LFS242 - Cloud Native Logging with Fluentd

Lab 7 – Using Multiple Fluentd Instances, Configuring High Availability and Testing Failover in Fluentd

Fluentd is capable of supporting multiple instances using the in_forward and out_forward plugins, allowing many Fluentd instances to communicate with each other. This allows many smaller instances of Fluentd to send logs to a centralized log aggregator for further processing. In addition to forwarding and aggregation, additional instances of Fluentd can be employed to make the logging pipeline more robust.

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

- Stand up a logging pipeline consisting of a log forwarder and log aggregator
- Explore the failover behavior of a multi-Fluentd instance deployment
- Learn how to set up a Highly Available Fluentd deployment

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 && cat $_

#!/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 additional Fluentd instances, 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. Setting Up a log forwarder & log aggregator

One of the simplest cases of using multiple instances of Fluentd is in a forwarder-aggregator setup. The log forwarder instance is intended to be deployed near to or on the same node as the application it is collecting logs from, and is typically light weight in nature. Processing is shifted to a log aggregator instance, which listens for traffic from log forwarders. Since log aggregators will typically perform the processing, they tend to be larger instances and may even be deployed on nodes provisioned just for Fluentd.

To begin, create a log aggregator Fluentd configuration:

```
ubuntu@labsys:~$ mkdir ~/lab7 && cd $_

ubuntu@labsys:~/lab7$ mkdir aggregator1 && cd $_

ubuntu@labsys:~/lab7/aggregator1$ nano aggregator1.conf && cat $_

<system>
  process_name aggregator1
</system>

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

<match>
  @type stdout
</match>

ubuntu@labsys:~/lab7/aggregator1$
```

The log aggregator in this example will resemble the Fluentd instances set up in previous labs. Log aggregators will host the majority (if not all) of the processing power within a multi-Fluentd deployment, so this is where the more complex `<filter>` and `<match>` directives will be hosted.

Run an instance of this log aggregator with Docker, making sure to mount the aggregator1 directory to make your configuration file available to the container:

```
ubuntu@labsys:~/lab7/aggregator1$ sudo docker run -p 24500:24500 -d \
--name aggregator1 \
-v $HOME/lab7/aggregator1:/fluentd/etc \
-e FLUENTD_CONF=aggregator*.conf fluent/fluentd:v1.14.6-1.1

...

0d7a050297f1d8109c8f3f82c697554a650828e40ceee32bd7c655073b79ba23

ubuntu@labsys:~/lab7/aggregator1$ sudo docker logs aggregator1 --tail 5

2022-06-20 18:16:25 +0000 [info]: #0 listening port port=24500 bind="0.0.0.0"
2022-06-20 18:16:25 +0000 [info]: #0 fluentd worker is now running worker=0
2022-06-20 18:16:25.283589771 +0000 fluent.info: {"pid":16,"ppid":7,"worker":0,"message":"starting
fluentd worker pid=16 ppid=7 worker=0"}
2022-06-20 18:16:25.283930039 +0000 fluent.info: {"port":24500,"bind":"0.0.0.0","message":"listening
port port=24500 bind=\"0.0.0.0\""}
2022-06-20 18:16:25.284440933 +0000 fluent.info: {"worker":0,"message":"fluentd worker is now
running worker=0"}

ubuntu@labsys:~/lab7/aggregator1$
```

Since this log aggregator is set up to output to STDOUT, run it without the `-d` flag to keep an eye on its output. This terminal will be locked.

- Feel free to set up the aggregator to output to a file or other destination
- Add additional filters for this aggregator
- To make changes, edit the aggregator1.conf and send the SIGHUP using `docker kill --signal=HUP aggregator1`

Once the aggregator is set up, it is ready to receive events.

Fluentd instances communicate with each other through the use of the `forward` input and output plugins over the network. In order for a forwarder-aggregator Fluentd setup to function correctly, all Fluentd instances must be able to communicate with each other. To start, the IP address is needed to send traffic to this log aggregator.

