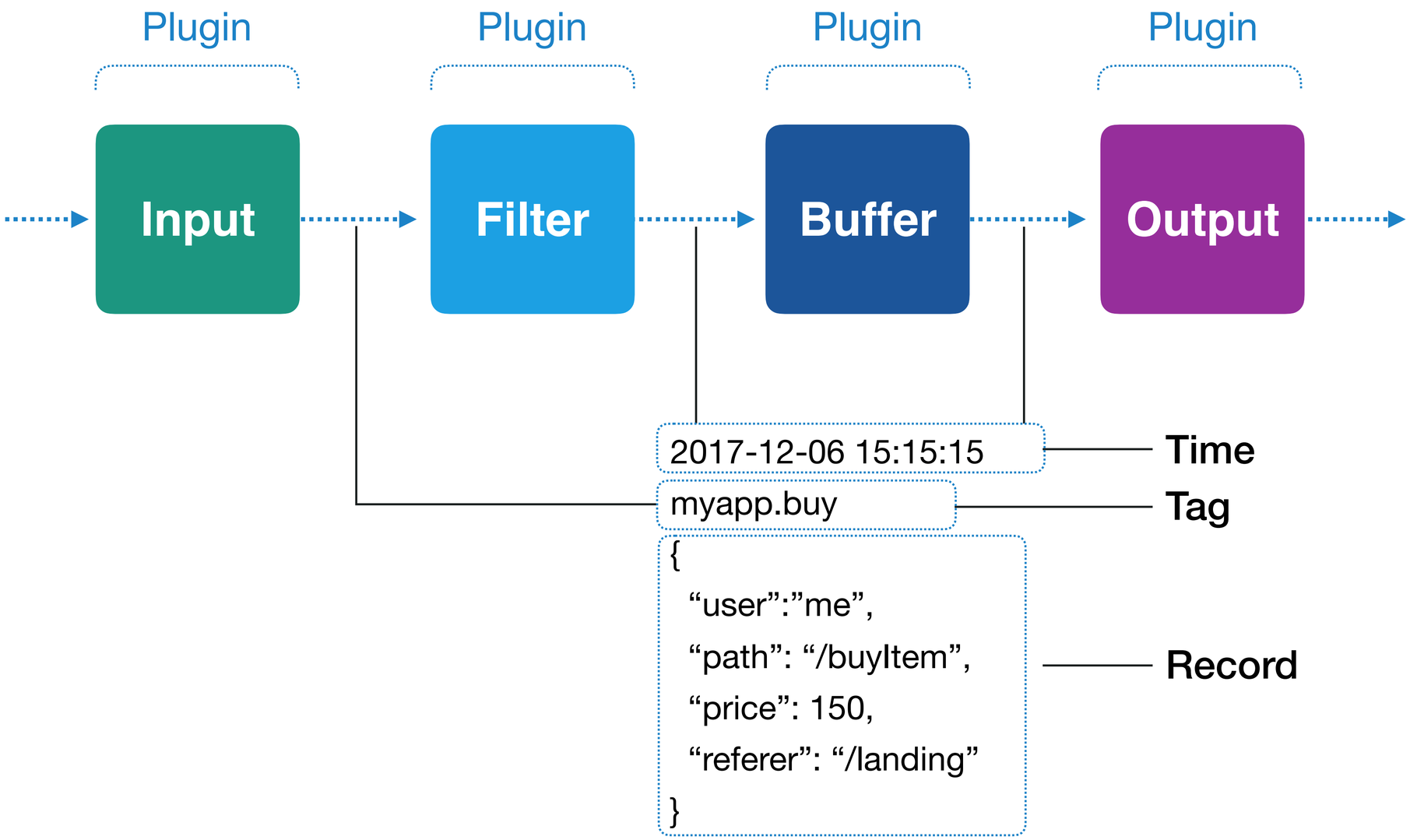


# A solution: unified logging layer

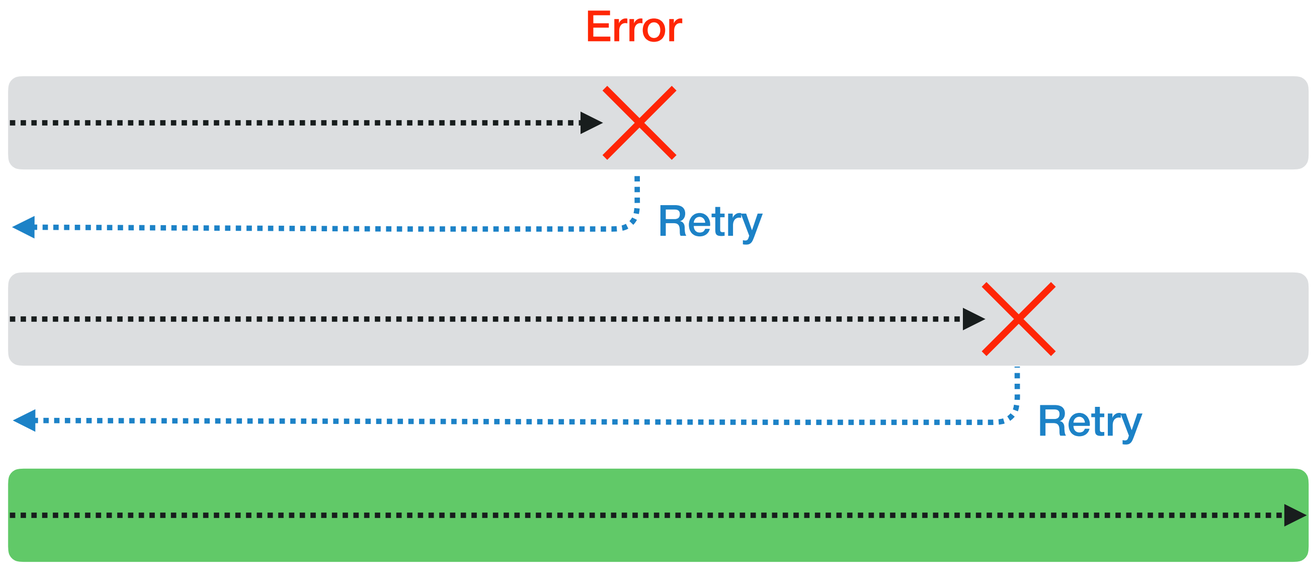


# Fluentd Architecture

# Internal Architecture (simplified)



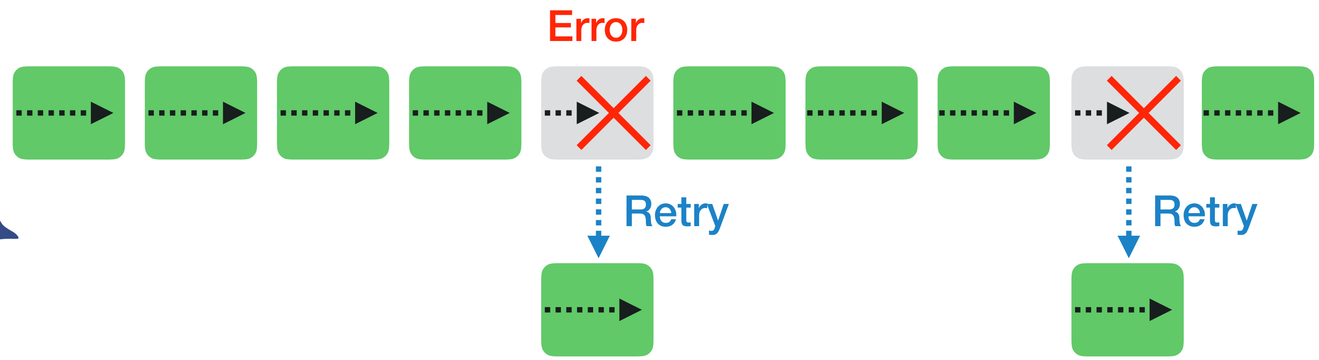
