

LFS242 - Cloud Native Logging with Fluentd

Lab 10 – Fluent Bit and Fluentd

Fluent Bit is a lightweight alternative to Fluentd from the original developers of Fluentd, Arm Treasure Data. It shares many functions with Fluentd, but focuses on high performance with a very small resource footprint - ideal for resource limited environments like embedded systems, IoT devices, and containers. As a part of the Fluentd project ecosystem, it can act as a lighter-weight forwarder to its larger cousin Fluentd, allowing both programs to take advantage of the speed and leanness of Fluent Bit and the flexibility provided by Fluentd.

This lab is designed to be completed on an Ubuntu 20.04 system. The labs install and configure software, so a cloud instance or local VM is recommended.

Objectives

- Understand the similarities and differences between Fluent Bit and Fluentd
- Configure Fluent Bit to work together with Fluentd
- See an example of a Fluent Bit and Fluentd working as part of a fully functional log collection and analysis stack

0. Prepare the lab system

A Fluentd instance that can be freely modified is required for this lab. Create and run the following script in your VM to quickly set up Fluentd:

```
ubuntu@labsys:~$ nano fluentd-setup

ubuntu@labsys:~$ cat fluentd-setup

#!/bin/sh

sudo apt update
sudo apt install ruby-full ruby-dev libssl-dev libreadline-dev zlib1g-dev gcc make -y
sudo gem install bundle
sudo gem install fluentd

ubuntu@labsys:~$ chmod +x fluentd-setup

ubuntu@labsys:~$ ./fluentd-setup

ubuntu@labsys:~$ fluentd --version

fluentd 1.14.6

ubuntu@labsys:~$
```

This lab will also be using Docker to run instances of Fluentd, Elasticsearch, and Kibana, so an installation of Docker will be required.

Install Docker with the quick installation script for Debian:

```
ubuntu@labsys:~$ wget -O - https://get.docker.com | sh

...

ubuntu@labsys:~$
```

All `docker` commands will be run with `sudo` in this lab so there is no need to set up rootless interaction.

1. Installing Fluent Bit

Fluent Bit is coded entirely in C and this lack of external dependencies on Ruby allows Fluent Bit to have a smaller footprint when compared to Fluentd. Fluent Bit is available as a package through many package managers, including Ubuntu's **apt**. The installation is very simple and can be done with a convenient installation script which installs the latest version:

```
ubuntu@ip-172-31-10-56:~$ mkdir ~/fluent-bit && cd $_

ubuntu@ip-172-31-10-56:~/fluent-bit$ curl https://raw.githubusercontent.com/fluent/fluent-bit/master/install.sh | sh

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
100 3105  100 3105    0     0  32343      0 --:--:-- --:--:-- --:--:-- 32343
=====
Fluent Bit Installation Script
=====
This script requires superuser access to install packages.
You will be prompted for your password by sudo.
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
100 3175  100 3175    0     0  28603      0 --:--:-- --:--:-- --:--:-- 28603
deb [signed-by=/usr/share/keyrings/fluentbit-keyring.gpg] https://packages.fluentbit.io/ubuntu/focal
focal main
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:4 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Hit:5 https://download.docker.com/linux/ubuntu focal InRelease
Hit:6 https://packages.cloud.google.com/apt kubernetes-xenial InRelease
Get:7 https://packages.fluentbit.io/ubuntu/focal focal InRelease [10.9 kB]
Get:8 https://packages.fluentbit.io/ubuntu/focal focal/main amd64 Packages [14.2 kB]
Fetched 361 kB in 1s (456 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libpq5
The following NEW packages will be installed:
  fluent-bit libpq5
0 upgraded, 2 newly installed, 0 to remove and 62 not upgraded.
Need to get 20.2 MB of archives.
After this operation, 59.2 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal-updates/main amd64 libpq5 amd64 12.11-0ubuntu0.20.04.1 [117 kB]
Get:2 https://packages.fluentbit.io/ubuntu/focal focal/main amd64 fluent-bit amd64 1.9.4 [20.1 MB]
Fetched 20.2 MB in 1s (19.2 MB/s)
Selecting previously unselected package libpq5:amd64.
(Reading database ... 90784 files and directories currently installed.)
Preparing to unpack .../libpq5_12.11-0ubuntu0.20.04.1_amd64.deb ...
Unpacking libpq5:amd64 (12.11-0ubuntu0.20.04.1) ...
Selecting previously unselected package fluent-bit.
Preparing to unpack .../fluent-bit_1.9.4_amd64.deb ...
Unpacking fluent-bit (1.9.4) ...
Setting up libpq5:amd64 (12.11-0ubuntu0.20.04.1) ...
Setting up fluent-bit (1.9.4) ...
Processing triggers for libc-bin (2.31-0ubuntu9.7) ...

Installation completed. Happy Logging!
```

Fluent Bit should now be installed.

Verify it by running `fluent-bit --version`:

```
ubuntu@labsys:~/fluent-bit$ sudo ln -s /opt/fluent-bit/bin/fluent-bit /usr/bin/fluent-bit

ubuntu@labsys:~/fluent-bit$ fluent-bit --version

Fluent Bit v1.9.4
Git commit:

ubuntu@labsys:~/fluent-bit$
```

2. Running and Configuring Fluent Bit

Fluent Bit shares functional concepts with Fluentd and can interoperate with Fluentd, but unlike Fluentd it is written completely in C, so there are some key differences that need to be explored before using Fluent Bit.

2a. Running Fluent Bit imperatively

Fluent Bit can be run with a certain configuration imperatively from the command line. This is recommended for testing configurations before committing them to more permanent configuration files. One advantage of running Fluent Bit this way is the that it eschews the need for storage space on the Fluent Bit host since no configuration file is required.

Return to your home directory, and dump the `--help` output for Fluent Bit:

```
ubuntu@labsys:~/fluent-bit/build$ mkdir ~/lab10 && cd $_

ubuntu@labsys:~/lab10$ fluent-bit --help

Usage: fluent-bit [OPTION]

Available Options
  -b --storage_path=PATH  specify a storage buffering path
  -c --config=FILE        specify an optional configuration file
  -d, --daemon            run Fluent Bit in background mode
  -D, --dry-run           dry run
  -f, --flush=SECONDS     flush timeout in seconds (default: 1)
  -C, --custom=CUSTOM     enable a custom plugin
  -i, --input=INPUT       set an input
  -F --filter=FILTER      set a filter
  -m, --match=MATCH       set plugin match, same as '-p match=abc'
  -o, --output=OUTPUT     set an output
  -p, --prop="A=B"        set plugin configuration property
  -R, --parser=FILE       specify a parser configuration file
  -e, --plugin=FILE       load an external plugin (shared lib)
  -l, --log_file=FILE     write log info to a file
  -t, --tag=TAG           set plugin tag, same as '-p tag=abc'
  -T, --sp-task=SQL       define a stream processor task
  -v, --verbose           increase logging verbosity (default: info)
  -vv                    trace mode (available)
  -w, --workdir           set the working directory
  -H, --http              enable monitoring HTTP server
  -P, --port              set HTTP server TCP port (default: 2020)
  -s, --coro_stack_size  set coroutines stack size in bytes (default: 24576)
  -q, --quiet             quiet mode
  -S, --sosreport         support report for Enterprise customers
  -V, --version           show version number
```

```

-h, --help          print this help

Inputs
  cpu                CPU Usage
  mem                Memory Usage
  thermal            Thermal
...

ubuntu@labsys:~/lab10$

```

The selection of plugins available represent plugins that are built into the Fluent Bit binary when it was installed. Plugins can be included or excluded at build time, meaning a Fluent Bit instance can be built from the ground up to be as lightweight as possible.

The `-i`, `-o`, and `-F` flags are of note here. These flags allow a user to specify plugins that a Fluent Bit instance can be launched with.

Run a Fluent Bit instance that will send CPU usage metrics to STDOUT:

```

ubuntu@labsys:~/lab10$ fluent-bit -i cpu -o stdout

Fluent Bit v1.9.4
* Copyright (C) 2015-2022 The Fluent Bit Authors
* Fluent Bit is a CNCF sub-project under the umbrella of Fluentd
* https://fluentbit.io

[2022/06/14 22:37:07] [ info] [fluent bit] version=1.9.4, commit=, pid=205042
[2022/06/14 22:37:07] [ info] [storage] version=1.2.0, type=memory-only, sync=normal,
checksum=disabled, max_chunks_up=128
[2022/06/14 22:37:07] [ info] [cmetrics] version=0.3.1
[2022/06/14 22:37:07] [ info] [sp] stream processor started
[2022/06/14 22:37:07] [ info] [output:stdout:stdout.0] worker #0 started
[0] cpu.0: [1655246227.851492750, {"cpu_p"=>0.000000, "user_p"=>0.000000, "system_p"=>0.000000,
"cpu0.p_cpu"=>0.000000, "cpu0.p_user"=>0.000000, "cpu0.p_system"=>0.000000, "cpu1.p_cpu"=>0.000000,
"cpu1.p_user"=>0.000000, "cpu1.p_system"=>0.000000}]
[0] cpu.0: [1655246228.851460098, {"cpu_p"=>0.000000, "user_p"=>0.000000, "system_p"=>0.000000,
"cpu0.p_cpu"=>0.000000, "cpu0.p_user"=>0.000000, "cpu0.p_system"=>0.000000, "cpu1.p_cpu"=>1.000000,
"cpu1.p_user"=>1.000000, "cpu1.p_system"=>0.000000}]

...