Open a new terminal session so that you can continue working. In your working terminal, retrieve the IP address of the aggregator container:

```
ubuntu@labsys:~/lab7/aggregator1$ cd ~/lab7

ubuntu@labsys:~/lab7$ sudo docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}
{{end}}' aggregator1

172.17.0.2

ubuntu@labsys:~/lab7$
```

The next piece in a multi-instance Fluentd deployment is a log forwarder. It is important for this instance to be as light weight as possible - typical log forwarders should have few, or no, filters configured to avoid taking resources from their primary source program.

It's time to set up a forwarder. In another terminal, create a simple configuration:

```
ubuntu@labsys:~$ mkdir ~/lab7/forwarder && cd $_

ubuntu@labsys:~/lab7/forwarder$ nano forwarder.conf && cat $_

<system>
  process_name forwarder
</system>

<source>
  @type forward
  port 32767
</source>

<match>
  @type forward
  <server>
    name aggregator1
    host 172.17.0.2
    port 24500
  </server>
</match>

ubuntu@labsys:~/lab7/forwarder$
```

The heart of the log forwarder Fluentd instance is the out_forward plugin. This plugin allows one or more servers to be specified as event destinations - the destination servers must be reachable by a hostname, FQDN, or IP address and be configured with an active in_forward `<source>` directive.

To target the docker aggregator1, the IP address of the container, and the forward port was included.

A local installation of Fluentd will be used to run the log forwarder:

```
ubuntu@labsys:~/lab7/forwarder$ fluentd -c forwarder.conf

2022-06-20 18:17:49 +0000 [info]: parsing config file is succeeded path="forwarder.conf"
2022-06-20 18:17:49 +0000 [info]: gem 'fluentd' version '1.14.6'
2022-06-20 18:17:49 +0000 [info]: adding forwarding server 'aggregator1' host="172.17.0.2"
port=24500 weight=60 plugin_id="object:730"
2022-06-20 18:17:49 +0000 [warn]: define <match fluent.**> to capture fluentd logs in top level is
deprecated. Use <label @FLUENT_LOG> instead
2022-06-20 18:17:49 +0000 [info]: using configuration file: <ROOT>
  <system>
    process_name "forwarder"
  </system>
  <source>
    @type forward
    port 32767
  </source>
  <match>
    @type forward
    <server>
      name "aggregator1"
      host "172.17.0.2"
      port 24500
    </server>
  </match>
</ROOT>
2022-06-20 18:17:49 +0000 [info]: starting fluentd-1.14.6 pid=51762 ruby="2.7.0"
2022-06-20 18:17:49 +0000 [info]: spawn command to main:  cmdline=["/usr/bin/ruby2.7", "-Eascii-
8bit:ascii-8bit", "/usr/local/bin/fluentd", "-c", "forwarder.conf", "--under-supervisor"]
2022-06-20 18:17:49 +0000 [info]: adding match pattern="**" type="forward"
2022-06-20 18:17:49 +0000 [info]: #0 adding forwarding server 'aggregator1' host="172.17.0.2"
port=24500 weight=60 plugin_id="object:730"
2022-06-20 18:17:49 +0000 [info]: adding source type="forward"
2022-06-20 18:17:49 +0000 [warn]: #0 define <match fluent.**> to capture fluentd logs in top level
is deprecated. Use <label @FLUENT_LOG> instead
2022-06-20 18:17:49 +0000 [info]: #0 starting fluentd worker pid=51767 ppid=51762 worker=0
2022-06-20 18:17:49 +0000 [info]: #0 listening port port=32767 bind="0.0.0.0"
2022-06-20 18:17:49 +0000 [info]: #0 fluentd worker is now running worker=0
```

This will also lock this terminal, so open another one to continue working.

If the correct IP (or hostname) and port were provided, the log forwarder instance should confirm that it found the log aggregator instance:

```
2022-06-20 18:17:49 +0000 [info]: #0 adding forwarding server 'aggregator1' host="172.17.0.2"
port=24500 weight=60 plugin_id="object:730"
```

There is an additional setting in the output that was not included in the original setup - weight. The weight of a log forwarder determines the distribution for load balancing. This functionality will be explored later in this lab.

