AWS re:Invent

CON309-R

Fluent Bit and AWS FireLens

Wesley Pettit
Software Engineer
Amazon Web Services

Eduardo Silva
Principal Engineer
Arm

AWS re:Invent

aws

Agenda

- High-level overview of Fluent Bit and AWS FireLens
- Discuss
  - What’s missing? What should we add?
  - What are your use cases? How can we solve them?

Logging

Logging Challenges

Multiple sources of information

- TCP/UDP
- File system, common log files
- Systemd/Journald
- Sensors!

Logging Challenges

…and each one with different formats

- Apache Logs

  [14/Mar/2019:23:43:52 +0000] GET /Frasera HTTP/1.0 500 2216
- MySQL

  2019-04-30T21:32:39.095880Z 0 [Note] InnoDB: Mutexes use GCC atomic builtins
- JSON Maps

  {“log”: “Hey re:Invent!”, “stream”: “stdout”, “time”: “2019-05-07T10:03:11.33507113Z”}
- Many others…!

Structured Messages

Unstructured to structured

Hey re:Invent!


```
{
  "log": "Hey re:Invent!",
  "source": "stdout",
  "time": "2019-12-2T14:03:11.335071137Z"
}
```

Structured Messages

Metadata


```
{
  "log": "Hey re:Invent!",
  "source": "stdout",
  "time": "2019-12-2T14:03:11.335071137Z",
  "container_id":
"e54cccfac2b87417f71877907f6787906842004282806 7ae0867e60a63529d35",
  "container_name": "/ecs-demo-6-container2-a4eafbb3d4c7f1e16e00",
  "ecs_cluster": "mycluster",
  "ecs_task_arn": "arn:aws:ecs:us-east-2:012345678910 11:task/mycluster/3de392df-6bfa-
470b-97ed-aa6f482cd7a6",
  "ecs_task_definition": "demo:7",
  "ec2_instance_id": "i-06bc83dbc2ac2fdf8"
}
```

Fluent Bit

AWS re:Invent

aws

About

Fluent Bit

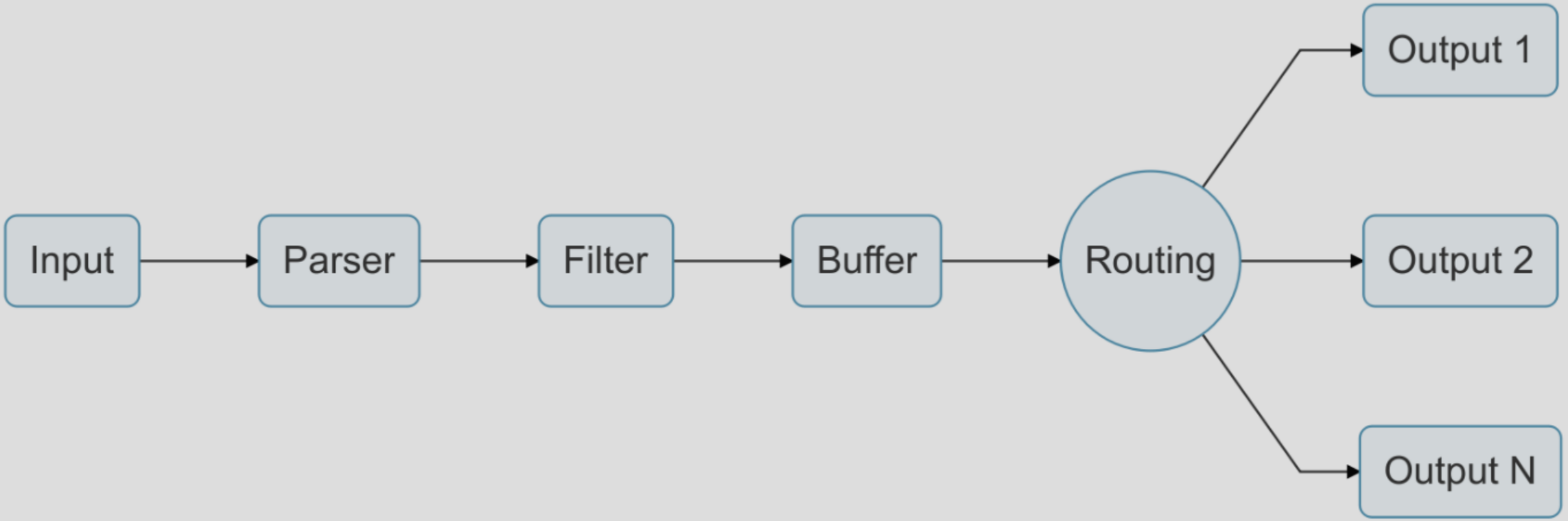
- Started in 2015
- Origins: Lightweight log processor for **embedded Linux**
- Quickly evolved as a solution for the **cloud** space
- Apache License v2