```

The emitted events share the same basic structure as a Fluentd instance's events:

- The event receives a tag, `cpu.0`
- A timestamp, in unix time, is attached to the event
- A record section enclosed in `{}` that contains the actual event data. In this case, it is output to STDOUT in msgpack.

Terminate the instance with `Ctrl C`.

```

...

^C

[2022/06/14 22:37:14] [engine] caught signal (SIGINT)
[2022/06/14 22:37:14] [ info] [input] pausing cpu.0
[2022/06/14 22:37:14] [ warn] [engine] service will shutdown in max 5 seconds
[0] cpu.0: [1655246233.851476002, {"cpu_p"=>0.000000, "user_p"=>0.000000, "system_p"=>0.000000,
"cpu0.p_cpu"=>0.000000, "cpu0.p_user"=>0.000000, "cpu0.p_system"=>0.000000, "cpu1.p_cpu"=>0.000000,
"cpu1.p_user"=>0.000000, "cpu1.p_system"=>0.000000}]

```

```
[2022/06/14 22:37:14] [ info] [engine] service has stopped (0 pending tasks)
[2022/06/14 22:37:14] [ info] [output:stdout:stdout.0] thread worker #0 stopping...
[2022/06/14 22:37:14] [ info] [output:stdout:stdout.0] thread worker #0 stopped
```

```
ubuntu@labsys:~/lab10$
```

Imperatively run configurations can configure plugins using the `-p` flag following the `-i`, `-o` or `-F` flags. The argument for the `-p` flag is the `key=value` for a plugin's setting.

Run Fluent Bit again with the forward plugin listening on port 31955 and outputting to STDOUT:

```
ubuntu@labsys:~/lab10$ fluent-bit -i forward -p port=31955 -o stdout -p format=json_lines
```

```
Fluent Bit v1.9.4
```

```
* Copyright (C) 2015-2022 The Fluent Bit Authors
```

```
* Fluent Bit is a CNCF sub-project under the umbrella of Fluentd
```

```
* https://fluentbit.io
```

```
[2022/06/14 22:37:58] [ info] [fluent bit] version=1.9.4, commit=, pid=205047
```

```
[2022/06/14 22:37:58] [ info] [storage] version=1.2.0, type=memory-only, sync=normal,
checksum=disabled, max_chunks_up=128
```

```
[2022/06/14 22:37:58] [ info] [cmetrics] version=0.3.1
```

```
[2022/06/14 22:37:58] [ info] [input:forward:forward.0] listening on 0.0.0.0:31955
```

```
[2022/06/14 22:37:58] [ info] [sp] stream processor started
```

```
[2022/06/14 22:37:58] [ info] [output:stdout:stdout.0] worker #0 started
```

- `-i forward` specifies that the forward plugin should be used to listen for Fluent protocol traffic
- `-p port=31955` immediately follows `-i forward` flag, telling the forward plugin to listen on port 31955
- `-o stdout` will submit events to STDOUT
- `-p format=json_lines` configures the stdout plugin to output events in JSON format rather than msgpack

In another terminal, send an event to Fluent Bit using `fluent-cat` :

```
ubuntu@labsys:~$ cd ~/lab10/
```

```
ubuntu@labsys:~/lab10$ echo '{"lfs242":"Hello from Fluent Bit"}' | fluent-cat mod10.lab -p 31955
```

```
ubuntu@labsys:~/lab10$
```

In the Fluent Bit terminal, the message is received:

```
{"date":1655246294.582881,"lfs242":"Hello from Fluent Bit"}
```

With the `json_lines` configuration for the STDOUT plugin, Fluent Bit outputs a JSON map containing the date timestamp and the message.

When you're done, use `CTRL C` in the Fluent Bit terminal to shut down the instance:

```
^C
```

```
[2022/06/14 22:38:57] [engine] caught signal (SIGINT)
```

```
[2022/06/14 22:38:57] [ info] [input] pausing forward.0
```

```
[2022/06/14 22:38:57] [ warn] [engine] service will shutdown in max 5 seconds
```

```
[2022/06/14 22:38:57] [ info] [engine] service has stopped (0 pending tasks)
```

```
[2022/06/14 22:38:57] [ info] [output:stdout:stdout.0] thread worker #0 stopping...
```

```
[2022/06/14 22:38:57] [ info] [output:stdout:stdout.0] thread worker #0 stopped
```

```
ubuntu@labsys:~/lab10$
```

Try running the following imperative Fluent Bit configurations:

- Send memory usage information to a file, `mod10.mem.txt`
- Tail a file, `log.txt`, and send output as `msgpack` to `stdout`

You should now know how to run Fluent Bit imperatively from the command line with a variety of configurations.

2b. Creating a Fluent Bit configuration file

Fluent Bit's configuration format differs greatly from Fluentd, though the overall principal is the same: inputs, filters and outputs are configured under their own sections. Each section declares a single plugin that dictates how an event is received, processed and delivered to a configured destination. Plugins are configured with a set of parameters, known as entries, that make up a majority of a section's syntax. Each entry is a key-value pair that influences how a plugin ultimately behaves.

Create a `fluent-bit.conf` file:

```
ubuntu@labsys:~/lab10$ nano fluent-bit.conf && cat $_

[INPUT]
name cpu

[OUTPUT]
name file
path ~/lab10/output.txt

ubuntu@labsys:~/lab10$
```

- `[INPUT]` is the equivalent of a Fluentd `<source>` directive and is the section that declares an event source
- `name cpu` is the plugin declaration, equivalent to the `@type` parameter in Fluentd
- `[OUTPUT]` is the functional equivalent to the `<match>` directive and declares an event destination. It will write to a file using the `out_file` plugin.
- `path ~/lab10/output.txt` entry tells the `out_file` plugin to write to `output.txt` in the working directory

There are no subsections that need to be configured. To further configure a plugin, only additional entries need to be added.

Try to run it:

```
ubuntu@labsys:~/lab10$ fluent-bit -c fluent-bit.conf

Fluent Bit v1.9.4
* Copyright (C) 2015-2022 The Fluent Bit Authors
* Fluent Bit is a CNCF sub-project under the umbrella of Fluentd
* https://fluentbit.io

[2022/06/14 22:42:12] [error] [config] indentation level is too low
[2022/06/14 22:42:12] [error] [config] error in fluent-bit.conf:6: invalid indentation level
[2022/06/14 22:42:12] [error] [config] section 'input' is missing the 'name' property
[2022/06/14 22:42:12] [error] configuration file contains errors, aborting.

ubuntu@labsys:~/lab10$
```

Fluent Bit enforces a strict indentation schema in its configuration files. The configuration file assembled earlier did not indent the plugin configuration entries. By default, Fluent Bit expects plugin entries to be indented with 4 spaces.

Revise the `fluent-bit.conf` file, adding four spaces to each entry under both `[INPUT]` and `[OUTPUT]` sections:

```
ubuntu@labsys:~/lab10$ nano fluent-bit.conf && cat $_

[INPUT]
    name cpu

[OUTPUT]
    name file
    path ~/lab10/cpu-output

ubuntu@labsys:~/lab10$
```

Give this one a try:

```
ubuntu@labsys:~/lab10$ fluent-bit -c fluent-bit.conf

[INPUT]
    name cpu

[OUTPUT]
    name file
    path ~/lab10/output.txt
ubuntu@labsys:~/lab10$ fluent-bit -c fluent-bit.conf
Fluent Bit v1.9.4
* Copyright (C) 2015-2022 The Fluent Bit Authors
* Fluent Bit is a CNCF sub-project under the umbrella of Fluentd
* https://fluentbit.io

[2022/06/14 22:42:51] [ info] [fluent bit] version=1.9.4, commit=, pid=205081
[2022/06/14 22:42:51] [ info] [storage] version=1.2.0, type=memory-only, sync=normal,
checksum=disabled, max_chunks_up=128
[2022/06/14 22:42:51] [ info] [cmetrics] version=0.3.1
[2022/06/14 22:42:51] [ info] [sp] stream processor started
[2022/06/14 22:42:51] [ info] [output:file:file.0] worker #0 started
[2022/06/14 22:42:52] [error] [plugins/out_file/file.c:475 errno=2] No such file or directory
[2022/06/14 22:42:52] [error] [output:file:file.0] error opening: ~/lab10/output.txt/cpu.0

^C

[2022/06/14 22:42:53] [engine] caught signal (SIGINT)
[2022/06/14 22:42:53] [ info] [input] pausing cpu.0
[2022/06/14 22:42:53] [ warn] [engine] service will shutdown in max 5 seconds
[2022/06/14 22:42:53] [error] [plugins/out_file/file.c:475 errno=2] No such file or directory
[2022/06/14 22:42:53] [error] [output:file:file.0] error opening: ~/lab10/output.txt/cpu.0
[2022/06/14 22:42:53] [ info] [engine] service has stopped (0 pending tasks)
[2022/06/14 22:42:53] [ info] [output:file:file.0] thread worker #0 stopping...
[2022/06/14 22:42:53] [ info] [output:file:file.0] thread worker #0 stopped

ubuntu@labsys:~/lab10$
```

It failed, with the `out_file` plugin unable to create the destination. This is because, like `Fluentd`, `Fluent Bit` configurations do not support UNIX environment variables (like `$HOME`) or their aliases (Like `~`) being directly in the configuration file. In order to refer to the home directory, a full path must be used.