Now that both a log aggregator and log forwarder have been configured, it’s time to test it.

In your working terminal, send a basic message to the log forwarder:

```
ubuntu@labsys:~/lab7$ echo '{"json":"message"}' | fluent-cat aggregate.me --port 32767
```

```
ubuntu@labsys:~/lab7$
```

In the forwarder terminal, see if there was any activity:

```
...

2022-06-20 18:17:49 +0000 [info]: #0 listening port port=32767 bind="0.0.0.0"
2022-06-20 18:17:49 +0000 [info]: #0 fluentd worker is now running worker=0
```

There was none recorded. Very light weight forwarders that only have a `<match>` directive will not have any additional outputs to STDOUT (unless configured for a `<filter>` directive using the filter_stdout plugin).

Check the aggregator1 terminal:

```
ubuntu@labsys:~/lab7$ sudo docker logs aggregator1 --tail 5

2022-06-20 18:16:25.284440933 +0000 fluent.info: {"worker":0,"message":"fluentd worker is now
running worker=0"}
2022-06-20 18:17:49.965756189 +0000 fluent.info:
{"pid":51767,"ppid":51762,"worker":0,"message":"starting fluentd worker pid=51767 ppid=51762
worker=0"}
2022-06-20 18:17:49.966932466 +0000 fluent.info: {"port":32767,"bind":"0.0.0.0","message":"listening
port port=32767 bind=\"0.0.0.0\""}
2022-06-20 18:17:49.967525910 +0000 fluent.info: {"worker":0,"message":"fluentd worker is now
running worker=0"}
2022-06-20 18:18:43.672084950 +0000 aggregate.me: {"json":"message"}

ubuntu@labsys:~/lab7$
```

After a short wait, the event will display in the `aggregator1` log output. When a `<match>` directive with the forward plugin is set up, events are sent directly to the server or servers that are configured.

Notice that there was a delay. The `out_forward` plugin is a buffered plugin, so it stored the events into a temporary buffer in disk or memory before sending them to their destination. This can be adjusted using `<buffer>` subdirectives under the forward match configuration.

2. Failover Scenario: Aggregator loss

Earlier in the lab, it was mentioned that in order for a forwarder-aggregator architecture to work correctly, both instances must be able to communicate with each other. What happens if the aggregator goes down?

Use `docker stop` on the aggregator1 terminal to stop the container:

```
ubuntu@labsys:~/lab7$ sudo docker stop aggregator1

aggregator1

ubuntu@labsys:~/lab7$
```

This container will remain on your system, so later in the lab you will only need to use `docker start` to get it running again.

However, the forwarder no longer has a destination to send its events to.

Try to send some events to it in your working terminal, flushing the buffer after sending it:

```
ubuntu@labsys:~/lab7$ echo '{"alive":"yes"}' | fluent-cat aggregate.me --port 32767

ubuntu@labsys:~/lab7$ pkill -SIGUSR1 fluentd
```