Fluent Bit

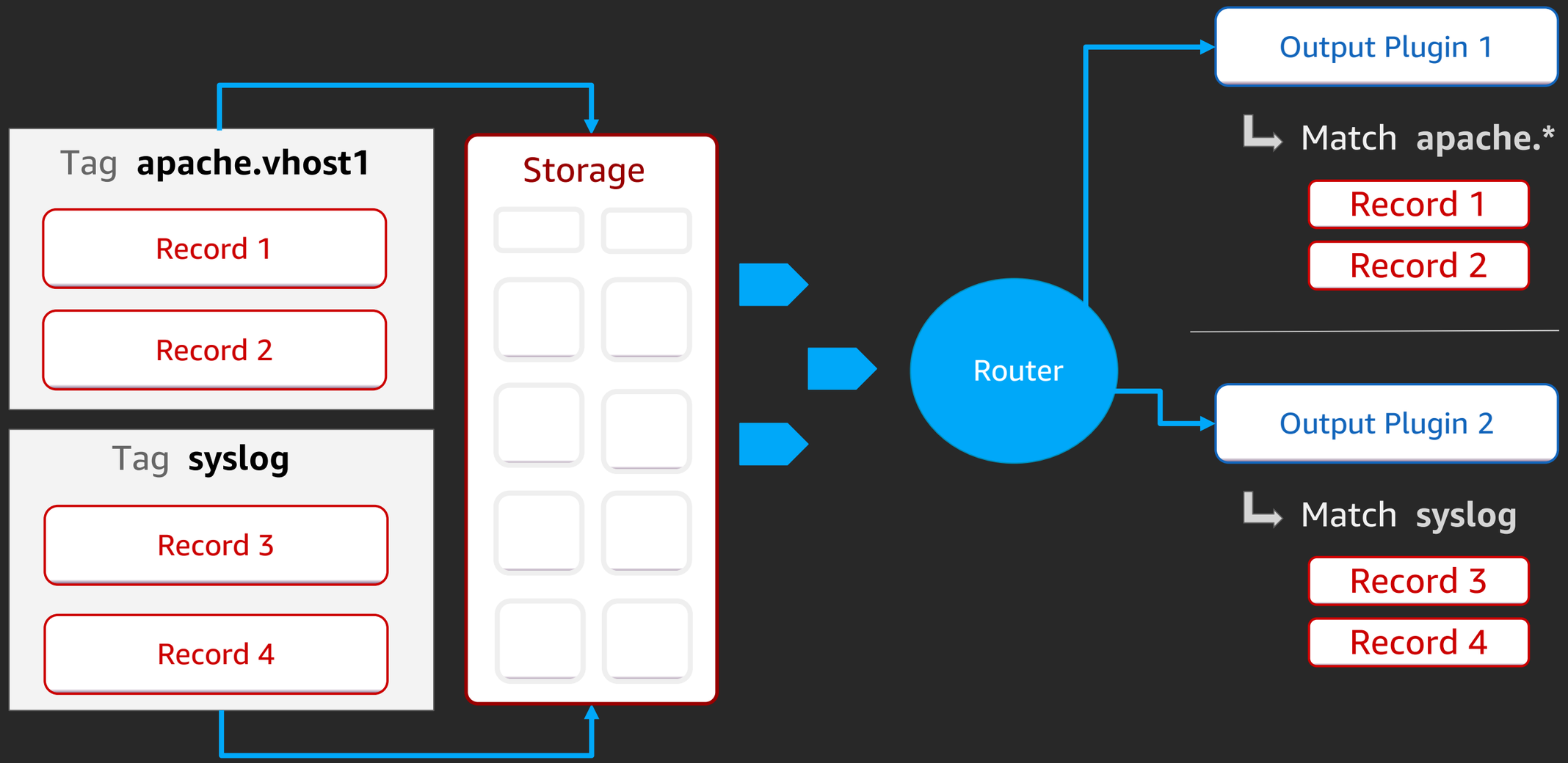
Design and Internals

- Written in **C**
- **Low** memory and CPU footprint
- Pluggable architecture (~50 plugins)
- Built-in security: TLS on network I/O

Fluent Bit


Input
Parser
Filter
Buffer
Routing
Output 1
Output 2
Output N

Logging & Routing



Fluent Bit Adoption

- >250,000 deployments **every single day!**
- Wide adoption
  - AWS
  - DataDog
  - Kubernetes community

AWS for Fluent Bit

Amazon/aws-for-fluent-bit

Output logs to:

- Amazon CloudWatch Logs
- Amazon Kinesis Data Firehose
- Amazon Kinesis Data Streams
- All destinations supported natively in Fluent Bit

Why did we choose Fluent Bit?

Amazon Kinesis Data Firehose plugins: Fluentd vs. Fluent Bit

Log Lines per Second	Data Out	Fluent Bit CPU (% vCPU/CPU Thread)	Fluent Bit Memory	Fluentd CPU (% vCPU/CPU Thread)	Fluentd Memory
100	25 KB/s	0.3%	27 MB	0.6%	84 MB
1000	250 KB/s	3.3%	37 MB	7.3%	102 MB
10000	2.5 MB/s	13%	55 MB	86%	438 MB

FireLens

When we designed FireLens, we envisioned two major segments of users:

1. Those who want a simple way to send logs anywhere, *powered by Fluentd & Fluent Bit*.

2. Those who want *the full power of Fluentd & Fluent Bit*, with AWS managing the undifferentiated labor that's needed to pipe a task's logs to these log routers.

- Wesley

Under the Hood: FireLens for Amazon ECS Tasks

A simple way to send logs anywhere

Configure Fluentd/Fluent Bit as if it was a log driver:

```
"logConfiguration": {
        "logDriver":"awsfirelens",
        "options": {
                "Name": "firehose",
                "region": "us-west-2",
                "delivery_stream":
                "my-stream"
        }
},
```

A simple way to send logs anywhere

Add a side-car to your Task Definition:

```
{
   "essential": true,
   "image": "amazon/aws-for-fluent-bit:latest",
   "name": "log_router",
   "firelensConfiguration": {
       "type": "fluentbit"
   }
}
```

That’s it!

AWS re:Invent

aws

The full power of Fluentd and Fluent Bit

Pull a configuration file from S3:

```
{
   "essential": true,
   "image": "amazon/aws-for-fluent-bit:latest",
   "name": "log_router",
   "firelensConfiguration": {
       "type": "fluentbit”,
        "options": {
             "config-file-type": "s3",
             "config-file-value":
       "arn:aws:s3:::yourbucket/yourdirectory/extra.conf"
       }
   }
}
```

The full power of Fluentd and Fluent Bit

Or bake it into your image:

```
{
   "essential": true,
   "image": ”012345678910.dkr.ecr.us-west-2.amazonaws.com/better-json-
   logger:latest ",
   "name": "log_router",
   "firelensConfiguration": {
       "type": "fluentbit”,
        "options": {
             "config-file-type": ”file",
             "config-file-value": ”/custom.conf"
       }
   }
}
```

That’s it!

AWS re:Invent

aws

Discussion: Questions and Use Cases

What should we work on?

Questions and feature requests

- Fluent Bit - Github.com/fluent/fluent-bit
- AWS & Fluent Bit - Github.com/aws/aws-for-fluent-bit
- FireLens & more - Github.com/aws/containers-roadmap
- Twitter - @edsiper and @TheWesleyPettit

Additional slides for common Q/A topics

Common Logging Use Cases

Use Case – Filter Logs

```
[FILTER]
    Name grep
    Match *
    Regex log [Ee]rror
```

Logging Use Case – Parse JSON

```
[SERVICE]
        Log_Level info
        Parsers_File /fluent-bit/parsers/parsers.conf
[FILTER]
        Name parser
        Match *
        Key_Name log
        Parser json
        Reserve_Data True
```

Logging Use Case – Parse JSON

This:

```
{

        "log": "{\"requestID\": \"b5d716fca19a4252ad90e7b8ec7cc8d2\", \"requestInfo\":
        {\"ipAddress\": \"204.16.5.19\", \"path\": \"/activate\", \"user\":
        \"TheDoctor\"}}”

}
```

Becomes:

```
{
        "requestID": "b5d716fca19a4252ad90e7b8ec7cc8d2",
        "requestInfo": {
                "ipAddress": "204.16.5.19",
                "path": "/activate",
                "user": "TheDoctor"
        }
}
```

Logging Use Case – Parse Common Log Formats

```
[SERVICE]
        Log_Level info
        Parsers_File /fluent-bit/parsers/parsers.conf
[FILTER]
        Name parser
        Match *
        Key_Name log
        Parser nginx
        Reserve_Data True
```

Logging Use Case – Parse Common Log Formats

This:

```
272.1.0.1 - - [03/Oct/2019:00:06:20 +0000] "GET / HTTP/1.1" 200 612 "-" "curl/7.54.0" ”-"
```