# Divide & Conquer for retry



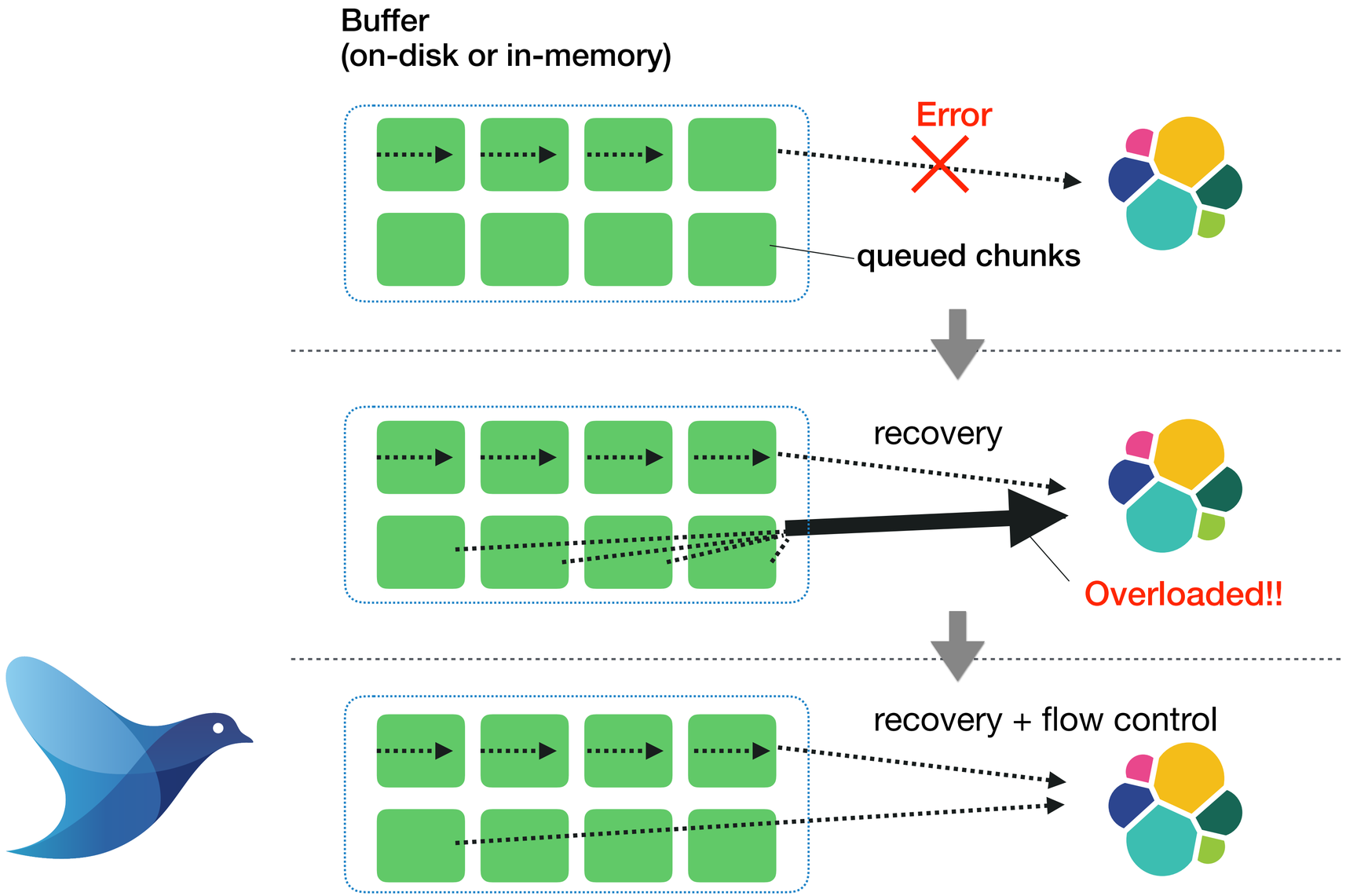

Batch

---

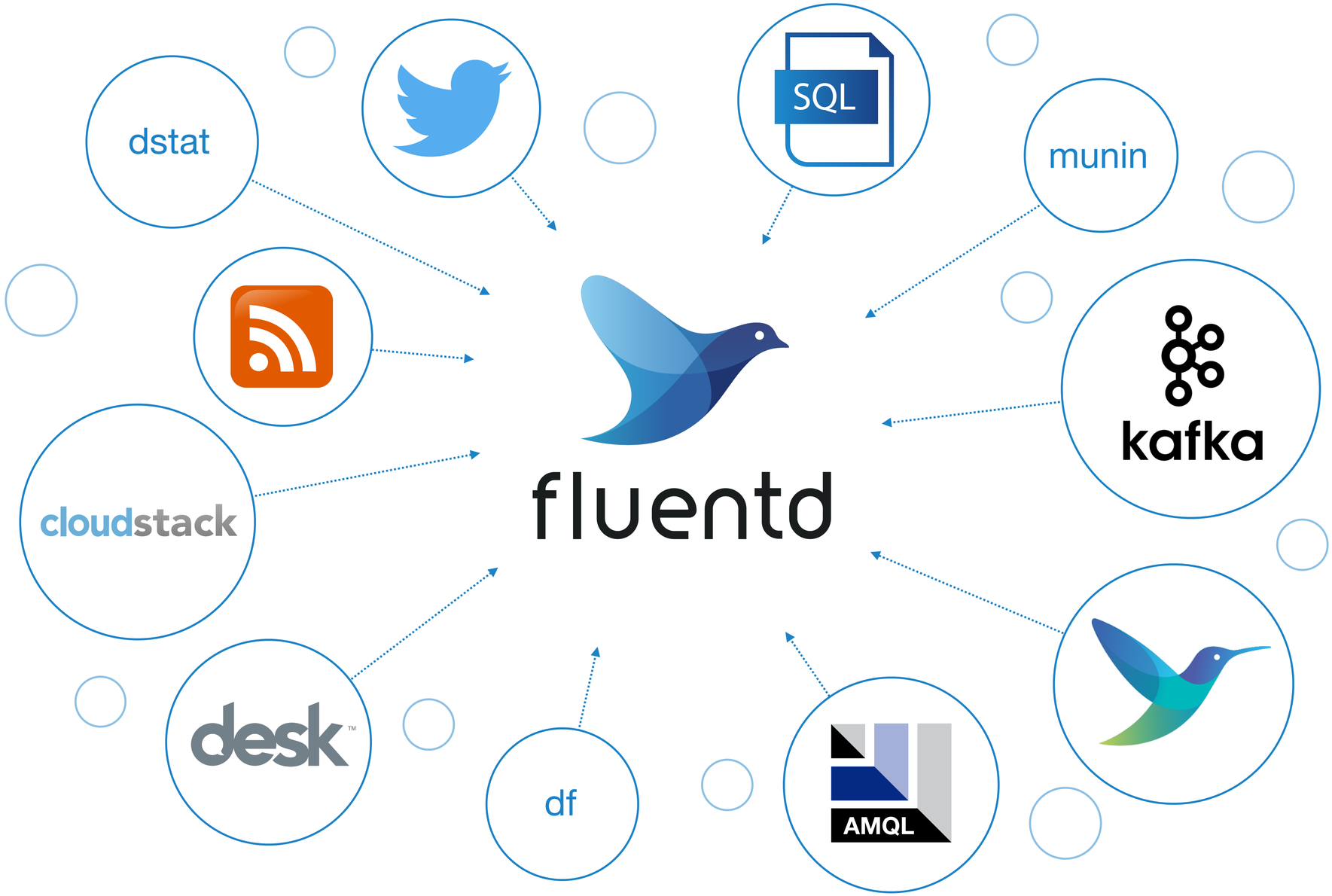
Stream



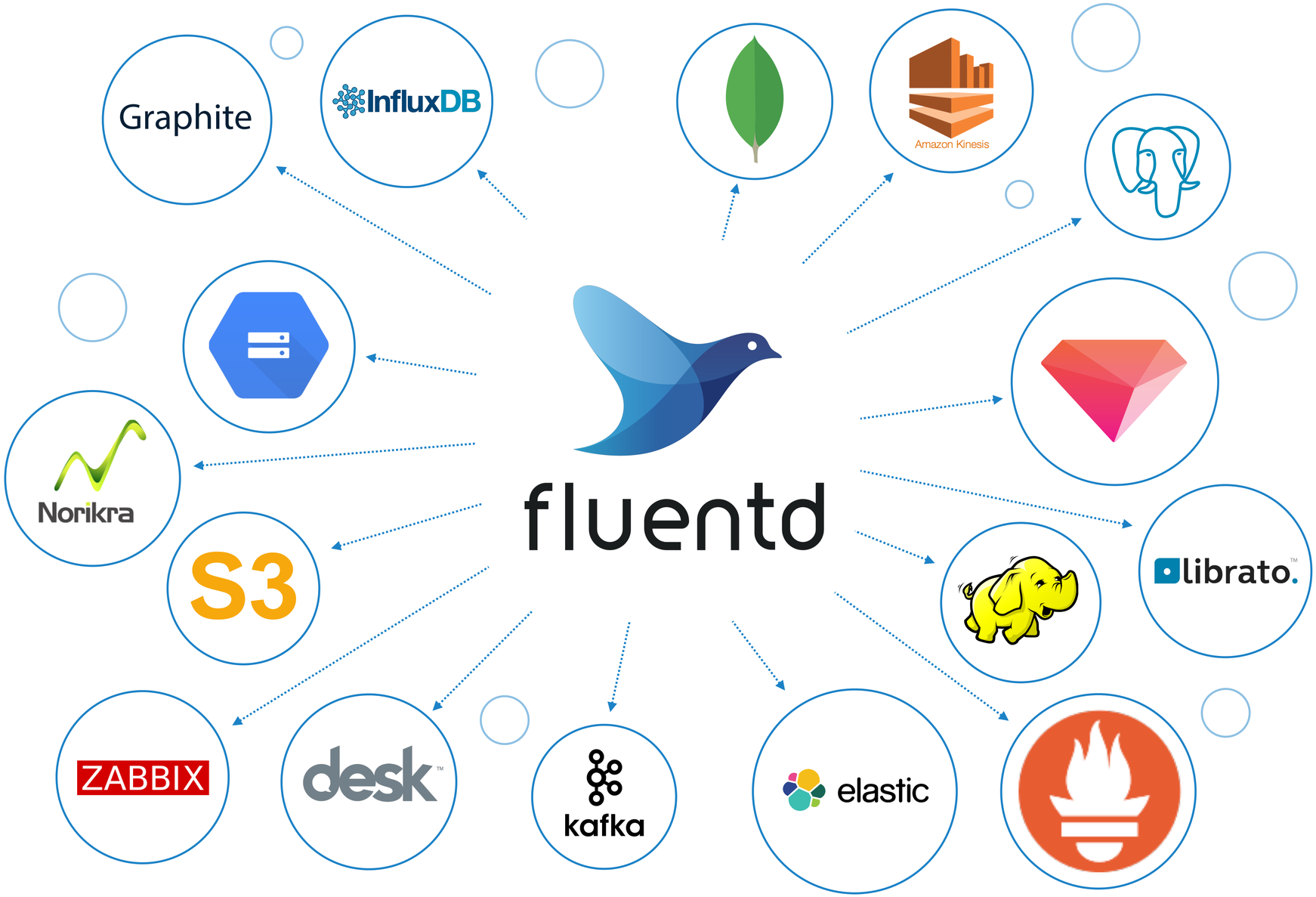