After some time, the forwarder will realize that its aggregator is down:

```
...

2022-06-20 18:21:05 +0000 [info]: #0 force flushing buffered events
2022-06-20 18:21:05 +0000 [info]: #0 flushing all buffer forcedly
2022-06-20 18:21:22 +0000 [warn]: #0 failed to flush the buffer. retry_times=0 next_retry_time=2022-
06-20 18:21:24 +0000 chunk="5e1e52a8da14c865466db311acfb6f0f" error_class=Errno::EHOSTUNREACH
error="No route to host - connect(2) for \"172.17.0.2\" port 24500"
  2022-06-20 18:21:22 +0000 [warn]: #0 /var/lib/gems/2.7.0/gems/fluentd-
1.14.6/lib/fluent/plugin_helper/socket.rb:62:in `initialize'
  2022-06-20 18:21:22 +0000 [warn]: #0 /var/lib/gems/2.7.0/gems/fluentd-
1.14.6/lib/fluent/plugin_helper/socket.rb:62:in `new'
  2022-06-20 18:21:22 +0000 [warn]: #0 /var/lib/gems/2.7.0/gems/fluentd-
1.14.6/lib/fluent/plugin_helper/socket.rb:62:in `socket_create_tcp'
  2022-06-20 18:21:22 +0000 [warn]: #0 /var/lib/gems/2.7.0/gems/fluentd-
1.14.6/lib/fluent/plugin/out_forward.rb:410:in `create_transfer_socket'
  2022-06-20 18:21:22 +0000 [warn]: #0 /var/lib/gems/2.7.0/gems/fluentd-
1.14.6/lib/fluent/plugin/out_forward/connection_manager.rb:46:in `call'
  2022-06-20 18:21:22 +0000 [warn]: #0 /var/lib/gems/2.7.0/gems/fluentd-
1.14.6/lib/fluent/plugin/out_forward/connection_manager.rb:46:in `connect'
  2022-06-20 18:21:22 +0000 [warn]: #0 /var/lib/gems/2.7.0/gems/fluentd-
1.14.6/lib/fluent/plugin/out_forward.rb:807:in `connect'
  2022-06-20 18:21:22 +0000 [warn]: #0 /var/lib/gems/2.7.0/gems/fluentd-
1.14.6/lib/fluent/plugin/out_forward.rb:676:in `send_data'
  2022-06-20 18:21:22 +0000 [warn]: #0 /var/lib/gems/2.7.0/gems/fluentd-
1.14.6/lib/fluent/plugin/out_forward.rb:365:in `block in write'
  2022-06-20 18:21:22 +0000 [warn]: #0 /var/lib/gems/2.7.0/gems/fluentd-
1.14.6/lib/fluent/plugin/out_forward/load_balancer.rb:46:in `block in select_healthy_node'
  2022-06-20 18:21:22 +0000 [warn]: #0 /var/lib/gems/2.7.0/gems/fluentd-
1.14.6/lib/fluent/plugin/out_forward/load_balancer.rb:37:in `times'
  2022-06-20 18:21:22 +0000 [warn]: #0 /var/lib/gems/2.7.0/gems/fluentd-
1.14.6/lib/fluent/plugin/out_forward/load_balancer.rb:37:in `select_healthy_node'
  2022-06-20 18:21:22 +0000 [warn]: #0 /var/lib/gems/2.7.0/gems/fluentd-
1.14.6/lib/fluent/plugin_helper/service_discovery/manager.rb:108:in `select_service'
  2022-06-20 18:21:22 +0000 [warn]: #0 /var/lib/gems/2.7.0/gems/fluentd-
1.14.6/lib/fluent/plugin_helper/service_discovery.rb:82:in `service_discovery_select_service'
  2022-06-20 18:21:22 +0000 [warn]: #0 /var/lib/gems/2.7.0/gems/fluentd-
1.14.6/lib/fluent/plugin/out_forward.rb:365:in `write'
  2022-06-20 18:21:22 +0000 [warn]: #0 /var/lib/gems/2.7.0/gems/fluentd-
1.14.6/lib/fluent/plugin/output.rb:1179:in `try_flush'
  2022-06-20 18:21:22 +0000 [warn]: #0 /var/lib/gems/2.7.0/gems/fluentd-
1.14.6/lib/fluent/plugin/out_forward.rb:353:in `try_flush'
  2022-06-20 18:21:22 +0000 [warn]: #0 /var/lib/gems/2.7.0/gems/fluentd-
1.14.6/lib/fluent/plugin/output.rb:1500:in `flush_thread_run'
  2022-06-20 18:21:22 +0000 [warn]: #0 /var/lib/gems/2.7.0/gems/fluentd-
1.14.6/lib/fluent/plugin/output.rb:499:in `block (2 levels) in start'
  2022-06-20 18:21:22 +0000 [warn]: #0 /var/lib/gems/2.7.0/gems/fluentd-
1.14.6/lib/fluent/plugin_helper/thread.rb:78:in `block in thread_create'
2022-06-20 18:21:31 +0000 [warn]: #0 detached forwarding server 'aggregator1' host="172.17.0.2"
port=24500 phi=16.14318557817386 phi_threshold=16
2022-06-20 18:21:31 +0000 [warn]: #0 failed to flush the buffer. retry_times=1 next_retry_time=2022-
06-20 18:21:34 +0000 chunk="5e1e52a8da14c865466db311acfb6f0f" error_class=Errno::EHOSTUNREACH
error="No route to host - connect(2) for \"172.17.0.2\" port 24500"

...
```

It will continue to retry the send until it makes contact with a log aggregator again.