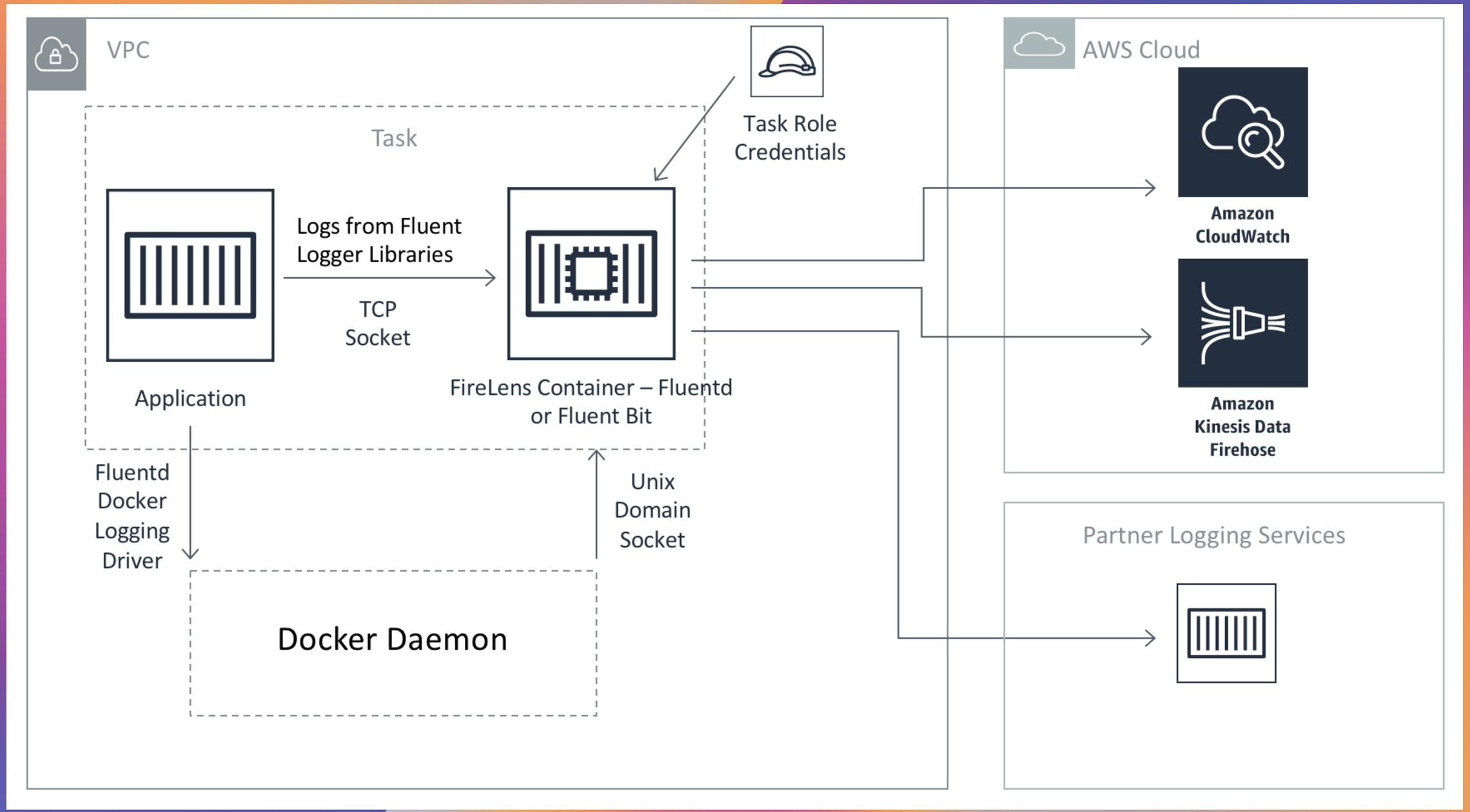
Becomes:

```
{
      "remote": ”272.1.0.1",
      "host": "-",
      "user": "-",
      "method": "GET",
      "path": "/",
      "code": "200",
      "size": "612",
      "referer": "-",
      "agent": "curl/7.54.0"
}
```

Logging Use Case – Send logs to multiple destinations

```
[OUTPUT]
        Name firehose
        Match * region
        us-west-2
        delivery_stream
        stream-one
[OUTPUT]
        Name firehose
        Match * region
        us-west-2
        delivery_stream stream-two
```