Change the home directory reference to a full path:

```
ubuntu@labsys:~/lab10$ nano fluent-bit.conf && cat $_
```

```
[INPUT]
  name cpu

[OUTPUT]
  name file
  path /home/ubuntu/lab10/cpu-output

ubuntu@labsys:~/lab10$
```

Try to run it now:

```
ubuntu@labsys:~/lab10$ mkdir ~/lab10/cpu-output

ubuntu@labsys:~/lab10$ fluent-bit -c fluent-bit.conf

...

[2022/06/14 22:43:41] [ info] [fluent bit] version=1.9.4, commit=, pid=205095
[2022/06/14 22:43:41] [ info] [storage] version=1.2.0, type=memory-only, sync=normal,
checksum=disabled, max_chunks_up=128
[2022/06/14 22:43:41] [ info] [cmetrics] version=0.3.1
[2022/06/14 22:43:41] [ info] [sp] stream processor started
[2022/06/14 22:43:41] [ info] [output:file:file.0] worker #0 started
```

Now it's running, and no errors are being reported.

In another terminal session, check the contents of the `~/lab10/cpu-output/` directory:

```
ubuntu@labsys:~/lab10$ ls -l ~/lab10/cpu-output/

total 4
-rw-rw-r-- 1 ubuntu ubuntu 4026 Jun 14 22:44 cpu.0

ubuntu@labsys:~/lab10$
```

output.txt has been created. Check the contents:

```
ubuntu@labsys:~/lab10$ tail -5 cpu-output/cpu.0

cpu.0: [1655246661.851526735,
{"cpu_p":0.0,"user_p":0.0,"system_p":0.0,"cpu0.p_cpu":0.0,"cpu0.p_user":0.0,"cpu0.p_system":0.0,"cpu
1.p_cpu":0.0,"cpu1.p_user":0.0,"cpu1.p_system":0.0}]
cpu.0: [1655246662.851512133,
{"cpu_p":0.0,"user_p":0.0,"system_p":0.0,"cpu0.p_cpu":0.0,"cpu0.p_user":0.0,"cpu0.p_system":0.0,"cpu
1.p_cpu":0.0,"cpu1.p_user":0.0,"cpu1.p_system":0.0}]
cpu.0: [1655246663.851535888,
{"cpu_p":0.0,"user_p":0.0,"system_p":0.0,"cpu0.p_cpu":0.0,"cpu0.p_user":0.0,"cpu0.p_system":0.0,"cpu
1.p_cpu":0.0,"cpu1.p_user":0.0,"cpu1.p_system":0.0}]
cpu.0: [1655246664.851534962,
{"cpu_p":0.0,"user_p":0.0,"system_p":0.0,"cpu0.p_cpu":0.0,"cpu0.p_user":0.0,"cpu0.p_system":0.0,"cpu
1.p_cpu":0.0,"cpu1.p_user":0.0,"cpu1.p_system":0.0}]
cpu.0: [1655246665.851524646,
{"cpu_p":0.0,"user_p":0.0,"system_p":0.0,"cpu0.p_cpu":0.0,"cpu0.p_user":0.0,"cpu0.p_system":0.0,"cpu
1.p_cpu":0.0,"cpu1.p_user":0.0,"cpu1.p_system":0.0}]

ubuntu@labsys:~/lab10$
```


Excellent! You should now know how to create a basic Fluent Bit configuration as well as imperatively run an instance of Fluent Bit. Remember that:

- The sections are named differently from Fluentd directives
- There are strict spacing and indentation requirements
- Unix Environment Variables (those prefixed with \$) cannot be written directly into configuration files

Terminate the Fluent Bit instance with `CTRL C` or `kill -f fluent-bit`:

```
^C
[2022/06/14 22:44:42] [engine] caught signal (SIGINT)
[2022/06/14 22:44:42] [ info] [input] pausing cpu.0
[2022/06/14 22:44:42] [ warn] [engine] service will shutdown in max 5 seconds
[2022/06/14 22:44:42] [ info] [engine] service has stopped (0 pending tasks)
[2022/06/14 22:44:42] [ info] [output:file:file.0] thread worker #0 stopping...
[2022/06/14 22:44:42] [ info] [output:file:file.0] thread worker #0 stopped

ubuntu@labsys:~/lab10$
```

3. Fluentd and Fluent Bit comparison

Fluent Bit aims to be much lighter weight than Fluentd, in both configuration complexity (as shown above) and resource footprint (which will be shown in the coming steps).

3a. Setting up Fluentd and Fluent Bit to work together:

Fluent Bit can work together with Fluentd in a multi-instance deployment, taking the role of an log forwarder. In this step, you will set up a multi-instance deployment with a Fluent Bit forwarder, Fluentd log forwarder and Fluentd log aggregator.

Create a simple Fluentd log aggregator configuration file:

```
ubuntu@labsys:~/lab10$ nano fd-aggregator.conf && cat $_

<source>
  @type forward
  port 24500
</source>

<match>
  @type stdout
</match>

ubuntu@labsys:~/lab10$
```

Run the log aggregator in a Docker container, mounting the `fd-aggregator.conf` above and running it with the `-o` flag to write the log aggregator's output to an easily accessible file:

```
ubuntu@labsys:~/lab10$ sudo docker run -d --network host --name aggregator1 \
-v $HOME/lab10/fd-aggregator.conf:/fluentd/etc/fd-aggregator.conf \
-e FLUENTD_CONF=fd-aggregator.conf \
fluent/fluentd:v1.14-1 fluentd -c /fluentd/etc/fd-aggregator.conf

...

40a279037688e5551328eb15088c98d8cebc1f219ed90eca2e1406cdee83b6a9
```

```

ubuntu@labsys:~/lab10$ sudo docker logs aggregator1 --tail 5

2022-06-14 23:06:55 +0000 [info]: #0 listening port port=24500 bind="0.0.0.0"
2022-06-14 23:06:55 +0000 [info]: #0 fluentd worker is now running worker=0
2022-06-14 23:06:55.191329748 +0000 fluent.info: {"pid":16,"ppid":7,"worker":0,"message":"starting
fluentd worker pid=16 ppid=7 worker=0"}
2022-06-14 23:06:55.191725212 +0000 fluent.info: {"port":24500,"bind":"0.0.0.0","message":"listening
port port=24500 bind=\"0.0.0.0\""}
2022-06-14 23:06:55.192198669 +0000 fluent.info: {"worker":0,"message":"fluentd worker is now
running worker=0"}

ubuntu@labsys:~/lab10$

```

There is now a Fluentd log aggregator instance waiting to receive events from log forwarders.

Next, prepare a Fluentd log forwarder configuration:

```

ubuntu@labsys:~/lab10$ touch app.log

ubuntu@labsys:~/lab10$ nano fd-forwarder.conf && cat $_

<source>
  @type tail
  path /home/ubuntu/lab10/app.log
  pos_file /tmp/app.log.pos
  tag fd.forward
  <parse>
    @type none
  </parse>
</source>

<match>
  @type forward
  <server>
    name aggregator1
    host 127.0.0.1
    port 24500
  </server>
</match>

ubuntu@labsys:~/lab10$

```

Since the log aggregator container was run with the `--network host`, it can be accessed via the local IP: 127.0.0.1.

Then run the Fluentd log forwarder:

```

ubuntu@labsys:~/lab10$ fluentd -c fd-forwarder.conf

...

2022-06-14 23:07:31 +0000 [info]: #0 starting fluentd worker pid=205262 ppid=205257 worker=0
2022-06-14 23:07:31 +0000 [info]: #0 following tail of /home/ubuntu/lab10/app.log
2022-06-14 23:07:31 +0000 [info]: #0 fluentd worker is now running worker=0

```

When running the log forwarder, make sure that it is able to find the app.log file, which was created before the log forwarder configuration was written.

