

LFS242 - Cloud Native Logging with Fluentd

Lab 6 – Organizing complex configurations with labels and includes

Fluentd configurations can become complex as directives are added to a configuration file (especially if Fluentd is being used as a log aggregator to handle many inputs from multiple sources and outputs to multiple destinations). One way of organizing Fluentd configurations is through the use of labels. Labels group and directives together into processing pipelines using the label directive. Other directives can forward messages to a given labeled pipeline by name.

In addition to labels, entirely separate configuration files can be used to separate directives from each other. By using the `@include` parameter in a Fluentd configuration file, the directives contained in a remote configuration file can be loaded as though they were part of the original configuration file. Handling scenarios with multiple data pipelines can be greatly simplified through the use of labels and `@include` statements.

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

- Use Fluentd as a log aggregator and processor for multiple container logs
- Utilize labels to create multiple pipelines within a Fluentd configuration file
- Create modularized Fluentd configurations using `@include` statements

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 be using Docker to run instances of Redis and Apache, 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. Utilizing Labels to organize data processing pipelines in Fluentd

Labels allow a user to send events to specific groups of `<filter>` and `<match>` directives. Labels are initially set in `<source>` directives and are plugins available that allow labels to be changed for additional routing. Events that are routed through a label are only processed by the directives nested under a `<label>` directive whose pattern matches their set label.

Labels are a way of establishing pipelines without having to worry about the order of directives in the configuration file.

Fluentd will be used to manage several docker application logs, so to start, create a simple source/match configuration consisting of all forwards:

```
ubuntu@labsys:~$ mkdir ~/lab6 ; cd $_
ubuntu@labsys:~/lab6$ nano lab6.conf && cat $_

<source>
  # WordPress Database
  @type forward
  port 24000
</source>

<source>
  # WordPress
  @type forward
  port 24100
</source>

<source>
  # Guestbook Database
  @type forward
  port 24200
</source>

<source>
  # Guestbook
  @type forward
  port 24300
</source>

<match *>
  @type stdout
</match>

ubuntu@labsys:~$
```

Remember each of the ports listed under the `<source>` directives.

This configuration will allow the Docker containers in the next step to be started using Fluentd as the log engine. Docker requires a Fluentd instance listening on 24244 by default in order to start using it.

Launch Fluentd using this configuration:

```
ubuntu@labsys:~/lab6$ fluentd -c lab6.conf
```

```

2022-06-20 17:36:22 +0000 [info]: parsing config file is succeeded path="lab6.conf"

...

2022-06-20 17:36:22 +0000 [info]: #0 fluentd worker is now running worker=0
2022-06-20 17:36:22.953966251 +0000 fluent.info:
{"pid":49891,"ppid":49886,"worker":0,"message":"starting fluentd worker pid=49891 ppid=49886
worker=0"}
2022-06-20 17:36:22.954702599 +0000 fluent.info: {"port":24300,"bind":"0.0.0.0","message":"listening
port port=24300 bind=\"0.0.0.0\""}
2022-06-20 17:36:22.955711108 +0000 fluent.info: {"port":24200,"bind":"0.0.0.0","message":"listening
port port=24200 bind=\"0.0.0.0\""}
2022-06-20 17:36:22.956658285 +0000 fluent.info: {"port":24100,"bind":"0.0.0.0","message":"listening
port port=24100 bind=\"0.0.0.0\""}
2022-06-20 17:36:22.958331974 +0000 fluent.info: {"port":24000,"bind":"0.0.0.0","message":"listening
port port=24000 bind=\"0.0.0.0\""}
2022-06-20 17:36:22.958616379 +0000 fluent.info: {"worker":0,"message":"fluentd worker is now
running worker=0"}

```

There is now a Fluentd instance listening on the ports configured. Make sure to use these ports when launching the containers.

1. Launching containerized applications

For this lab, two different applications will be launched. One will be a WordPress instance backed by MariaDB, which is a fork of MySQL created by the original developers of MySQL. The second application will be a guestbook application from the Kubernetes documentation, found here: <https://kubernetes.io/docs/tutorials/stateless-application/guestbook/>. All of these will be run under Docker, and you will be interacting with them to generate events.