FireLens Internals Diagram



Daemon Service and Aggegator Service

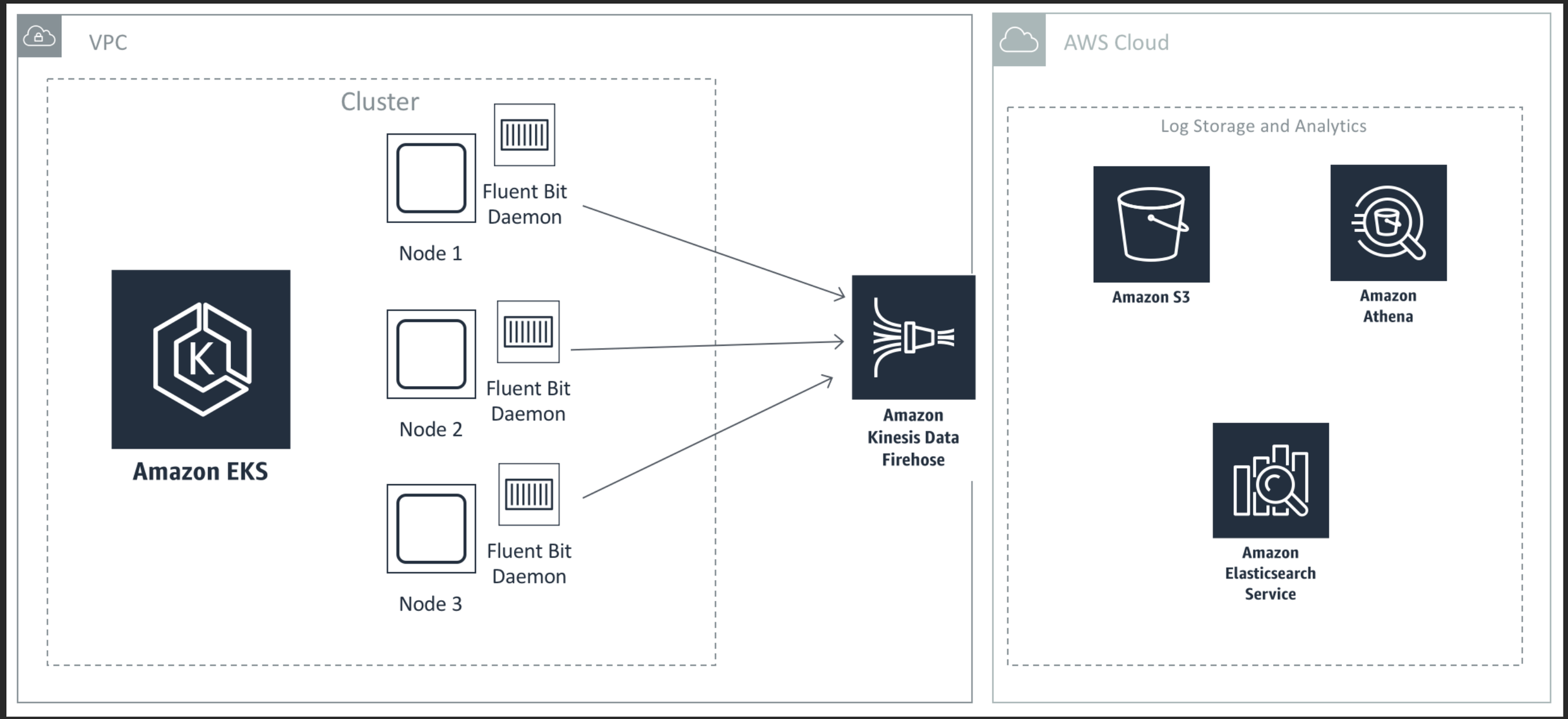
AWS re:Invent

aws

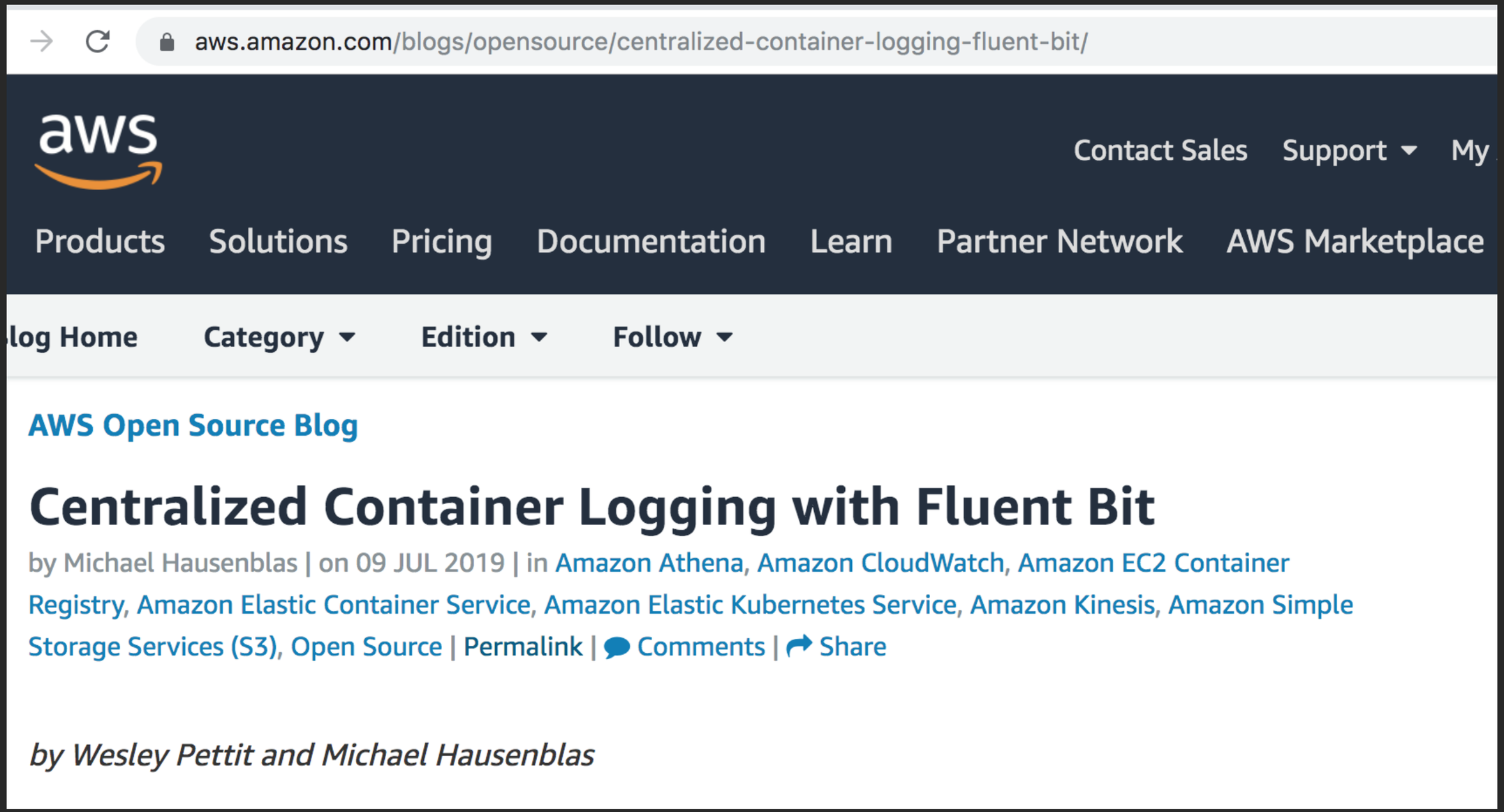
Deployment Modes – Daemon Service

- Run one instance of Fluentd or Fluent Bit per host
- Most popular on Kubernetes
- Use Fluent Bit because its resource usage is lower