In a new terminal window, create a log forwarder instance of Fluent Bit with a configuration file that's tailing the same file as the Fluentd log forwarder:

```
ubuntu@labsys:~$ cd lab10

ubuntu@labsys:~/lab10$ nano fb-forwarder.conf && cat $_

[INPUT]
  name tail
  path /home/ubuntu/lab10/app.log

[OUTPUT]
  name forward
  host 127.0.0.1
  port 24500

ubuntu@labsys:~/lab10$
```

Notice how much simpler the Fluent Bit log forwarder file is compared to the Fluentd log forwarder's file:

- There is no `<parse>` -equivalent entry in the Fluent Bit configuration
- The tail input plugin for Fluent Bit does not record its last read position to a file

Run Fluent Bit, using the `-v` flag to get higher verbosity:

```
ubuntu@labsys:~/lab10$ fluent-bit -c fb-forwarder.conf -v

Fluent Bit v1.9.4
* Copyright (C) 2015-2022 The Fluent Bit Authors
* Fluent Bit is a CNCF sub-project under the umbrella of Fluentd
* https://fluentbit.io

[2022/06/14 23:08:02] [ info] Configuration:
[2022/06/14 23:08:02] [ info] flush time      | 1.000000 seconds
[2022/06/14 23:08:02] [ info] grace         | 5 seconds
[2022/06/14 23:08:02] [ info] daemon        | 0
[2022/06/14 23:08:02] [ info] _____
[2022/06/14 23:08:02] [ info] inputs:
[2022/06/14 23:08:02] [ info]   tail
[2022/06/14 23:08:02] [ info] _____
[2022/06/14 23:08:02] [ info] filters:
[2022/06/14 23:08:02] [ info] _____
[2022/06/14 23:08:02] [ info] outputs:
[2022/06/14 23:08:02] [ info]   forward.0
[2022/06/14 23:08:02] [ info] _____
[2022/06/14 23:08:02] [ info] collectors:
[2022/06/14 23:08:02] [ info] [fluent bit] version=1.9.4, commit=, pid=205270
[2022/06/14 23:08:02] [debug] [engine] coroutine stack size: 24576 bytes (24.0K)
[2022/06/14 23:08:02] [ info] [storage] version=1.2.0, type=memory-only, sync=normal,
checksum=disabled, max_chunks_up=128
[2022/06/14 23:08:02] [ info] [cmetrics] version=0.3.1
[2022/06/14 23:08:02] [debug] [tail:tail.0] created event channels: read=21 write=22
[2022/06/14 23:08:02] [debug] [input:tail:tail.0] flb_tail_fs_inotify_init() initializing inotify
tail input
[2022/06/14 23:08:02] [debug] [input:tail:tail.0] inotify watch fd=27
[2022/06/14 23:08:02] [debug] [input:tail:tail.0] scanning path /home/ubuntu/lab10/app.log
[2022/06/14 23:08:02] [debug] [input:tail:tail.0] inode=1039240 with offset=0 appended as
/home/ubuntu/lab10/app.log
[2022/06/14 23:08:02] [debug] [input:tail:tail.0] scan_glob add(): /home/ubuntu/lab10/app.log, inode
1039240
```

```
[2022/06/14 23:08:02] [debug] [input:tail:tail.0] 1 new files found on path
'/home/ubuntu/lab10/app.log'
[2022/06/14 23:08:02] [debug] [forward:forward.0] created event channels: read=29 write=30
[2022/06/14 23:08:02] [ info] [output:forward:forward.0] worker #0 started
[2022/06/14 23:08:02] [debug] [router] default match rule tail.0:forward.0
[2022/06/14 23:08:02] [ info] [output:forward:forward.0] worker #1 started
[2022/06/14 23:08:02] [ info] [sp] stream processor started
[2022/06/14 23:08:02] [debug] [input:tail:tail.0] inode=1039240 file=/home/ubuntu/lab10/app.log
promote to TAIL_EVENT
[2022/06/14 23:08:02] [ info] [input:tail:tail.0] inotify_fs_add(): inode=1039240 watch_fd=1
name=/home/ubuntu/lab10/app.log
[2022/06/14 23:08:02] [debug] [input:tail:tail.0] [static files] processed 0b, done
```

In a new or free terminal session, use `ps` to see all running instances of Fluentd and Fluent Bit:

```
ubuntu@labsys:~/lab10$ ps -ef | grep fluent

systemd+ 205222 205197 0 23:06 ?        00:00:00 tini -- /bin/entrypoint.sh fluentd -c
/fluentd/etc/fd-aggregator.conf
systemd+ 205234 205222 0 23:06 ?        00:00:00 /usr/bin/ruby /usr/bin/fluentd -c
/fluentd/etc/fd-aggregator.conf --plugin /fluentd/plugins
systemd+ 205243 205234 0 23:06 ?        00:00:00 /usr/bin/ruby -Eascii-8bit:ascii-8bit
/usr/bin/fluentd -c /fluentd/etc/fd-aggregator.conf --plugin /fluentd/plugins --under-supervisor
ubuntu 205257 204978 1 23:07 pts/0    00:00:01 /usr/bin/ruby2.7 /usr/local/bin/fluentd -c fd-
forwarder.conf
ubuntu 205262 205257 0 23:07 pts/0    00:00:00 /usr/bin/ruby2.7 -Eascii-8bit:ascii-8bit
/usr/local/bin/fluentd -c fd-forwarder.conf --under-supervisor
ubuntu 205270 201448 0 23:08 pts/1    00:00:00 fluent-bit -c fb-forwarder.conf -v
ubuntu 205399 205373 0 23:08 pts/2    00:00:00 grep --color=auto fluent

ubuntu@labsys:~/lab10$
```

Note the following:

- PID `205234` and `205243` are the log aggregator instance supervisor and worker processes
- The Fluentd forwarder processes are running on PIDs `205257` and `205262`
- Fluent-Bit is running on pid `205270`

Now use `top` to compare their resource footprints. Use `pgrep` to retrieve the pids for all processes run with `fluent` and `ruby`:

```
ubuntu@labsys:~/lab10$ top -p `pgrep -d "," fluent\|ruby`

top - 23:09:28 up 6:31, 3 users, load average: 0.04, 0.04, 0.01
Tasks: 5 total, 0 running, 5 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.2 us, 0.2 sy, 0.0 ni, 99.3 id, 0.0 wa, 0.0 hi, 0.0 si, 0.3 st
MiB Mem : 3865.9 total, 2011.0 free, 417.8 used, 1437.0 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used. 3187.0 avail Mem

   PID USER      PR  NI   VIRT   RES    SHR S  %CPU  %MEM    TIME+  COMMAND
 205243 systemd+  20   0 108548  39676  6596 S   0.3   1.0   0:00.77 ruby
 205270 ubuntu    20   0  87008  12860 10760 S   0.3   0.3   0:00.02 fluent-bit
 205234 systemd+  20   0 105712  38820  6620 S   0.0   1.0   0:00.80 fluentd
 205257 ubuntu    20   0 305320  47064  9624 S   0.0   1.2   0:01.32 fluentd
 205262 ubuntu    20   0 440412  49564  9260 S   0.0   1.3   0:00.78 ruby2.7

q

ubuntu@labsys:~/lab10$
```

Compared to the Fluentd log forwarder, Fluent Bit is consuming far fewer resources than either of its Fluentd cousins. It's easy to see that in a resource constrained environment, Fluent Bit would be the way to go.

3b. Forwarding Events to Fluentd with Fluent Bit

Now it's time to compare the resulting events forwarded from the same file between Fluentd and Fluent Bit.

In a free terminal, send an event to the app.log file that both log forwarders are tailing. Be sure to force the Fluentd log forwarder to flush its buffers.

```
ubuntu@labsys:~/lab10$ echo "This is an event" >> ~/lab10/app.log

ubuntu@labsys:~/lab10$ pkill -sigusr1 fluentd

pkill: killing pid 29564 failed: Operation not permitted

ubuntu@labsys:~/lab10$
```

In the Fluent Bit terminal, it should have recorded the event being captured:

```
...

[2022/06/14 23:09:59] [debug] [out flush] cb_destroy coro_id=0
[2022/06/14 23:09:59] [debug] [task] destroy task=0x7fab59a3d460 (task_id=0)
[2022/06/14 23:10:01] [debug] [input:tail:tail.0] scanning path /home/ubuntu/lab10/app.log
[2022/06/14 23:10:01] [debug] [input:tail:tail.0] scan_blog add(): dismissed:
/home/ubuntu/lab10/app.log, inode 1039240
[2022/06/14 23:10:01] [debug] [input:tail:tail.0] 0 new files found on path
'/home/ubuntu/lab10/app.log'
[2022/06/14 23:10:29] [debug] [upstream] drop keepalive connection #-1 to 127.0.0.1:24500 (keepalive
idle timeout)
```

Check the log aggregator output to see if it received events from its log forwarders:

```
ubuntu@labsys:~/lab10$ sudo docker logs aggregator1 --tail 5

2022-06-14 23:07:31.002443014 +0000 fluent.info: {"worker":0,"message":"fluentd worker is now
running worker=0"}
2022-06-14 23:09:58.921970203 +0000 tail.0: {"log":"This is an event"}
2022-06-14 23:09:58.922182768 +0000 fd.forward: {"message":"This is an event"}
2022-06-14 23:10:01.937402218 +0000 fluent.info: {"message":"force flushing buffered events"}
2022-06-14 23:10:01.937904760 +0000 fluent.info: {"message":"flushing all buffer forcedly"}

ubuntu@labsys:~/lab10$
```

The log aggregator received an event from both Fluentd (as seen from the `fd.forward` tagged event) and Fluent Bit (which sent the event under `tail.0`). Fluent Bit is now successfully forwarding events to Fluentd!