# Divide & Conquer for recovery



# 3rd party input plugins

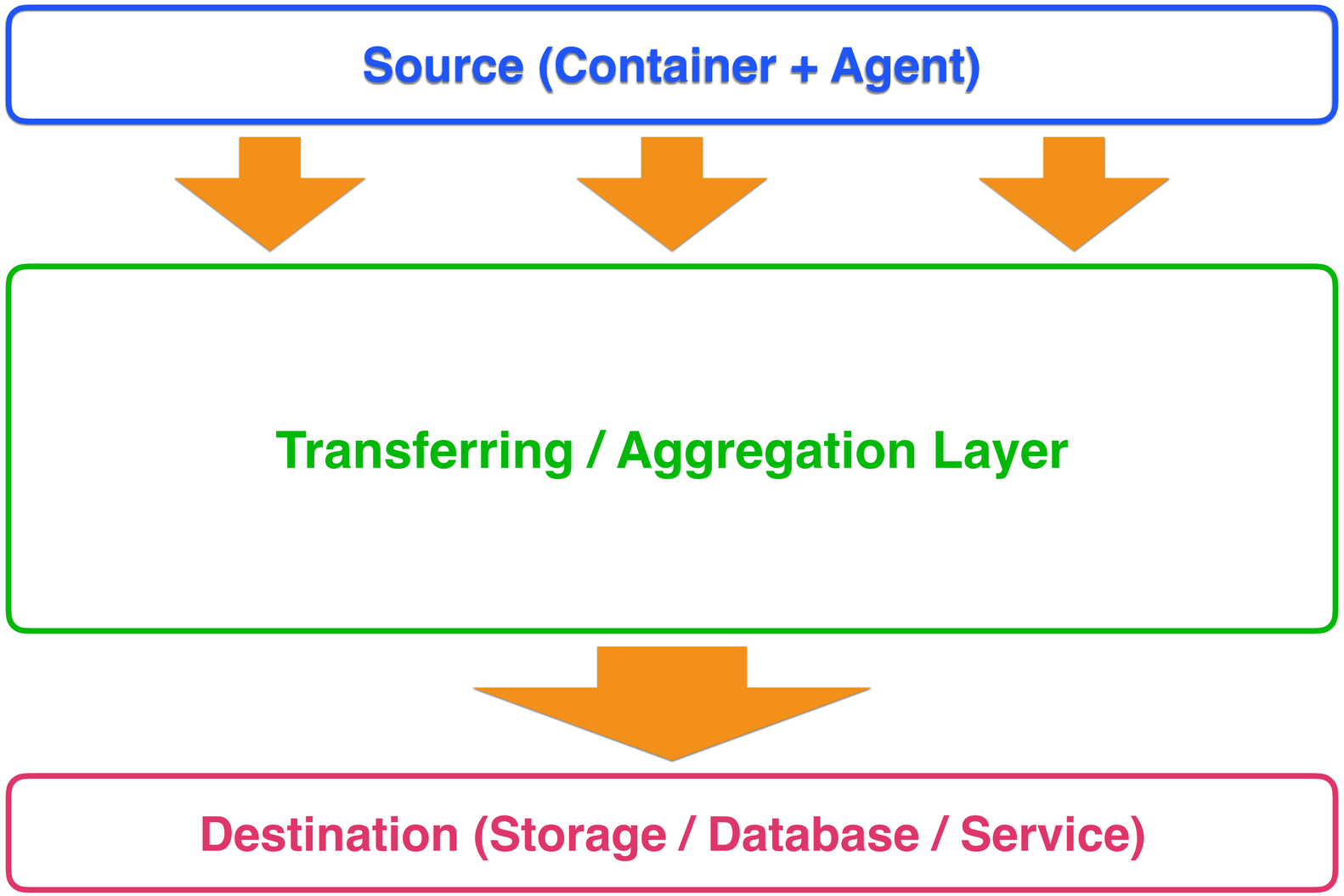


# 3rd party output plugins



# Distributed Logging

# Architecture



# Logging Workflow

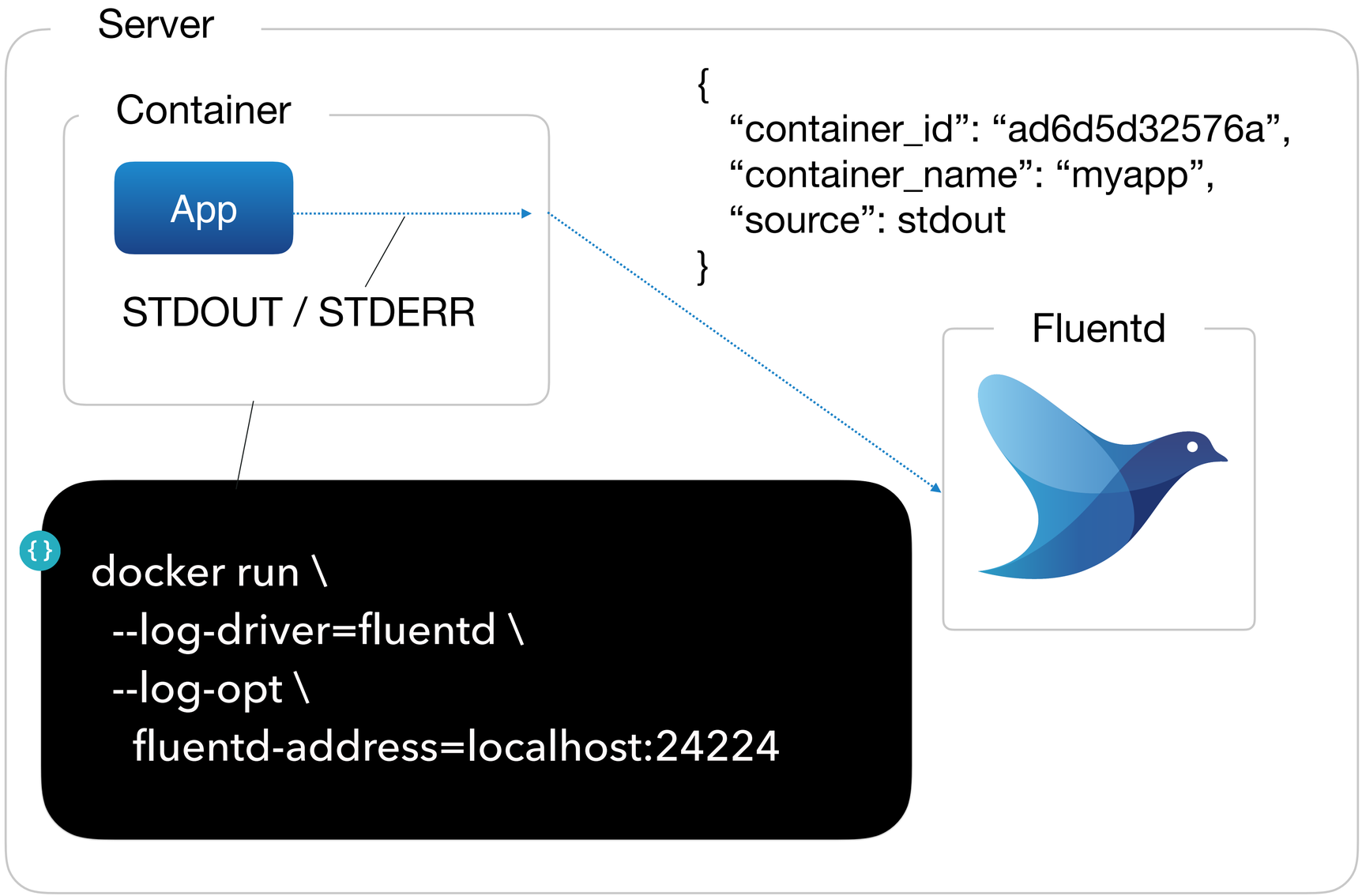