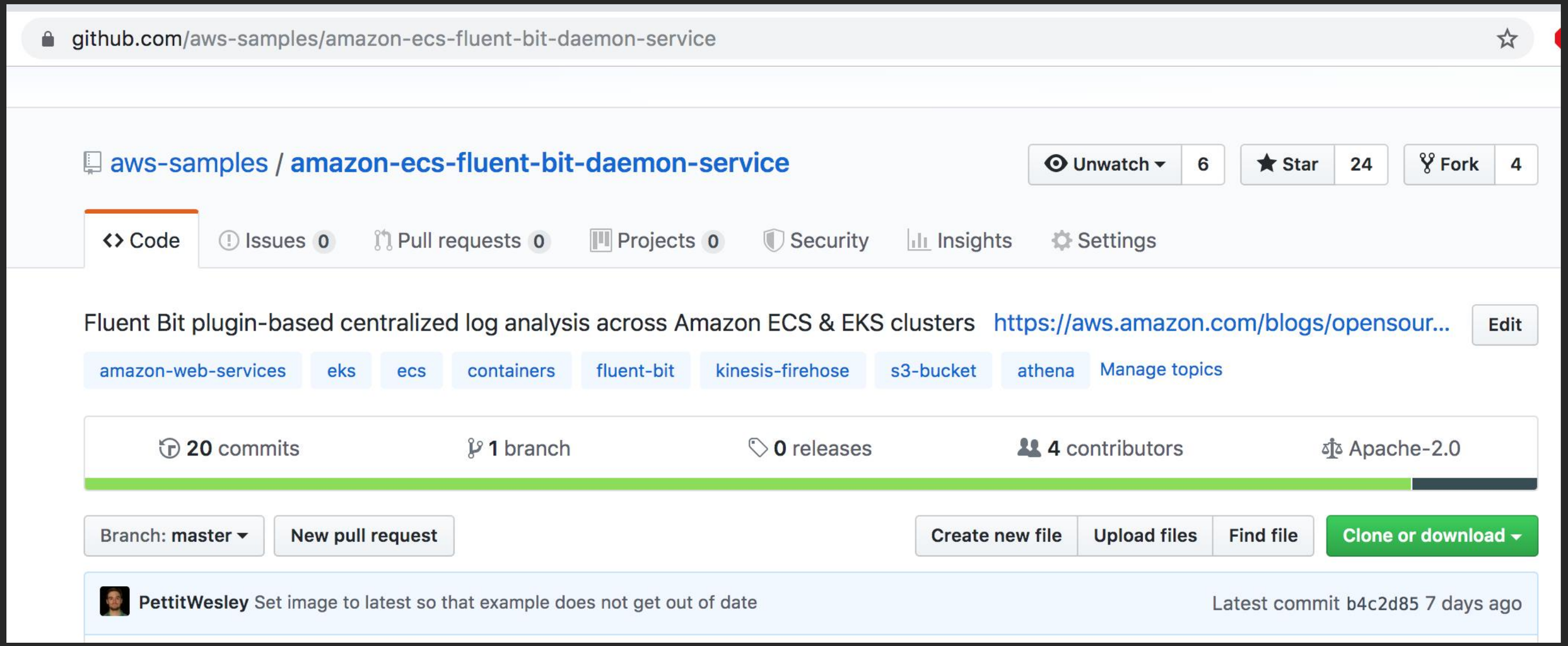
Deployment Modes – Daemon Service



Deployment Modes – Daemon Service



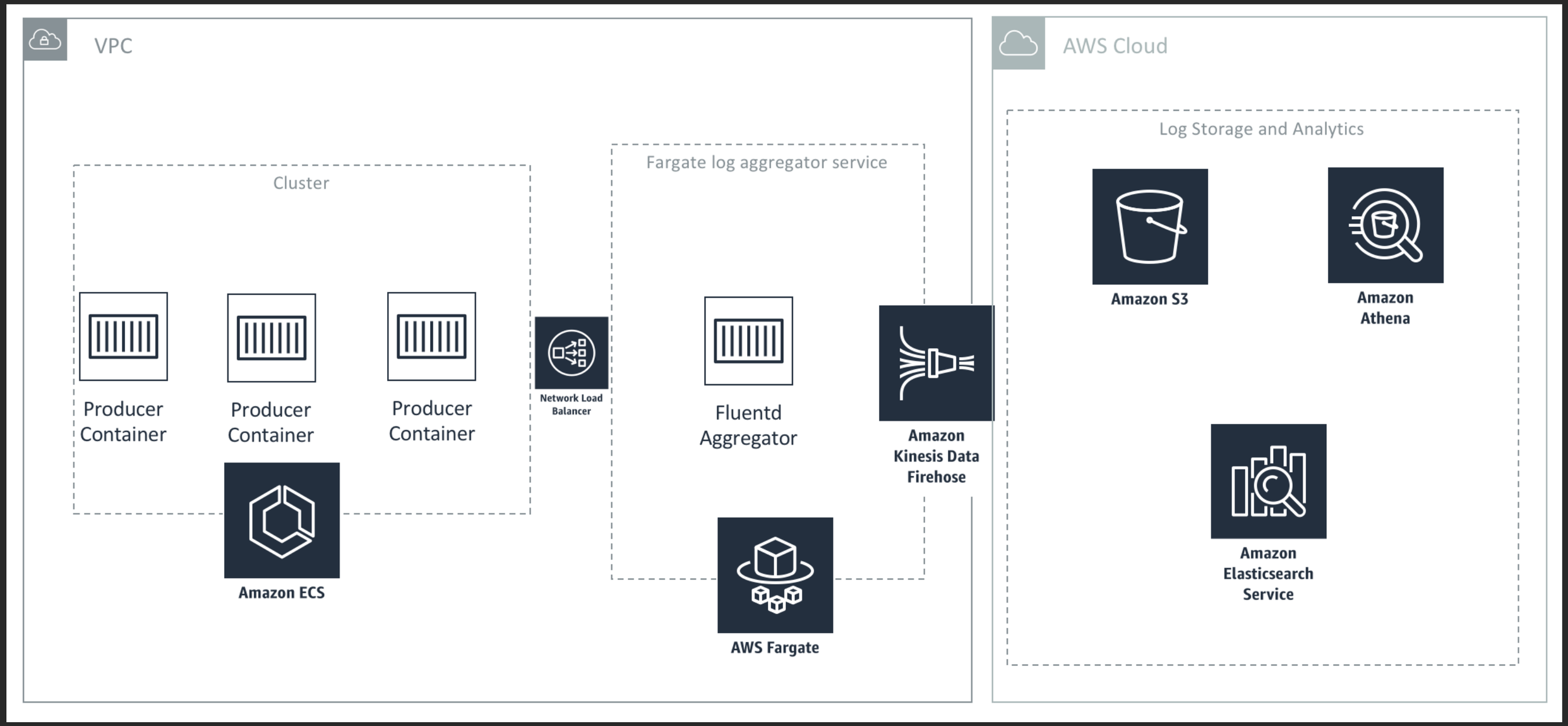
Deployment Modes – Daemon Service



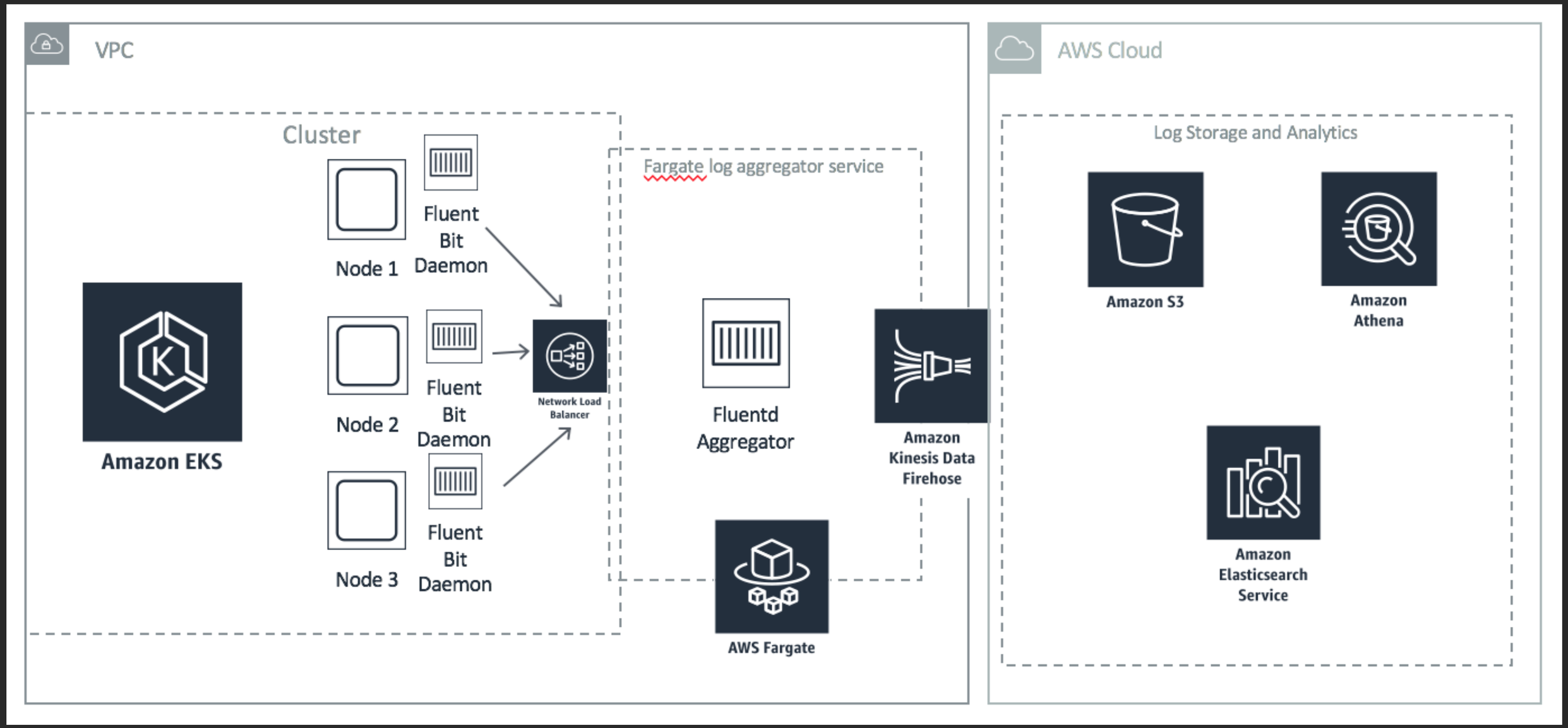