Bring the log aggregator container back online. Since `docker stop` was used to shut it down, use `docker container start` to restore the log aggregator and `docker container attach` to return to its terminal:

```
ubuntu@labsys:~/lab7$ sudo docker container start aggregator1

aggregator1

ubuntu@labsys:~/lab7$
```

Eventually, the forwarder will make contact once the log aggregator instance is operational:

```
...

2022-06-20 18:22:10 +0000 [warn]: #0 recovered forwarding server 'aggregator1' host="172.17.0.2"
port=24500
```

Once the next retry time is reached, all of the buffered events will be sent to the log aggregator in one chunk:

```
...

2022-06-20 18:22:28 +0000 [warn]: #0 retry succeeded. chunk_id="5e1e52a8da14c865466db311acfb6f0f"
```

The log aggregator should have received the original event, and any other events that were buffered in its absence:

```
ubuntu@labsys:~/lab7$ sudo docker logs aggregator1 --tail 10

2022-06-20 18:21:05.495140251 +0000 fluent.info: {"message":"flushing all buffer forcedly"}
2022-06-20 18:21:22.549751002 +0000 fluent.warn: {"retry_times":0,"next_retry_time":"2022-06-20
18:21:24 +0000","chunk":"5e1e52a8da14c865466db311acfb6f0f","error":"#<Errno::EHOSTUNREACH: No route
to host - connect(2) for \"172.17.0.2\" port 24500>","message":"failed to flush the buffer.
retry_times=0 next_retry_time=2022-06-20 18:21:24 +0000 chunk=\"5e1e52a8da14c865466db311acfb6f0f\"
error_class=Errno::EHOSTUNREACH error=\"No route to host - connect(2) for \\\"172.17.0.2\\\" port
24500\""}

...

2022-06-20 18:22:10.383419640 +0000 fluent.warn:
{"host":"172.17.0.2","port":24500,"message":"recovered forwarding server 'aggregator1'
host=\"172.17.0.2\" port=24500"}
2022-06-20 18:22:28.939801239 +0000 fluent.warn:
{"chunk_id":"5e1e52a8da14c865466db311acfb6f0f","message":"retry succeeded.
chunk_id=\"5e1e52a8da14c865466db311acfb6f0f\""}
```

The fact that the `out_forward` plugin is a buffered plugin is advantageous because, in the event that a log aggregator destination goes down, the data is not lost. Due to how loose the log aggregator setup is, all of the other retry events were captured and recorded to the log aggregator as well. Periodic retries are one way to guard against data loss and the ability to save events within a buffer means that, with a small time delay, events are not lost.

Knowing that buffering and retrying are involved with failover with Fluentd, try the following:

- Use `Ctrl C` on the forwarder first, send to an event to it, then restore the forwarder. What happened?
  - What behavior was observed?
  - Was the observed behavior similar or different to what happened when a log aggregator fails?
- Send an event to the forwarder, **then** use `Ctrl C` to shut it down before it sends events to the log aggregator.
  - Restore the forwarder after a minute or so.
  - Did the log aggregator ever receive the event?

3. Exploring High Availability

In the previous step, a single log aggregator went down and events were subject to a delay before they could be received. Time sensitive data may require little to no downtime.

This is a case that can be solved by adding another log aggregator instance for high availability.