In a new working terminal, deploy the WordPress instance using the ports selected in the Fluentd configuration file:

```

ubuntu@lab6:~$ cd ~/lab6

ubuntu@lab6:~/lab6$ sudo docker run --name wordpress-db \
-d \
-e MYSQL_ROOT_PASSWORD=mypassword \
-e MYSQL_DATABASE=wpdb \
-e MYSQL_USER=wpuser \
-e MYSQL_PASSWORD=wppassword \
--log-driver fluentd --log-opt tag={{.Name}} --log-opt fluentd-address=localhost:24000 \
mariadb:10.7.4-focal

...

b9be61b57968443cd3edfc2708eb58b36147ca57d7ff216d8475baa9b44c6bd6

ubuntu@lab6:~$

```

- The WordPress database container will run MariaDB
- It will be named `wordpress-db` and run as a daemon in detached mode
- Each `-e` variable is needed to set the root credentials and database for MariaDB to function correctly
- The `--log-driver` and `--log-opt` s flags will ensure that it will forward traffic to Fluentd on port 24000

This will handle the backend database for WordPress, launch it now:

```

ubuntu@lab6:~/lab6$ sudo docker run --name wordpress \
-d \

```

```
-P --link wordpress-db \
-e WORDPRESS_DB_USER=wpuser \
-e WORDPRESS_DB_PASSWORD=wppassword \
-e WORDPRESS_DB_NAME=wpdb \
-e WORDPRESS_DB_HOST=wordpress-db \
--log-driver fluentd --log-opt tag={{.Name}} --log-opt fluentd-address=localhost:24100 \
wordpress:6.0.0-apache

...

439b40dd421f1322072553c44871824813b1d2331365cf80195d1e779a79e926

ubuntu@lab6sys:~/lab6$
```

In order to establish a connection with the WordPress database container, the `--link` flag is used so that the WordPress container will see the wordpress-db container as a valid host on the internal container network.

- On which port will WordPress send its logs to Fluentd?

Now that WordPress is deployed, it's time to deploy another web application: a simple PHP-based Guestbook application. This application working alongside WordPress, and will provide a good example of how Fluentd can manage logs from two separate services.

Deploy the Redis database and frontend guestbook instances as containers:

```
ubuntu@lab6sys:~/lab6$ sudo docker network create guestbook

ubuntu@lab6sys:~/lab6$ sudo docker run --name guestbook-db \
-d \
-p 6379 \
--log-driver fluentd --log-opt tag={{.Name}} --log-opt fluentd-address=localhost:24200 \
--network guestbook \
k8s.gcr.io/redis:e2e

...

43cfc54a187d86384b591cefcdad66e258e33a59d7fe3a7d3a9400cfb656db

ubuntu@lab6sys:~/lab6$ sudo docker run --name guestbook \
-d -P \
-e GET_HOSTS_FROM=env \
-e REDIS_MASTER_SERVICE_HOST=guestbook-db \
-e REDIS_SLAVE_SERVICE_HOST=guestbook-db \
--log-driver fluentd --log-opt tag={{.Name}} --log-opt fluentd-address=localhost:24300 \
--network guestbook \
gcr.io/google-samples/gb-frontend:v4

...

92afd9985ebb087d956c42fd7207d67f451e3851dea45e7fdfb50f3717c6ee89

ubuntu@lab6sys:~/lab6$
```

The Guestbook application should now be deployed on the virtual machine, sending application data to Fluentd listeners taking traffic on ports 24200 and 24300. Each of the containers will output all their log traffic to their local STDOUT buffers, which are picked up by Fluentd and sent as standardized JSON events.

This lab will require some interaction with both WordPress and the Guestbook. These interactions can be performed on the virtual machine's host using the local IP of the VM itself or the external IP address if using a cloud instance.