Deployment Modes – Aggregator Service

- Run a centralized Fluentd service that collects all logs from a VPC or Region

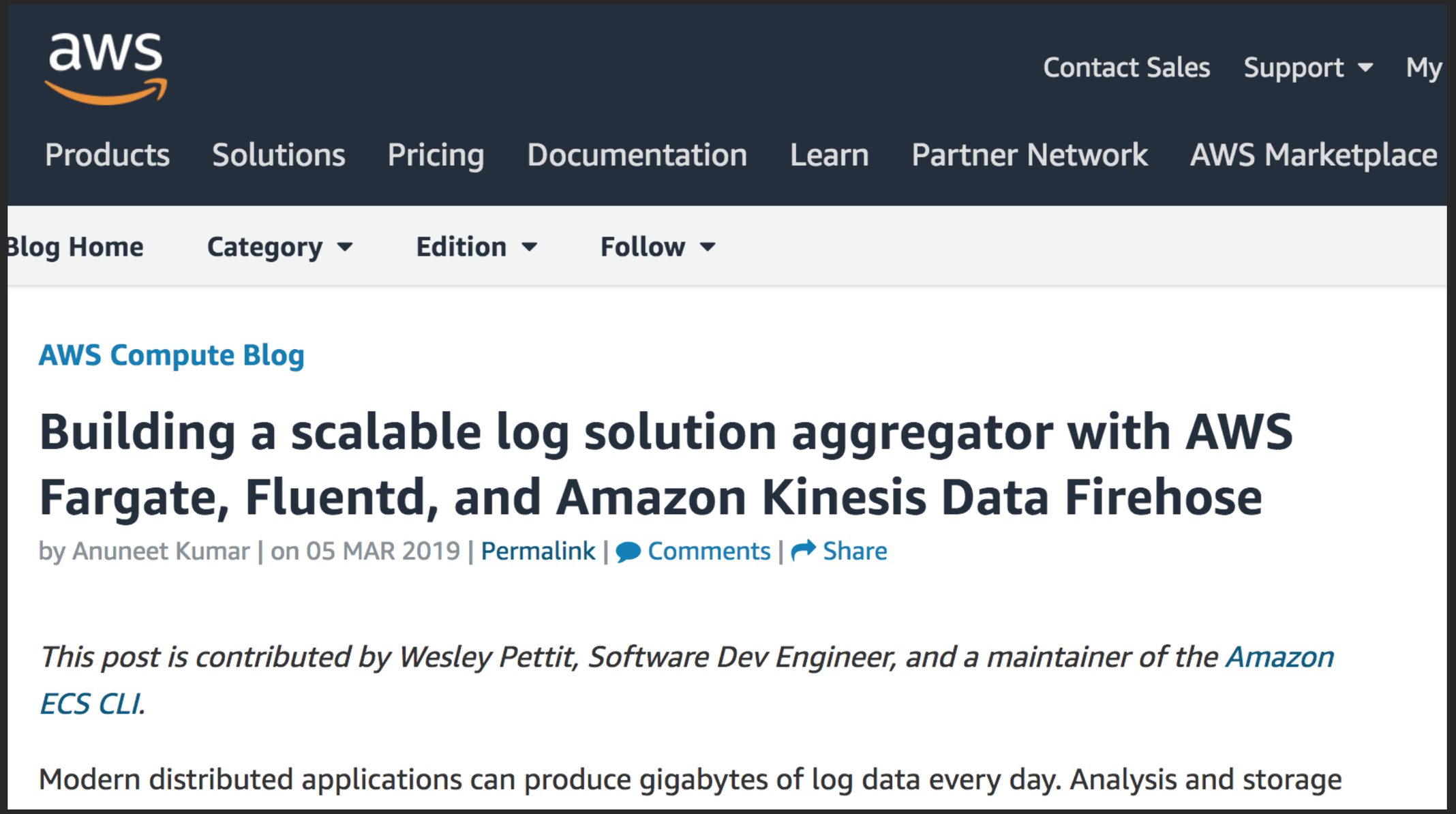
Deployment Modes – Aggregator Service (Amazon ECS)