4. Complete Logging Pipeline with Apache, Fluent Bit, Fluentd, Elasticsearch and Kibana

Fluentd is as part of a full log collection and analysis stack with Elasticsearch, an open source search engine based on Lucene, and Kibana, an open source data visualization frontend for Elasticsearch. This combination is called an EFK stack. In this final step, you will set up a full EFK stack, with Fluent Bit acting as a Fluentd forwarder.

4a. Apache to Fluent Bit and Fluentd

To start setting up this EFK stack, the Apache Webserver will be used to feed Fluent Bit, Fluentd, and Elasticsearch with event data.

Install the Apache webserver:

```
ubuntu@labsys:~$ sudo apt install apache2 -y

...

ubuntu@labsys:~$
```

Relaunch the Fluent Bit instance from the previous step so it tails the Apache access log and forwards to Fluentd:

```
...
^C

[2022/06/14 23:11:49] [engine] caught signal (SIGINT)
[2022/06/14 23:11:49] [ info] [input] pausing tail.0
[2022/06/14 23:11:49] [ warn] [engine] service will shutdown in max 5 seconds
[2022/06/14 23:11:49] [ info] [engine] service has stopped (0 pending tasks)
[2022/06/14 23:11:49] [debug] [input:tail:tail.0] inode=1039240 removing file name
/home/ubuntu/lab10/app.log
[2022/06/14 23:11:49] [ info] [input:tail:tail.0] inotify_fs_remove(): inode=1039240 watch_fd=1
[2022/06/14 23:11:49] [ info] [output:forward:forward.0] thread worker #0 stopping...
[2022/06/14 23:11:49] [ info] [output:forward:forward.0] thread worker #0 stopped
[2022/06/14 23:11:49] [ info] [output:forward:forward.0] thread worker #1 stopping...
[2022/06/14 23:11:49] [ info] [output:forward:forward.0] thread worker #1 stopped

ubuntu@labsys:~/lab10$

ubuntu@labsys:~/lab10$ fluent-bit -i tail -p path=/var/log/apache2/access.log -o
forward://127.0.0.1:24500 -v

* Copyright (C) 2015-2022 The Fluent Bit Authors
* Fluent Bit is a CNCF sub-project under the umbrella of Fluentd
* https://fluentbit.io

[2022/06/14 23:11:59] [ info] Configuration:
[2022/06/14 23:11:59] [ info]   flush time      | 1.000000 seconds
[2022/06/14 23:11:59] [ info]   grace         | 5 seconds
[2022/06/14 23:11:59] [ info]   daemon        | 0
[2022/06/14 23:11:59] [ info]   _____
[2022/06/14 23:11:59] [ info]   inputs:
[2022/06/14 23:11:59] [ info]     tail
[2022/06/14 23:11:59] [ info]   _____
[2022/06/14 23:11:59] [ info]   filters:
[2022/06/14 23:11:59] [ info]   _____
[2022/06/14 23:11:59] [ info]   outputs:
[2022/06/14 23:11:59] [ info]     forward.0
[2022/06/14 23:11:59] [ info]   _____
[2022/06/14 23:11:59] [ info]   collectors:
[2022/06/14 23:11:59] [ info] [fluent bit] version=1.9.4, commit=, pid=206431
[2022/06/14 23:11:59] [debug] [engine] coroutine stack size: 24576 bytes (24.0K)
[2022/06/14 23:11:59] [ info] [storage] version=1.2.0, type=memory-only, sync=normal,
checksum=disabled, max_chunks_up=128
[2022/06/14 23:11:59] [ info] [cmetrics] version=0.3.1
[2022/06/14 23:11:59] [debug] [tail:tail.0] created event channels: read=21 write=22
[2022/06/14 23:11:59] [debug] [input:tail:tail.0] flb_tail_fs_inotify_init() initializing inotify
```

```

tail input
[2022/06/14 23:11:59] [debug] [input:tail:tail.0] inotify watch fd=27
[2022/06/14 23:11:59] [debug] [input:tail:tail.0] scanning path /var/log/apache2/access.log
[2022/06/14 23:11:59] [debug] [input:tail:tail.0] inode=1802060 with offset=0 appended as
/var/log/apache2/access.log
[2022/06/14 23:11:59] [debug] [input:tail:tail.0] scan_glob add(): /var/log/apache2/access.log,
inode 1802060
[2022/06/14 23:11:59] [debug] [input:tail:tail.0] 1 new files found on path
'/var/log/apache2/access.log'
[2022/06/14 23:11:59] [debug] [forward:forward.0] created event channels: read=29 write=30
[2022/06/14 23:11:59] [debug] [router] default match rule tail.0:forward.0
[2022/06/14 23:11:59] [ info] [sp] stream processor started
[2022/06/14 23:11:59] [debug] [input:tail:tail.0] inode=1802060 file=/var/log/apache2/access.log
promote to TAIL_EVENT
[2022/06/14 23:11:59] [ info] [input:tail:tail.0] inotify_fs_add(): inode=1802060 watch_fd=1
name=/var/log/apache2/access.log
[2022/06/14 23:11:59] [debug] [input:tail:tail.0] [static files] processed 0b, done
[2022/06/14 23:11:59] [ info] [output:forward:forward.0] worker #0 started
[2022/06/14 23:11:59] [ info] [output:forward:forward.0] worker #1 started

```

In a free terminal, modify the Fluentd log aggregator configuration that parses the incoming log message with a filter using the Apache2 parser:

```

ubuntu@labsys:~$ nano ~/lab10/fd-aggregator.conf && cat $_

<source>
  @type forward
  port 24500
</source>

<filter tail.0>
  @type parser
  key_name log
  <parse>
    @type apache2
  </parse>
</filter>

<match tail.0>
  @type stdout
</match>

ubuntu@labsys:~/lab10$

```

As observed from the previous step, Fluent Bit is forwarding events under the `tail.0` tag, so configure both the `<filter>` and `<match>` directives to capture events tagged `tail.0`.

Restart the log aggregator Fluentd instance to reload the configuration and check the log aggregator's output to ensure it restarted correctly:

```

ubuntu@labsys:~/lab10$ sudo docker restart aggregator1

aggregator1

ubuntu@labsys:~/lab10$ sudo docker logs aggregator1 --tail 5

2022-06-14 23:13:08 +0000 [info]: adding match pattern="tail.0" type="stdout"
2022-06-14 23:13:08 +0000 [info]: adding source type="forward"

```

```
2022-06-14 23:13:08 +0000 [info]: #0 starting fluentd worker pid=16 ppid=7 worker=0
2022-06-14 23:13:08 +0000 [info]: #0 listening port port=24500 bind="0.0.0.0"
2022-06-14 23:13:08 +0000 [info]: #0 fluentd worker is now running worker=0

ubuntu@labsys:~/lab10$
```

The log aggregator is now capturing all events tagged `tail.0` with both a `<filter>` and `<match>` directive. Your Fluent Bit and Fluentd stack is now ready to receive events.

Send a `curl` request to localhost to write an event to Apache:

```
ubuntu@labsys:~$ curl localhost

...

ubuntu@labsys:~/lab10$ sudo docker logs aggregator1 --tail 1

2022-06-14 23:13:22.000000000 +0000 tail.0:
{"host":"127.0.0.1","user":null,"method":"GET","path":"/","code":200,"size":11173,"referer":null,"agent":"curl/7.68.0"}

ubuntu@labsys:~/lab10$
```

Nice! Fluent Bit successfully submitted an Apache access event to Fluentd.

So far, the pipeline consists of a Fluent Bit log forwarder tailing the Apache access log, which forwards to a Fluentd log aggregator that performs the parsing. Fluent Bit is submitting those events to Fluentd under the `tail.0` tag.

Now that Fluentd is able to receive events from Fluent Bit, it's time to establish a more meaningful (and common) destination for Fluentd events: Elasticsearch.

Run an Elasticsearch and Kibana container, using `--network host` to ensure those containers can communicate with the Fluentd log aggregator.