Open another terminal and create a configuration for a second log aggregator:

```
ubuntu@labsys:~$ mkdir ~/lab7/aggregator2

ubuntu@labsys:~/lab7/aggregator2$ nano ~/lab7/aggregator2/aggregator2.conf && cat $_

<system>
  process_name aggregator2
</system>

<source>
  @type forward
  port 25500
</source>

<match>
  @type stdout
</match>

ubuntu@labsys:~/lab7/aggregator2$
```

Since this log aggregator is intended to be used as a backup for the primary aggregator, it will be identically configured to the first one.

> This lab shows a single node setup, meaning that a separate port needs to be declared for this setup to be functional. In some environments, the same port may be used if different nodes will host aggregator instances.

Run this log aggregator under Docker:

```
ubuntu@labsys:~/lab7/aggregator2$ sudo docker run -p 25500:25500 -d --name aggregator2 -v
$HOME/lab7/aggregator2:/fluentd/etc \
-e FLUENTD_CONF=aggregator*.conf fluent/fluentd:v1.14.6-1.1

ccce3f7d4dd85016bbff250500eb138fe6ca21e0c55622835e5e601f1b61f1e5

ubuntu@labsys:~/lab7$ sudo docker logs aggregator2 --tail 5

2022-06-20 18:29:36 +0000 [info]: #0 listening port port=25500 bind="0.0.0.0"
2022-06-20 18:29:36 +0000 [info]: #0 fluentd worker is now running worker=0
2022-06-20 18:29:36.529998794 +0000 fluent.info: {"pid":16,"ppid":7,"worker":0,"message":"starting
fluentd worker pid=16 ppid=7 worker=0"}
2022-06-20 18:29:36.530408174 +0000 fluent.info: {"port":25500,"bind":"0.0.0.0","message":"listening
port port=25500 bind=\"0.0.0.0\""}
```

```
2022-06-20 18:29:36.530880378 +0000 fluent.info: {"worker":0,"message":"fluentd worker is now
running worker=0"}

ubuntu@labsys:~/lab7$
```

As before, the forwarder will need a valid address in order to communicate with the second log aggregator. This can be an IP address, a DNS name, or even a service name if you are running Fluentd within some service mesh-backed system (like Kubernetes).

In your working terminal, use `docker inspect` to retrieve the new Fluentd instance's IP address:

```
ubuntu@labsys:~/lab7$ sudo docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}
{{end}}' aggregator2

172.17.0.3

ubuntu@labsys:~/lab7$
```

The aggregator instance should now be functional. It's time to make the forwarder aware that there is a backup instance of Fluentd available for failover.

```
ubuntu@labsys:~/lab7$ nano ~/lab7/forwarder/forwarder.conf && cat $_

<system>
  process_name forwarder
</system>

<source>
  @type forward
  port 32767
</source>

<match>
  @type forward
  <server>
    name aggregator1
    host 172.17.0.2
    port 24500
  </server>
  <server>
    name aggregator2
    host 172.17.0.3
    port 25500
    standby true
  </server>
</match>

ubuntu@labsys:~/lab7$
```

- A second `<server>` subdirective has been added to the forward `<match>` directive
- The `standby true` parameter is in place to inform the forwarder that aggregator2 is a backup instance

Send a `SIGUSR2` to the forwarder to reconfigure it:

```
ubuntu@labsys:~/lab7$ pkill -SIGUSR2 fluentd

pkill: killing pid 51964 failed: Operation not permitted
pkill: killing pid 52132 failed: Operation not permitted
```

```
ubuntu@labsys:~/lab7$
```