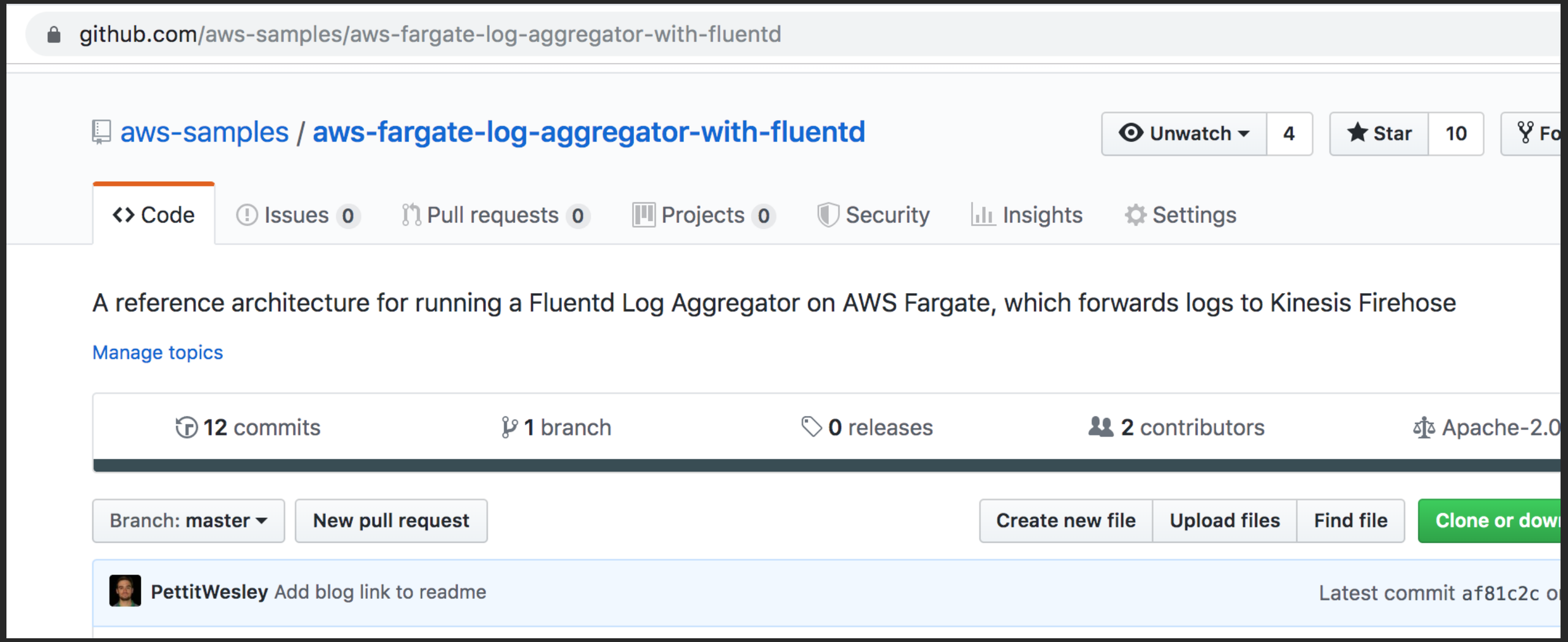
Deployment Modes – Aggregator Service (Amazon EKS)



Deployment Modes – Aggregator Service



Deployment Modes – Aggregator Service

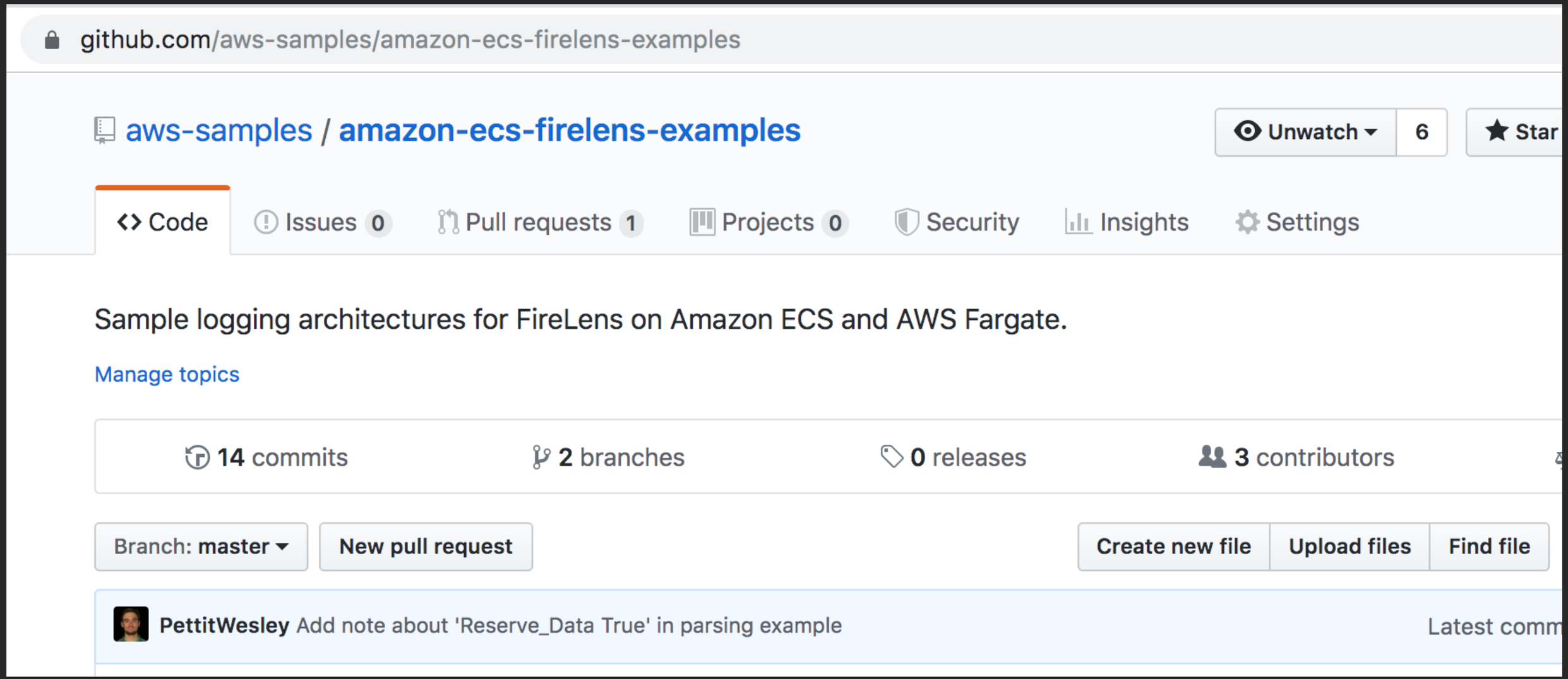


FireLens Example Walkthrough

FireLens for Amazon ECS

- A Fluent Bit or Fluentd side-car managed by AWS
- Available for Amazon ECS on Amazon EC2 and AWS Fargate

FireLens for Amazon ECS



FireLens Example Walkthrough

- We will do the following
  - Parse JSON logs
  - Inject metadata into logs
  - Filter logs with regular expressions
  - Send logs to Amazon S3 via Amazon Kinesis Data Firehose

FireLens Example Walkthrough - JSON

Container Outputs this to standard out:


```
{
  "requestID": "b5d716fca19a4252ad90e7b8ec7cc8d2",
  "requestInfo": {
    "ipAddress": "204.16.5.19",
    "path": "/activate",
    "user": "TheDoctor",
    "level": "error"
  }
}
```

FireLens Example Walkthrough - JSON

But it arrives at the logging agent as:


```
{
    "log": "{\"requestID\": \"b5d716fca19a4252ad90e7b8ec7cc8d2\",
\"requestInfo\": {\"ipAddress\": \"204.16.5.19\", \"path\": \"/activate\", \"user\":
\"TheDoctor\", \"level\": \"error\"}}"
}
```

FireLens Example – Output to Kinesis Data Firehose

```
"logConfiguration": {
        "logDriver":"awsfirelens",
        "options": {
            "Name": "firehose",
            "region": "ap-south-1",
            "delivery_stream": "demo-stream",
        }
},
```

FireLens Example – Output to Kinesis Data Firehose

```
[OUTPUT]
    Name firehose
    Match *
    region ap-south-1
    delivery_stream demo-stream
```

FireLens Example – Filter Logs with a Regex

```
"logConfiguration": {
        "logDriver":"awsfirelens",
        "options": {
            "Name": "firehose",
            "region": "ap-south-1",
            "delivery_stream": "demo-stream",
            "include-pattern": "[Ee]rror"
        }
},
```