N.B. Due to the complexity of setup introduced by Elasticsearch 8, this lab will run Elasticsearch 7. If you want more details on setting up a Fluentd-powered log pipeline with Elasticsearch 8, see lab 4 of this course.

```
ubuntu@labsys:~$ sudo docker run -d --name elasticsearch -p 9200:9200 -e "discovery.type=single-node" docker.elastic.co/elasticsearch/elasticsearch:7.17.4

...

22afa36eaa1235ffcf85f096b5f1869db383a3a7a682fe2d3d38832ef5537c33

ubuntu@labsys:~$ sudo docker run -d --name kibana --net host -e
ELASTICSEARCH_HOSTS=http://localhost:9200 docker.elastic.co/kibana/kibana:7.17.4

...

0c842526eff763d713e6ffc3b1b4455d69dec4f74d19422db8fd06a6129b3201

ubuntu@labsys:~$
```

Now it's time to prepare the Fluentd log aggregator to submit events to Elasticsearch.

Gem installations of Fluentd do not install the Elasticsearch plugin by default, so you will need to install the Fluentd Elasticsearch Plugin in the log aggregator container.

To do this cleanly, create a Dockerfile that will install the Elasticsearch plugin:

```
ubuntu@labsys:~/lab10$ nano Dockerfile && cat $_  
  
FROM fluent/fluentd:v1.14-1  
USER root  
RUN ["fluent-gem", "install", "elasticsearch", "-N", "-v", "7.17.1"]  
RUN ["fluent-gem", "install", "elasticsearch-transport", "-N", "-v", "7.17.1"]  
RUN ["fluent-gem", "install", "fluent-plugin-elasticsearch", "-N", "-v", "5.2.2"]  
USER fluent  
  
ubuntu@labsys:~/lab10$
```

Now build the container, tagging it `lfs242/fluent` with the tag `elasticsearch` :

```
ubuntu@labsys:~/lab10$ sudo docker build -t lfs242/fluentd:elasticsearch .  
  
Sending build context to Docker daemon 19.46kB  
Step 1/6 : FROM fluent/fluentd:v1.14-1  
--> 396489e40ea6  
Step 2/6 : USER root  
--> Using cache  
--> a03e2f8b9ada  
Step 3/6 : RUN ["fluent-gem", "install", "elasticsearch", "-N", "-v", "7.17.1"]  
--> Running in 8d2963b49921  
Successfully installed multi_json-1.15.0  
Successfully installed faraday-em_http-1.0.0  
Successfully installed faraday-em_synchrony-1.0.0  
Successfully installed faraday-excon-1.1.0  
Successfully installed faraday-httpclient-1.0.1  
Successfully installed multipart-post-2.2.3  
Successfully installed faraday-multipart-1.0.4  
Successfully installed faraday-net_http-1.0.1  
Successfully installed faraday-net_http_persistent-1.2.0  
Successfully installed faraday-patron-1.0.0  
Successfully installed faraday-rack-1.0.0  
Successfully installed faraday-retry-1.0.3  
Successfully installed ruby2_keywords-0.0.5  
Successfully installed faraday-1.10.0  
Successfully installed elasticsearch-transport-7.17.1  
Successfully installed elasticsearch-api-7.17.1  
Successfully installed elasticsearch-7.17.1  
17 gems installed  
Removing intermediate container 8d2963b49921  
--> f8b9aae39822  
Step 4/6 : RUN ["fluent-gem", "install", "elasticsearch-transport", "-N", "-v", "7.17.1"]  
--> Running in e62b7e0ca648  
Successfully installed elasticsearch-transport-7.17.1  
1 gem installed  
Removing intermediate container e62b7e0ca648  
--> 3a6b3c45169c  
Step 5/6 : RUN ["fluent-gem", "install", "fluent-plugin-elasticsearch", "-N", "-v", "5.2.2"]  
--> Running in af1d2b2845ba  
Successfully installed excon-0.92.3  
Successfully installed fluent-plugin-elasticsearch-5.2.2  
2 gems installed  
Removing intermediate container af1d2b2845ba  
--> f9341a8fd0da  
Step 6/6 : USER fluent
```

```
---> Running in d31767fc6803
Removing intermediate container d31767fc6803
---> 7e51a29a099f
Successfully built 7e51a29a099f
Successfully tagged lfs242/fluentd:elasticsearch

ubuntu@labsys:~/lab10$
```

Now configure the log aggregator to output events to Elasticsearch:

```
ubuntu@labsys:~$ nano ~/lab10/fd-aggregator.conf && cat $_

<source>
  @type forward
  port 24500
</source>

<filter tail.0>
  @type parser
  key_name log
  <parse>
    @type apache2
  </parse>
</filter>

<match tail.0>
  @type elasticsearch
  host localhost
  port 9200
  logstash_format true
  include_timestamp true
</match>

ubuntu@labsys:~$
```

Restart the log aggregator container to reload the configuration:

```
ubuntu@labsys:~/lab10$ sudo docker container rm $(sudo docker container stop aggregator1)

aggregator1

ubuntu@labsys:~/lab10$ sudo docker run -d --network host --name aggregator1 -v $HOME/lab10/fd-
aggregator.conf:/fluentd/etc/fd-aggregator.conf -e FLUENTD_CONF=fd-aggregator.conf
lfs242/fluentd:elasticsearch -c /fluentd/etc/fd-aggregator.conf

4789ca4b0d038d1252c0c6765f10b68987d8ecb68cd32bfe019ae1bc466f5853

ubuntu@labsys:~/lab10$ sudo docker logs aggregator1 --tail 5

2022-06-14 23:36:20 +0000 [warn]: #0 Detected ES 7.x: `_doc` will be used as the document `_type`.
2022-06-14 23:36:20 +0000 [info]: adding source type="forward"
2022-06-14 23:36:20 +0000 [info]: #0 starting fluentd worker pid=16 ppid=7 worker=0
2022-06-14 23:36:20 +0000 [info]: #0 listening port port=24500 bind="0.0.0.0"
2022-06-14 23:36:20 +0000 [info]: #0 fluentd worker is now running worker=0

ubuntu@labsys:~/lab10$
```

Curl Apache to generate another event:

```
ubuntu@labsys:~$ curl localhost
```

```
...
```

```
ubuntu@labsys:~$
```

Flush the Fluentd Buffer to send the event to Elasticsearch:

```
ubuntu@labsys:~/lab10$ sudo docker kill --signal=SIGUSR1 aggregator1
```

```
aggregator1
```

```
ubuntu@labsys:~/lab10$ sudo docker logs aggregator1 --tail 5
```

```
2022-06-14 23:36:20 +0000 [info]: #0 starting fluentd worker pid=16 ppid=7 worker=0
2022-06-14 23:36:20 +0000 [info]: #0 listening port port=24500 bind="0.0.0.0"
2022-06-14 23:36:20 +0000 [info]: #0 fluentd worker is now running worker=0
2022-06-14 23:36:41 +0000 [info]: #0 force flushing buffered events
2022-06-14 23:36:41 +0000 [info]: #0 flushing all buffer forcedly
```

```
ubuntu@labsys:~/lab10$
```

Now check the Elasticsearch indices to see if any data has been written:

```
ubuntu@labsys:~$ curl 'localhost:9200/_cat/indices?v'
```

health	status	index	uuid	pri	rep	docs.count	docs.deleted
green	open	.geoip_databases	8a3cCTHoTyqFSw2T0ak-2Q	1	0	41	0
38.9mb		38.9mb					
green	open	.apm-custom-link	qlaRKbhdQpuNxtB3euRbcQ	1	0	0	0
226b		226b					
green	open	.apm-agent-configuration	LbGiQD_9TQOasmU7WYMAIw	1	0	0	0
226b		226b					
yellow	open	logstash-2022.06.14	ORiTznsQR3eNmXLgGM_cPA	1	1	1	0
6.3kb		6.3kb					
green	open	.kibana_task_manager_7.17.4_001	KLvhu06mTv-1MC7u5t12Rg	1	0	17	183
156.3kb		156.3kb					
green	open	.kibana_7.17.4_001	nOfw9d6wRJiTYHh0Rr7lMw	1	0	11	0
2.3mb		2.3mb					

```
ubuntu@labsys:~$
```

Fluentd has been configured to submit its events to Elasticsearch like Logstash (another log processing solution also made by Elastic), so it is forwarding events to Elasticsearch to a **logstash-** prefixed index.