`pkill` is trying to restart the containerized Fluentd processes running in your containers but fails to do so. You can safely ignore those warnings.

```
2022-06-20 18:30:35 +0000 [info]: Reloading new config
2022-06-20 18:30:35 +0000 [info]: adding forwarding server 'aggregator1' host="172.17.0.2"
port=24500 weight=60 plugin_id="object:780"
2022-06-20 18:30:35 +0000 [info]: adding forwarding server 'aggregator2' host="172.17.0.3"
port=25500 weight=60 plugin_id="object:780"
2022-06-20 18:30:35 +0000 [info]: using configuration file: <ROOT>
  <system>
    process_name forwarder
  </system>
  <source>
    @type forward
    port 32767
  </source>
  <match>
    @type forward
    <server>
      name "aggregator1"
      host "172.17.0.2"
      port 24500
    </server>
    <server>
      name "aggregator2"
      host "172.17.0.3"
      port 25500
      standby true
    </server>
  </match>
</ROOT>
2022-06-20 18:30:35 +0000 [info]: shutting down input plugin type=:forward plugin_id="object:758"
2022-06-20 18:30:35 +0000 [info]: shutting down output plugin type=:forward plugin_id="object:730"
2022-06-20 18:30:35 +0000 [info]: adding match pattern="**" type="forward"
2022-06-20 18:30:35 +0000 [info]: #0 adding forwarding server 'aggregator1' host="172.17.0.2"
port=24500 weight=60 plugin_id="object:848"
2022-06-20 18:30:35 +0000 [info]: #0 adding forwarding server 'aggregator2' host="172.17.0.3"
port=25500 weight=60 plugin_id="object:848"
2022-06-20 18:30:35 +0000 [info]: adding source type="forward"
2022-06-20 18:30:35 +0000 [info]: #0 shutting down fluentd worker worker=0
2022-06-20 18:30:35 +0000 [info]: #0 shutting down input plugin type=:forward plugin_id="object:758"
2022-06-20 18:30:36 +0000 [info]: #0 shutting down output plugin type=:forward
plugin_id="object:730"
2022-06-20 18:30:36 +0000 [info]: #0 restart fluentd worker worker=0
2022-06-20 18:30:36 +0000 [warn]: #0 define <match fluent.**> to capture fluentd logs in top level
is deprecated. Use <label @FLUENT_LOG> instead
2022-06-20 18:30:36 +0000 [info]: #0 listening port port=32767 bind="0.0.0.0"
```

- What weight was assigned to aggregator2?

Now, stop aggregator1 again using `docker stop`:

```
ubuntu@labsys:~/lab7$ sudo docker container stop aggregator1

aggregator1
```

```
ubuntu@labsys:~/lab7$ sudo docker logs aggregator1 --tail 5

2022-06-20 18:31:06.940383345 +0000 fluent.info: {"worker":0,"message":"fluentd worker is now
stopping worker=0"}
2022-06-20 18:31:06 +0000 [info]: #0 shutting down fluentd worker worker=0
2022-06-20 18:31:06 +0000 [info]: #0 shutting down input plugin type=:forward plugin_id="object:758"
2022-06-20 18:31:06 +0000 [info]: #0 shutting down output plugin type=:stdout plugin_id="object:730"
2022-06-20 18:31:07 +0000 [info]: Worker 0 finished with status 0

ubuntu@labsys:~/lab7$
```

In your working terminal, send an event over to the forwarder, ensuring to flush the buffers after:

```
ubuntu@labsys:~/lab7$ echo '{"json":"message"}' | fluent-cat aggregate.me --port 32767

ubuntu@labsys:~/lab7$ pkill -sigusr1 fluentd

pkill: killing pid 119973 failed: Operation not permitted

ubuntu@labsys:~/lab7$
```

Eventually, the forwarder will realize that aggregator1 is no longer accessible:

```
2022-06-20 18:31:29 +0000 [info]: #0 force flushing buffered events
2022-06-20 18:31:29 +0000 [info]: #0 flushing all buffer forcedly
2022-06-20 18:31:45 +0000 [warn]: #0 detached forwarding server 'aggregator1' host="172.17.0.2"
port=24500 phi=17.037374590761466 phi_threshold=16
2022-06-20 18:31:45 +0000 [warn]: #0 using standby node 172.17.0.3:25500 weight=60
```

Depending on when the event was sent, it may try to flush its events, only to find that it cannot connect to it! Luckily, the standby node was picked up by the forwarder and the event was sent there instead.