FireLens Example – Filter Logs with a Regex

```
[FILTER]
    Name grep
    Match app-firelens*
    Regex log [Ee]rror
```

FireLens Example – Fluent Bit Side Car

```
{
                "essential": true,
                "image": ”amazon/aws-for-fluent-bit:1.3.2",
                "name": "log_router",
                "firelensConfiguration": {
                        "type": "fluentbit",
                }
}
```

FireLens Example – Inject Custom Config

```
{
                "essential": true,
                "image": ”amazon/aws-for-fluent-bit:1.3.2",
                "name": "log_router",
                "firelensConfiguration": {
                        "type": "fluentbit",
                         "options": {
                                "config-file-type": "s3",
                                "config-file-value":
"arn:aws:s3:::mybucket/fluent.conf"
                        }
                }
}
```

FireLens Example – Custom Config - JSON

```
[SERVICE]
    Log_Level debug
    Parsers_File /fluent-bit/parsers/parsers.conf

[FILTER]
    Name parser
    Match *
    Key_Name log
    Parser json
    Reserve_Data True
```

FireLens Example – Custom Config – Inject a Field

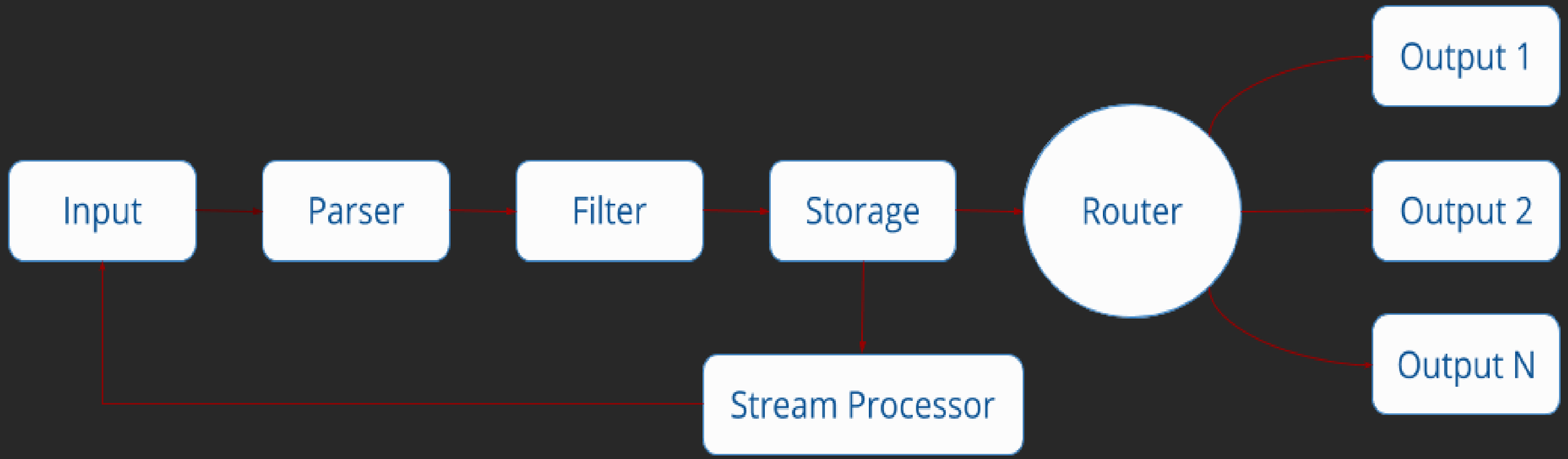
```
[FILTER]
    Name record_modifier
    Match *
    Record app-version ${APP_VERSION}
    Reserve_Data True
```

Fluent Bit Stream Processing

AWS re:Invent

aws

Stream Processing



Fluent Bit + Stream Processing

- Input plugins generate streams of data
- After storage phase, optional stream processing
- Perform SQL queries
- Create **new streams** using results from the queries

Fluent Bit + Stream Processing

SQL: SELECT statement

- Select all keys

  SELECT * FROM STREAM:apache;
- Select specific keys

  SELECT host, status, size FROM STREAM:apache;

Fluent Bit + Stream Processing

SQL: Aggregation functions

- COUNT()
- SUM()
- MAX()
- MIN()
- AVG()

Fluent Bit + Stream Processing

Windowing

```
SELECT
    device_id,
    AVG(temp)
FROM
    STREAM:devices WINDOW TUMBLING (5 second)
GROUP BY
    device_id
```

Fluent Bit + Stream Processing

Re-ingest data into Fluent Bit log pipeline

- Create a stream from the results of a query

  CREATE STREAM users AS SELECT user from STREAM:apache;

- Tag and re-ingest stream into Fluent Bit data pipeline

  CREATE STREAM users **WITH (tag='users')** AS SELECT user from STREAM:apache;

”Forking Logs” with Stream Processing

Tutorial on “Forking” Logs to Multiple Destinations

AWS Open Source Blog

Splitting an application’s logs into multiple streams: a Fluent tutorial

by Wesley Pettit | on 20 NOV 2019 | in Advanced (300), Amazon CloudWatch, Amazon Elastic Container Service, Amazon Elastic Kubernetes Service, AWS Fargate, Containers, Go, Open Source | Permalink | Comments | Share

Not all logs are of equal importance. Some require real-time analytics, others simply need to be stored long term so that they can be analyzed if needed. In this tutorial, I will show three different

A practical example – Fluent Bit Stream Processing

```
[STREAM_TASK]
    Name   parse_test
    Exec   CREATE STREAM results WITH (tag='error') AS SELECT
           * from TAG:'app*' WHERE level = 'error';
```

A practical example – Fluent Bit Stream Processing

```
CREATE STREAM results
WITH (tag='error')
AS SELECT * from TAG:'app*'
WHERE level = 'error';
```

A practical example – Fluent Bit Stream Processing

```
CREATE STREAM results
WITH (tag='error')
AS SELECT * from TAG:'app*'
WHERE level = 'error';
```

A practical example – Fluent Bit Stream Processing

```
CREATE STREAM results
WITH (tag='error')
AS SELECT * from TAG:'app*'
WHERE level = 'error';
```

A practical example – Fluent Bit Stream Processing

```
CREATE STREAM results
WITH (tag='error')
AS SELECT * from TAG:'app*'
WHERE level = 'error';
```

Additional Fluent Bit Slides

Fluent Bit Roadmap

v1.4 – January 2020

- Core
  - Config maps
  - Old charset encoding to UTF-8
- Stream Processor
  - Snapshots
  - Machine learning algorithms
- Performance improvements and more connectors!

Fluent Bit – Filtering Logs

Optional filtering with Lua scripts!

```
function cb_replace(tag, timestamp, record)
        new_record = {}
        new_record["new"] = 12345
        new_record[”old"] = record
        return 1, timestamp, new_record
end
```

Fluent Bit – Monitoring

Metrics and Prometheus support