Search the **logstash-** index for data. Elasticsearch should return as many results as you have sent to Apache so far:

```
ubuntu@labsys:~$ curl -s localhost:9200/logstash-*/_search?pretty
```

```
{
  "took" : 1,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  }
}
```

```

},
"hits" : {
  "total" : {
    "value" : 1,
    "relation" : "eq"
  },
  "max_score" : 1.0,
  "hits" : [
    {
      "_index" : "logstash-2022.06.14",
      "_type" : "_doc",
      "_id" : "MReTZIEB0ikbxoe_kbLg",
      "_score" : 1.0,
      "_source" : {
        "host" : "127.0.0.1",
        "user" : null,
        "method" : "GET",
        "path" : "/",
        "code" : 200,
        "size" : 11173,
        "referer" : null,
        "agent" : "curl/7.68.0",
        "@timestamp" : "2022-06-14T23:36:38.000000000+00:00"
      }
    }
  ]
}
}
}

ubuntu@labsys:~$

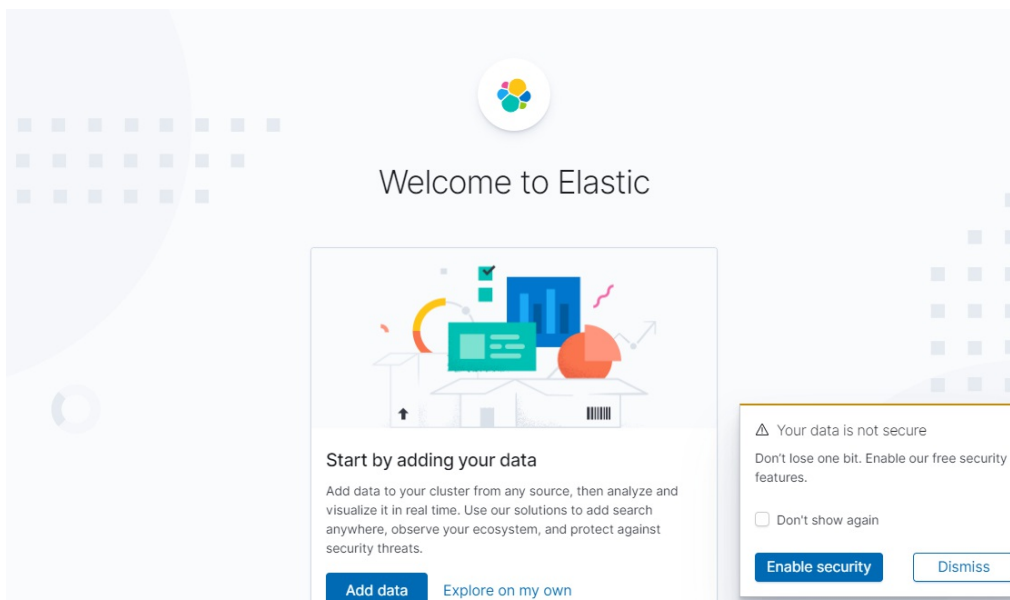
```

An Apache event has been sent into Elasticsearch by Fluent Bit and Fluentd!

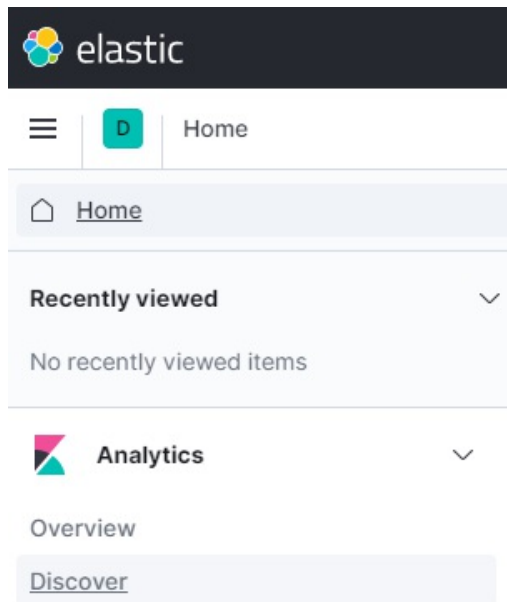
Now that Apache is sending logs to Elasticsearch with Fluentd, you can use Kibana to perform data analysis. One of the easiest ways to do this is by using Kibana's data visualizations to quickly discern patterns over time. In order to use data from events in visualizations, an index pattern needs to be created to tell Kibana which Elasticsearch indices it needs to pull data from.

Using a browser, navigate to your Kibana instance at your VM's IP at port **5601**.

When the welcome screen comes up choose "Explore on my own" and dismiss the security window.



In the Side Menu, click "Discover"



In the "Create index pattern" section, enter `logstash*` in the "Step 1 of 2: Define index pattern" input field.

Create index pattern

Name

Use an asterisk (*) to match multiple characters. Spaces and the characters `/`, `?`, `"`, `<`, `>`, `|` are not allowed.

Timestamp field

Select a timestamp field for use with the global time filter.

[Show advanced settings](#)

✓ Your index pattern matches 1 source.

logstash-2022.06.14

Index

Rows per page: 10

You should see the "Your index pattern matches 1 source." message and the Films index selected.

Choose the "@timestamp" option from the "Timestamp field" dropdown.

Create index pattern

Name

Use an asterisk (*) to match multiple characters. Spaces and the characters `/`, `?`, `"`, `<`, `>`, `|` are not allowed.

Timestamp field

Select a timestamp field for use with the global time filter.

[Show advanced settings](#)

✓ Your index pattern matches 1 source.

logstash-2022.06.14

Index

Rows per page: 10













Now click "Create index pattern".

This should bring you to a dialog that allows you to explore the Logstash (Fluentd) index structure.

Time field: '@timestamp'

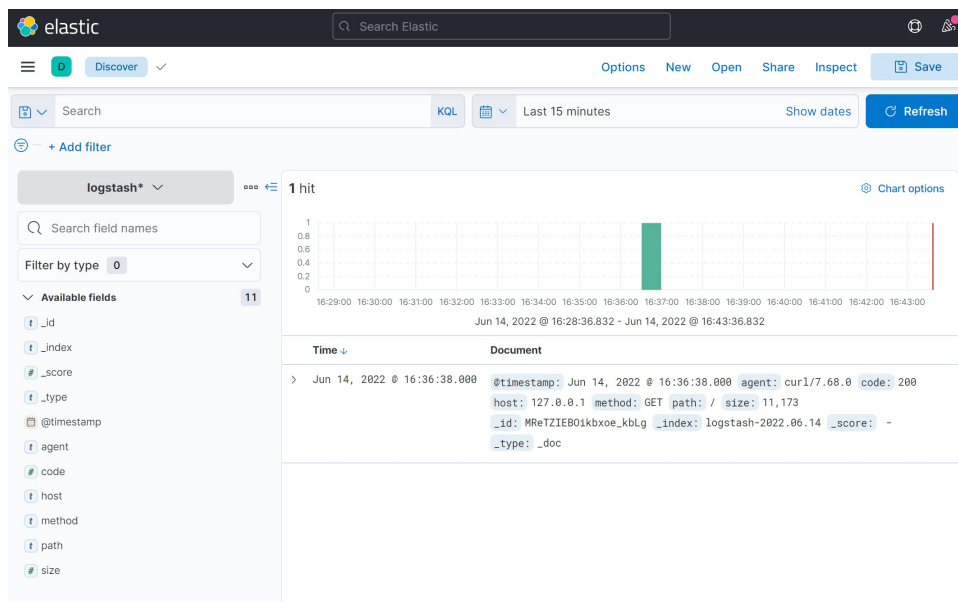
View and edit fields in **logstash***. Field attributes, such as type and searchability, are based on [field mappings](#) in Elasticsearch.

Fields (16) **Scripted fields (0)** **Field filters (0)**




<input type="text" value="Search"/> All field types Add field					
Name ↑	Type	Format	Searchable	Aggregatable	Excluded
@timestamp 	date				
_id	_id				
_index	_index				
_score					
_source	_source				


The main reason why Fluentd was configured to utilize the logstash format when sending events to Elasticsearch is that if it had been configured more simply without the logstash format, then there would be no time filters available to sort data with.


If you go to Discover again, you should now be able to see the logstash index. If not, adjust the time interval. That amount of events depends on how many `curl` requests you've sent to Apache:






If you would like to create your own visualizations, go to **Visualize Library** and select **"Create a visualization"**:

  Discover 

 Home

Recently viewed 
No recently viewed items

 **Analytics** 
Overview
Discover
Dashboard
Canvas
Maps
Machine Learning
Visualize Library

 Visualize Library

 Building a dashboard? Create content directly from the [Dashboard application](#) using a new integrated workflow.



Create your first visualization

You can create different visualizations based on your data.

 Create new visualization

Pick "Lens":

New visualization



Lens

Create visualizations with our drag and drop editor. Switch between visualization types at any time. *Recommended for most users.*



Maps

Create and style maps with multiple layers and indices.