Check the events of the backup aggregator, aggregator2:

```
ubuntu@labsys:~/lab7$ sudo docker logs aggregator2 --tail 5

2022-06-20 18:29:36 +0000 [info]: #0 fluentd worker is now running worker=0
2022-06-20 18:29:36.529998794 +0000 fluent.info: {"pid":16,"ppid":7,"worker":0,"message":"starting
fluentd worker pid=16 ppid=7 worker=0"}
2022-06-20 18:29:36.530408174 +0000 fluent.info: {"port":25500,"bind":"0.0.0.0","message":"listening
port port=25500 bind=\"0.0.0.0\""}
2022-06-20 18:29:36.530880378 +0000 fluent.info: {"worker":0,"message":"fluentd worker is now
running worker=0"}
2022-06-20 18:31:26.230841588 +0000 aggregate.me: {"json":"message"}

ubuntu@labsys:~/lab7$
```

The backup aggregator was able to pick up the events and receive them, with very little downtime in between. It will continue to do so until aggregator1 is restored.

Use `docker container start -a` to recover aggregator1:

```
ubuntu@labsys:~/lab7/aggregator1$ sudo docker container start aggregator1

ubuntu@labsys:~/lab7$ sudo docker logs aggregator1 --tail 5
```

```
2022-06-20 18:33:20 +0000 [info]: #0 listening port port=24500 bind="0.0.0.0"
2022-06-20 18:33:20 +0000 [info]: #0 fluentd worker is now running worker=0
2022-06-20 18:33:20.868116085 +0000 fluent.info: {"pid":16,"ppid":7,"worker":0,"message":"starting
fluentd worker pid=16 ppid=7 worker=0"}
2022-06-20 18:33:20.868543627 +0000 fluent.info: {"port":24500,"bind":"0.0.0.0","message":"listening
port port=24500 bind=\"0.0.0.0\""}
2022-06-20 18:33:20.869045569 +0000 fluent.info: {"worker":0,"message":"fluentd worker is now
running worker=0"}

ubuntu@labsys:~/lab7$
```

Soon after, the forwarder will be able to recover its connection with aggregator1:

```
...

2022-06-20 18:33:31 +0000 [warn]: #0 recovered forwarding server 'aggregator1' host="172.17.0.2"
port=24500
```

You have now learned how to perform a basic setup for a multi-instance Fluentd deployment, and how to ensure high availability with failover capabilities.

To recap, much of the high availability solution revolves around the `forward` input and output plugins, and their ability to communicate. As long as they are able to communicate, Fluentd instances will use a combination of repeated retries and event buffering to preserve events that come in before or after a Fluentd instance loss. Understanding how to best configure those plugins is central to a solid Fluentd deployment.

Try tweaking this solution for better performance with the following tasks:

- Remove the `standby true` from the aggregator2 `<server>` subdirective on the forwarder, and send events to it.
  - Does the behavior change?
  - If it does, does the `weight` parameter have any effect?

Cleanup

To prepare for any subsequent labs, please shut down any existing instances using the following commands:

Tear down any running Docker containers:

```
ubuntu@labsys:~/lab7$ sudo docker container rm $(sudo docker container stop aggregator1 aggregator2)

aggregator1
aggregator2

ubuntu@labsys:~/lab7$ cd

ubuntu@labsys:~$
```

Use `CTRL C` to terminate any locally running instances of Fluentd, like the forwarder:

```
^C

2022-06-20 18:36:54 +0000 [info]: Received graceful stop
2022-06-20 18:36:55 +0000 [info]: #0 fluentd worker is now stopping worker=0
2022-06-20 18:36:55 +0000 [info]: #0 shutting down fluentd worker worker=0
2022-06-20 18:36:55 +0000 [info]: #0 shutting down input plugin type=:forward plugin_id="object:870"
2022-06-20 18:36:55 +0000 [info]: #0 shutting down output plugin type=:forward
```

```
plugin_id="object:848"
2022-06-20 18:36:58 +0000 [info]: Worker 0 finished with status 0

ubuntu@labsys:~/lab7/forwarder$ cd

ubuntu@labsys:~$
```

It may take some time for the log forwarder to shut down, if it has not detected that both of the log aggregators it is connected to have gone down.

Congratulations, you have completed the lab!