- Retrieve logs: File / Network / API …
- Parse payload for structured logging


- Get logs from multiple sources
- Split/Merge logs into streams

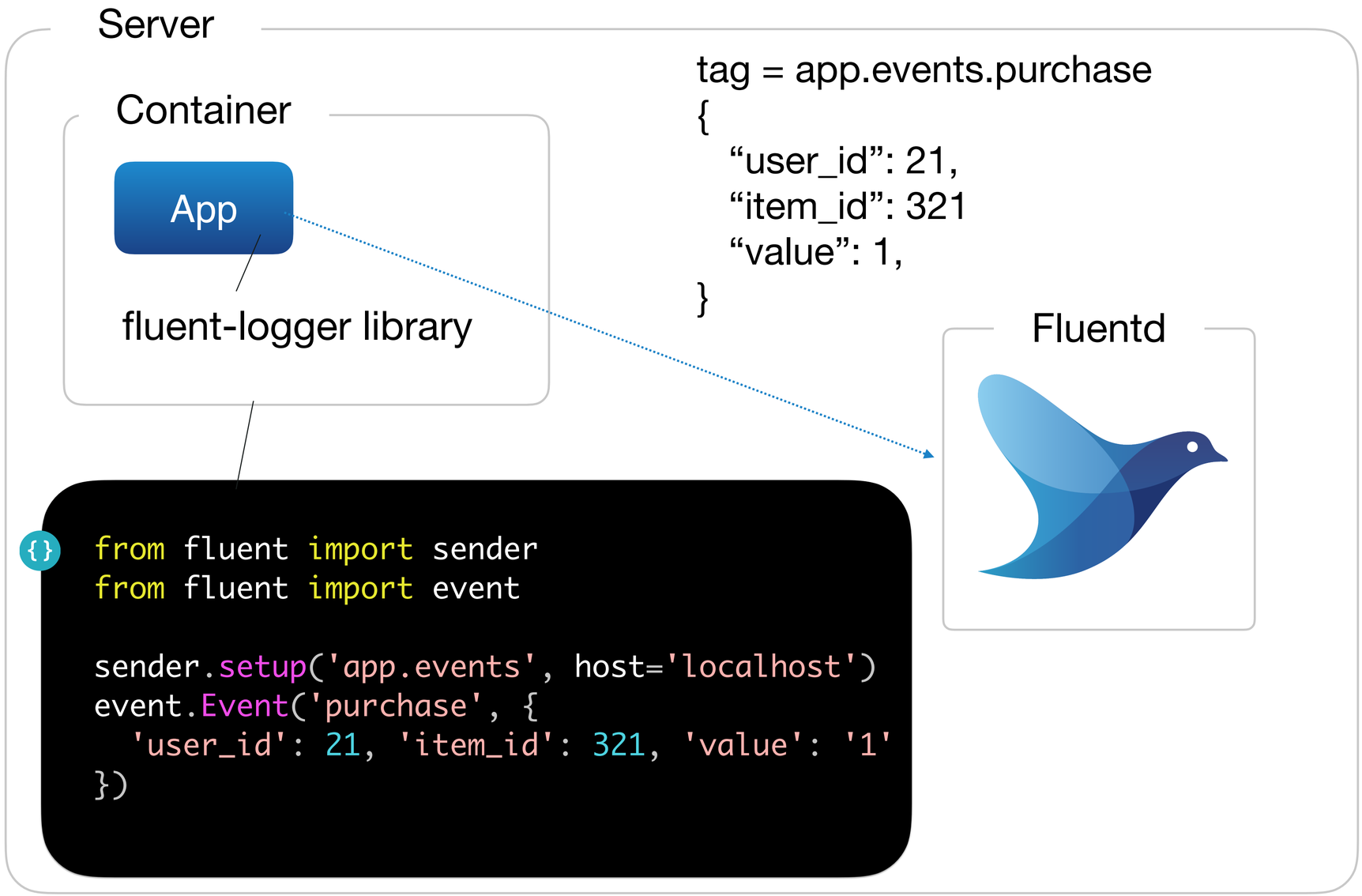