TSVB

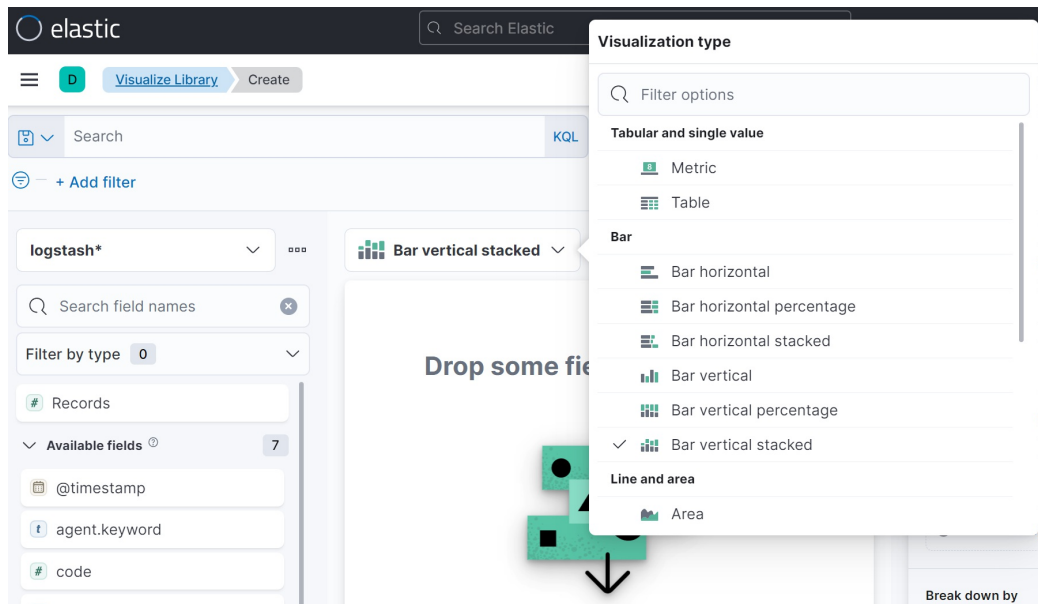
Perform advanced analysis of your time series data.



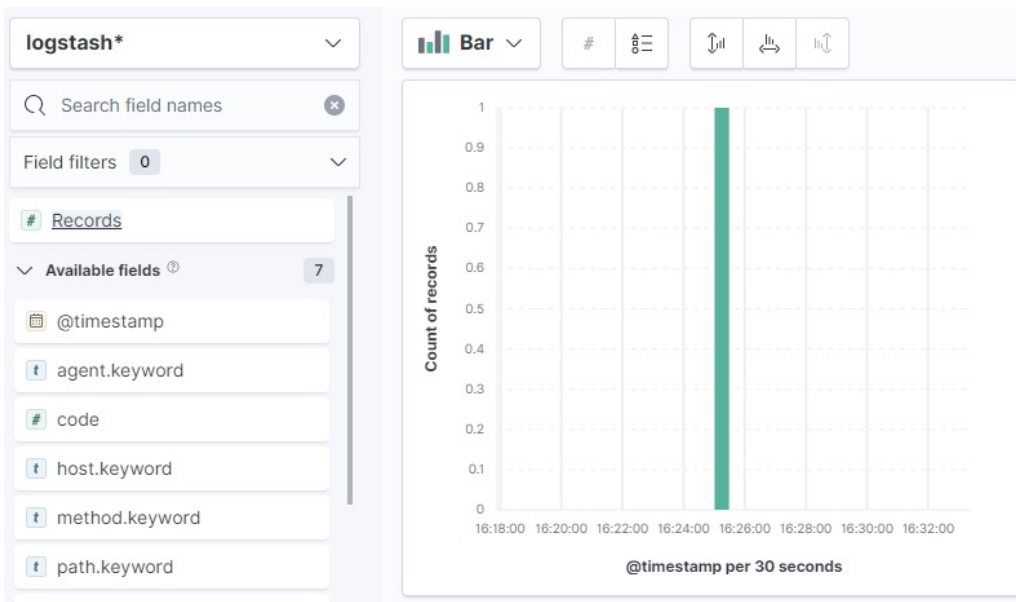
Custom visualization

Use Vega to create new types of visualizations. *Requires knowledge of Vega syntax.*

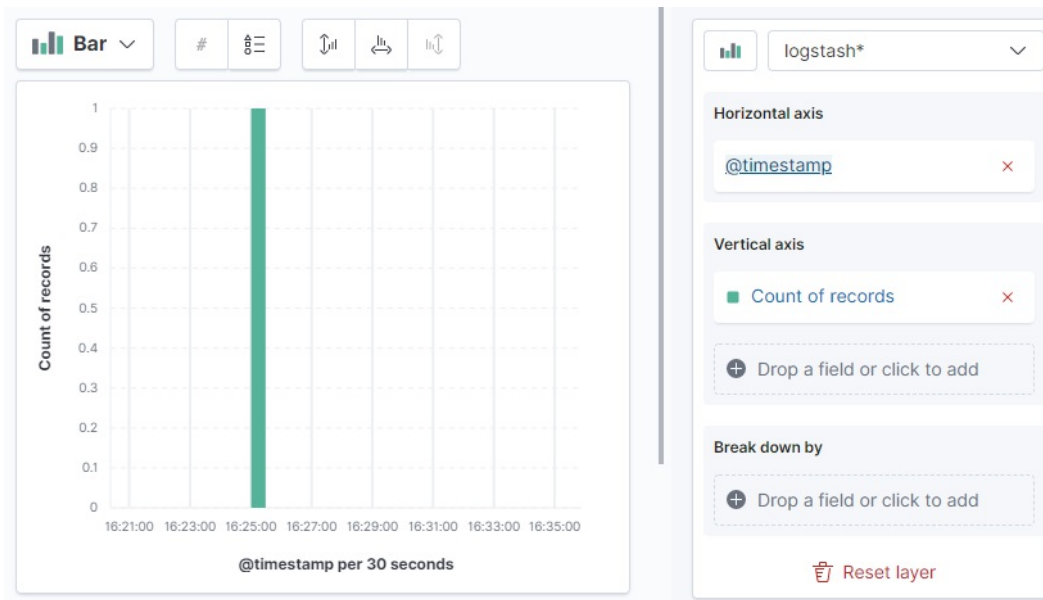
Select "Bar vertical stacked":



For the source, make sure **logstash*** is selected and drag the **Records** field to the center:



Under On the right side pane, select `@timestamp` under **Horizontal axis**:



Check "Customize time interval" and set the minimum interval to `1` and pick `minutes` from the dropdown.

← Horizontal axis configuration

Select a function

Date histogram

Filters

Intervals

Top values

Select a field

@timestamp

How it works

☒ Customize time interval

Minimum interval

1

minutes

Display name

@timestamp

Click **X Close** to dismiss the Horizontal Axis settings pane. Your graph will update accordingly:



Submitting additional curl requests to your Apache webserver will cause this graph to grow.

Click Save and name your visualization so you can access it later.

Excellent! With this lab, you have now successfully established a basic Elasticsearch, Fluentd (fed by Fluent Bit!) and Kibana logging stack.

5. Clean up

When you have finished exploring, follow these steps to tear down all containers, Fluent Bit and Fluentd instances.

Use **pkill** to shut down all Fluentd and Fluent Bit instances:

```
ubuntu@labsys:~$ pkill -f fluent
ubuntu@labsys:~$
```

Fluent Bit should shut down:

```
...
[2022/06/14 23:47:36] [engine] caught signal (SIGTERM)
[2022/06/14 23:47:36] [ info] [input] pausing tail.0
[2022/06/14 23:47:36] [ warn] [engine] service will shutdown in max 5 seconds
[2022/06/14 23:47:36] [ info] [engine] service has stopped (0 pending tasks)
[2022/06/14 23:47:36] [debug] [input:tail:tail.0] inode=1802060 removing file name
/var/log/apache2/access.log
[2022/06/14 23:47:36] [ info] [input:tail:tail.0] inotify_fs_remove(): inode=1802060 watch_fd=1
[2022/06/14 23:47:36] [ info] [output:forward:forward.0] thread worker #0 stopping...
[2022/06/14 23:47:36] [ info] [output:forward:forward.0] thread worker #0 stopped
[2022/06/14 23:47:36] [ info] [output:forward:forward.0] thread worker #1 stopping...
[2022/06/14 23:47:36] [ info] [output:forward:forward.0] thread worker #1 stopped

ubuntu@labsys:~/lab10$
```

The Fluentd forwarder too:

```
...
2022-06-14 23:47:36 +0000 [info]: Received graceful stop
2022-06-14 23:47:36 +0000 [info]: #0 fluentd worker is now stopping worker=0
2022-06-14 23:47:36 +0000 [info]: #0 shutting down fluentd worker worker=0
2022-06-14 23:47:36 +0000 [info]: #0 shutting down input plugin type=:tail plugin_id="object:758"
2022-06-14 23:47:37 +0000 [info]: #0 shutting down output plugin type=:forward
plugin_id="object:730"
2022-06-14 23:47:38 +0000 [info]: Worker 0 finished with status 0

ubuntu@labsys:~/lab10$
```

Finally, use `docker container rm` to shut down all containers:

```
ubuntu@labsys:~$ sudo docker container rm $(sudo docker container stop aggregator1 elasticsearch kibana)

aggregator1
elasticsearch
kibana

ubuntu@labsys:~$
```

Congratulations, you have completed the Lab.