LFS242 - Cloud Native Logging with Fluentd

Lab 2 – Configuring Fluentd

- Were there any plugins loaded?
  - No. Since there were no directives written in the configuration file, no plugins were loaded.
- Can Fluentd perform any functions at this point?
  - No. Without any directives configuring any plugins for event processing, this configuration is effectively nonfunctional.
- What would the benefit of running a blank configuration file be?
  - If Fluentd is able to start at all, it shows that all of its dependencies are fulfilled and that Fluentd itself is healthy.
- What circumstances would it be appropriate to run Fluentd with a specific configuration file like this be?
  - Development
  - Debugging
  - Testing a Fluentd deployment on a specific host
  - Reusing known good configuration files originating from other systems
- Where would using an environment variable to set the configuration file be most appropriate?
  - Containers and pre-packaged deployments benefit the most from configuration files defined in environment variables.
- Would it be possible to run more than one Fluentd instance on the same machine using more than one terminal?
  - Yes, as long as their ports do not conflict on the host.
- What happens during a restart event?
  - Fluentd receives the signal, terminates the worker and then restarts with the new configuration.
- How many sources are loaded now?
  - 2 sources are now loaded: one forward and the other http.
- Based on the above configuration, what files will be tailed under the /tmp/fluent directory?
  - All of them, since a wildcard was put into place.
- What does the tail plugin tell you about the way this plugin will work?
  - It will use the tail function to create an event from an entry within a specific file.
- Before it failed to reload, how many plugins were loaded after the SIGHUP was sent?
  - Two: The in_forward and in_http plugins were after Fluentd was restarted and before it crashed.
- What is different about this crash compared to the previous one?
  - It is much shorter, barely identifying that the optional OJ json parser was missing before crashing.
- Were any plugins loaded before this error was generated?
  - Not this time, no.
- Did you get any indication that any files are being tracked under the /tmp/fluent/ directory?
  - Yes, Fluentd reported that it is tailing the following files: `/tmp/fluent/lab2.tail.pos` and `/tmp/fluent/applogs`
- What plugins are loaded when this Fluentd configuration file is used?
  - in_forward, in_http, in_tail, parse_none
- Are these plugins loaded in any particular order?
  - Yes, the plugins are loaded in the order that they are listed in the configuration file as long as Fluentd successfully parses it at runtime.
- Is there any benefit to being able to run Fluentd configurations with only `<source>` directives?

    - Only for developing and testing new inputs, otherwise no.
- What plugin will be called?
    - out_stdout
- From the plugin that's been selected, what can you infer will happen to matching events?
    - Events should be sent to the host's standard output.
- What event tags will be caught by this `<match>` directive?
    - All event tags, since a double wildcard was used.
- How does the new event abide by the new configurations?
    - Events printed to STDOUT now have the "event_tag" key-value pair inside them, which was the name given in the configuration.
- Try to use a combination of the other wildcard types in another `<match>` directive
    - An example:

```
<match *.lab2 lfs242.*>
  @type stdout
</match>
```

- How can you change the in_tail `<source>` directive to work with the `<match mod2.\*>` directive?
    - The tail `<source>` directive will need to be changed to send a tag with mod2.*

```
<source>
  @type tail
  path /tmp/fluent/*.log
  <parse>
    @type none
  </parse>
  tag mod2.tailed
  pos_file /tmp/fluent/lab2.tail.pos
</source>
```