- Receive logs from Aggreagtors
- Store formatted logs

# How to collect logs from containers using Fluentd in source layer?

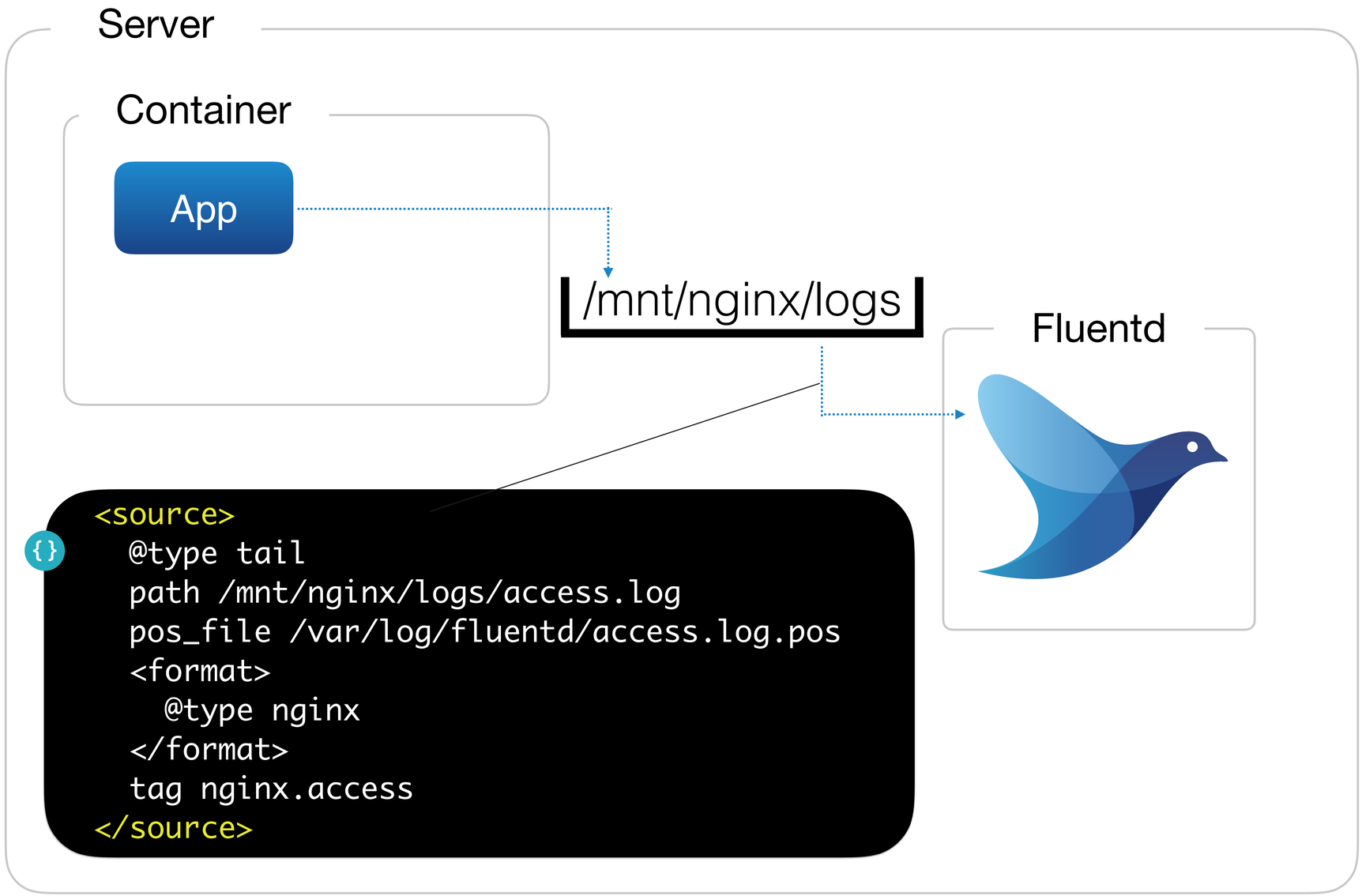
# Text logging with --log-driver=fluentd



# Metrics collection with fluent-logger



# Shared data volume and tailing

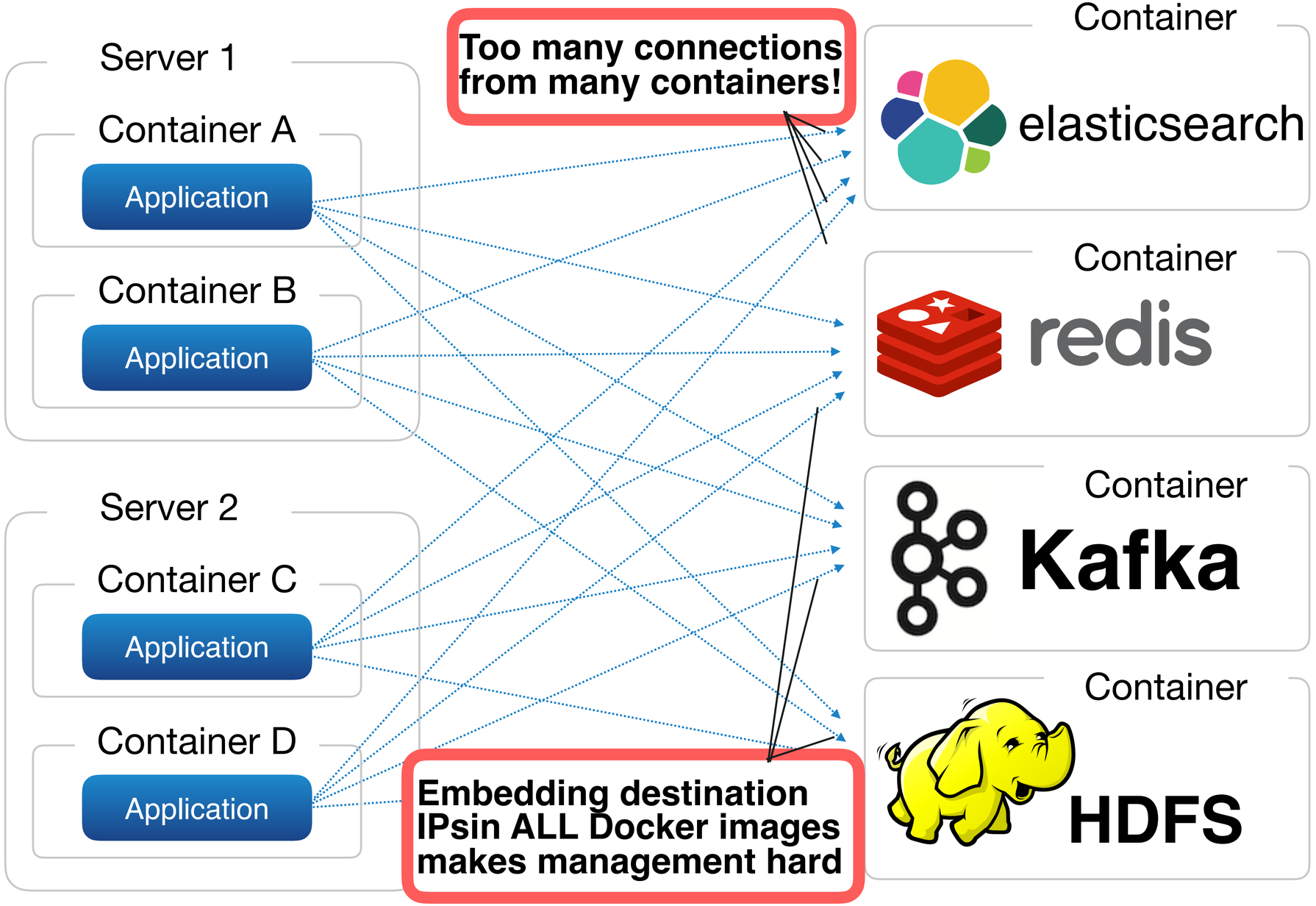


# Logging methods for each purpose

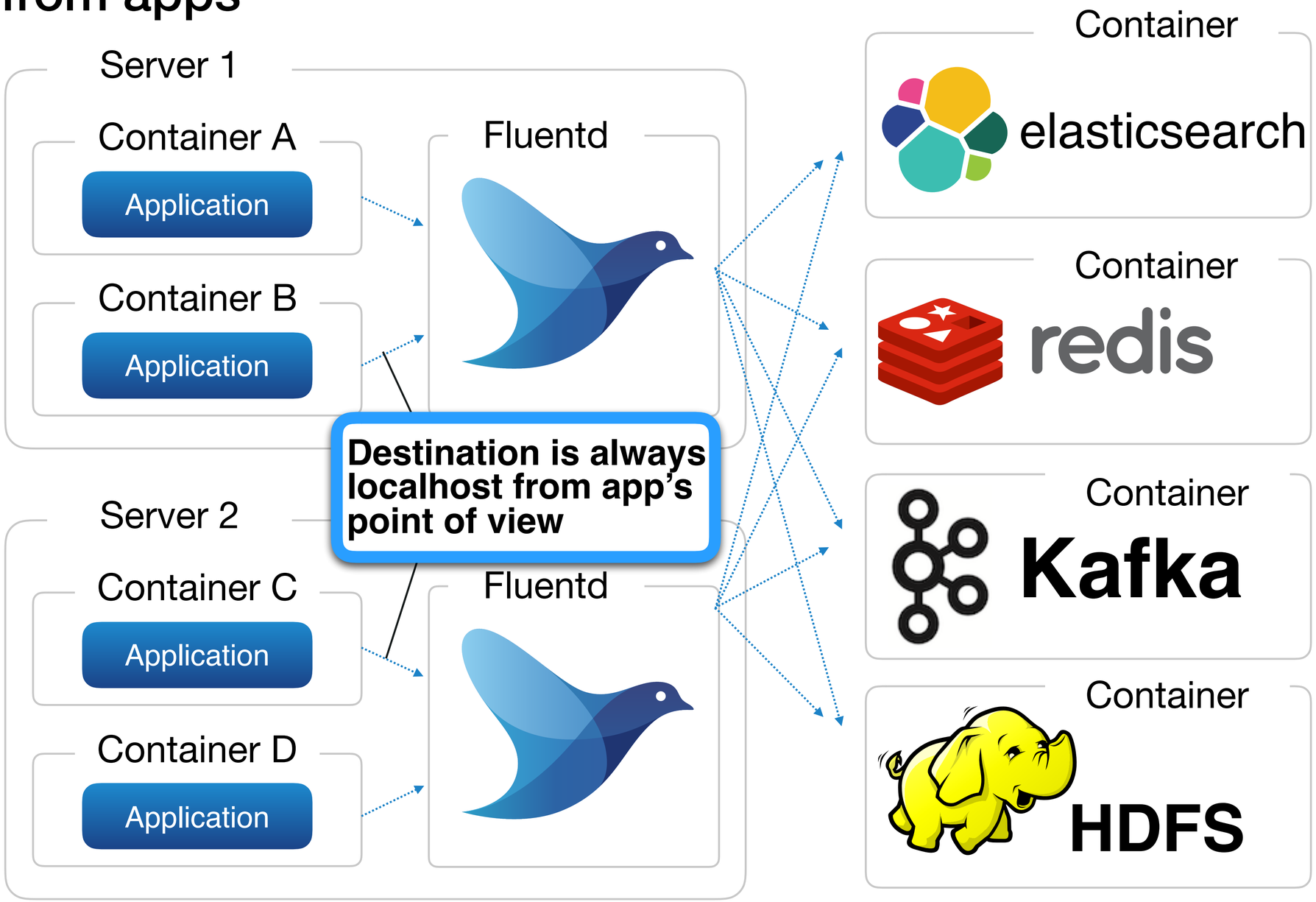
- Collecting log messages
  - --log-driver=fluentd
- Application metrics
  - fluent-logger
- Access logs, logs from middleware
  - Shared data volume
- System metrics (CPU usage, Disk capacity, etc.)
  - Fluentd’s input plugins (Fluentd pulls data periodically)
  - Prometheus or other monitoring agent

# Deployment Patterns

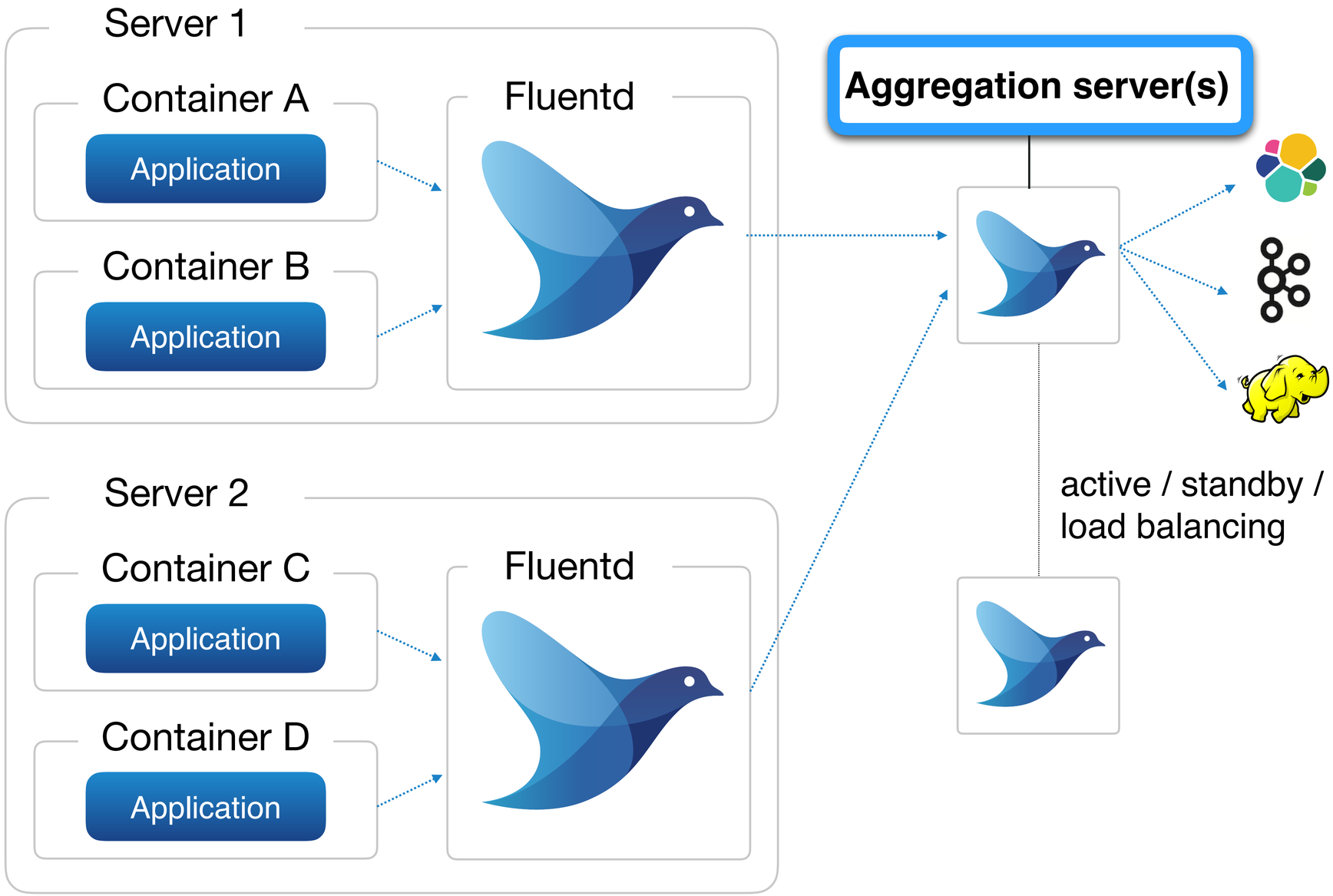
# Primitive deployment…



# Source aggregation decouples config from apps



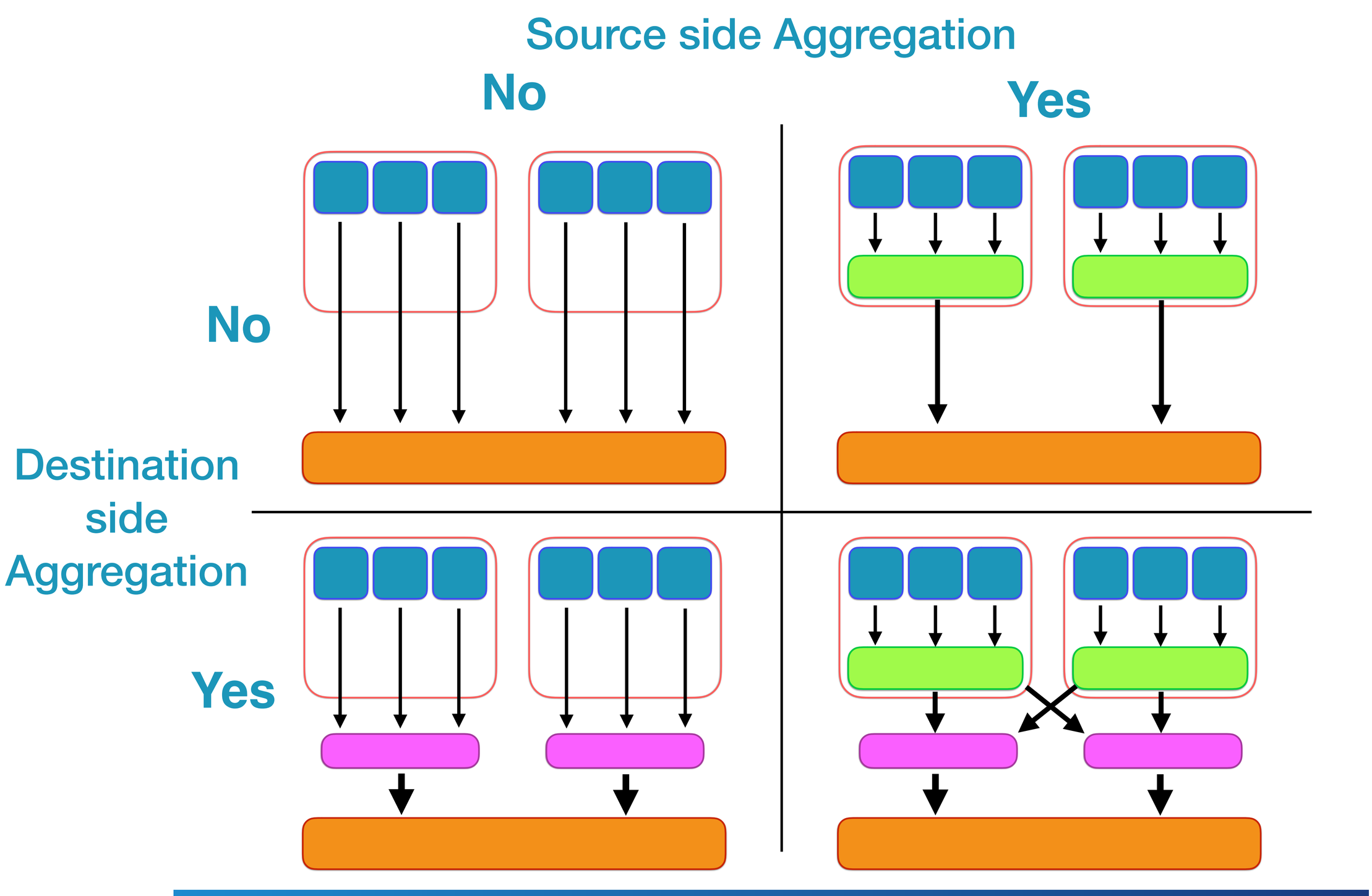
# Destination aggregation makes storages scalable for high traffic



# Aggregation servers

- Logging directly from microservices makes log storages overloaded.
  - Too many connections
  - Too frequent import API calls

- Aggregation servers make the logging infrastracture more reliable and scalable.
  - Connection aggregation
  - Buffering for less frequent import API calls
  - Data persistency during downtime
  - Automatic retry at recovery from downtime


Source side Aggregation
No
Yes
No
Destination side Aggregation
Yes

# Should use these patterns?

- Source-side aggregation: **Yes**
  - Fluentd frees logging pain from applications
    - Buffering, Retry, HA, etc…
  - Application don’t need to care destination changes

- Destination-side aggregation: **It depends**
  - good for high traffic
  - maybe, no need for cloud logging services
  - may need for self-hosted distributed systems or cloud services which charges per request

# Scalable Distributed Logging

- Network
  - Split heavy traffic into traffics to nodes
  - Merge connections
- CPU / Memory
  - Distribute processing to nodes about heavy processing
- High Availability
  - Switch / fallback from a node to another for failure
- Agility
  - Avoid reconfiguring whole logging layer to modify destinations

# Fluentd ♡ Container

- Fluentd model fits container based systems
  - Pluggable and Robust pipelines
  - Support typical deployment patterns

- Smart CNCF products for scalable system
  - k8s: Container orchestration
  - Prometheus: Monitoring
  - Fluentd: Logging
  - JAEGER: Distributed Tracing
  - etc…

**Let’s make scalable and stable system!**

# Enjoy logging!