Collect the IP address of the virtual machine. For local VMs use the following command to look up the local ethernet IP address:

For cloud instances, simply use the same external IP used to SSH into the machine.

```
ubuntu@labsys:~/lab6$ hostname -I
labsys 172.17.0.1
ubuntu@labsys:~/lab6$
```

Then, using `docker ps`, find out which ports are mapped to each of the frontend services:

```
ubuntu@labsys:~/lab6$ sudo docker ps
```

CONTAINER ID	IMAGE	COMMAND	NAMES	CREATED
92afd9985ebb	gcr.io/google-samples/gb-frontent:v4	"apache2-foreground"	guestbook	46 seconds ago
Up 45 seconds	0.0.0.0:49165->80/tcp, :::49165->80/tcp			
43cfc54a187d	k8s.gcr.io/redis:e2e	"redis-server /etc/r..."	guestbook-db	About a minute ago
Up About a minute	0.0.0.0:49164->6379/tcp, :::49164->6379/tcp			
439b40dd421f	wordpress:6.0.0-apache	"docker-entrypoint.s..."	wordpress	2 minutes ago
Up 2 minutes	0.0.0.0:49163->80/tcp, :::49163->80/tcp			
b9be61b57968	mariadb:10.7.4-focal	"docker-entrypoint.s..."	wordpress-db	2 minutes ago
Up 2 minutes	3306/tcp			

```
ubuntu@labsys:~/lab6$
```

The Guestbook frontend is mapped to port `49165` on the host machine, which points to port `80` on the container. The WordPress frontend is mapped to port `49163`, which maps to port `80` of its own container. Your resulting ports may vary, so moving forward be sure to select the ports that are shown in your terminal.

In a browser on either the local machine or the host machine, connect to each frontend using

`http://<VM External IP>:<Wordpress Container Port>` and
`http://<VM External IP>:<Guestbook Container Port>` to access WordPress and the Guestbook respectively.

34.222.173.218:49153/wp-admin/install.php



English (United States)

Afrikaans

العربية

العربية المغربية

অসমীয়া

Azərbaycan dili

گۆنئی آذربایجان

The WordPress Frontend should send you to an installation page that asks for a language selection. There is no need to proceed with the installation at this time.

Guestbook

Messages

Submit

The Guestbook should present a page with a simple text box for Guestbook entries with a submission button. There is no need to add any entries at this time, but feel free to do so to test functionality.

If you look back in the Fluentd terminal, each of the application containers should have sent their output to Fluentd's terminal:

```
...

2022-06-20 17:37:09.000000000 +0000 wordpress-db:
{"container_id":"b9be61b57968443cd3edfc2708eb58b36147ca57d7ff216d8475baa9b44c6bd6","container_name":
"/wordpress-db","source":"stderr","log":"2022-06-20 17:37:09 0 [Note] mariadb: ready for
connections."}

...

2022-06-20 17:37:11.000000000 +0000 wordpress:
{"container_id":"439b40dd421f1322072553c44871824813b1d2331365cf80195d1e779a79e926","container_name":
"/wordpress","source":"stderr","log":"[Mon Jun 20 17:37:11.691271 2022] [core:notice] [pid 1]
AH00094: Command line: 'apache2 -D FOREGROUND'"}

...

2022-06-20 17:38:17.000000000 +0000 guestbook-db: {"container_name":"/guestbook-
db","source":"stdout","log":"[1] 20 Jun 17:38:17.102 * The server is now ready to accept connections
on port 6379","container_id":"43cfc54a187d86384b591cefc0cdad66e258e33a59d7fe3a7d3a9400cfb656db"}

...

2022-06-20 17:38:43.000000000 +0000 guestbook:
{"container_id":"92afd9985ebb087d956c42fd7207d67f451e3851dea45e7fdfb50f3717c6ee89","container_name":
"/guestbook","source":"stdout","log":"[Mon Jun 20 17:38:43.453776 2022] [mpm_prefork:notice] [pid 1]
AH00163: Apache/2.4.10 (Debian) PHP/5.6.20 configured -- resuming normal operations"}

...
```

The multi-application environment is now ready to go with two applications - WordPress and a Guestbook - up and running.

2. Organizing pipelines with labels

Labels are a way of organizing directives inside a configuration file outside of to using tags. They are typically set inside a `<source>` directive, but they can be set at any point in a pipeline as long as a plugin supports the `@label` parameter.

Once a label is assigned to a set of events, it will be routed to a `<label>` directive with a pattern that matches the set label. Each `<label>` directive contains its own set of `<filter>` and `<match>` directives that are only executed against events that are captured by the `<label>` directive. Once captured, standard tag-based processing occurs.

`<label>` directives can be placed within any Fluentd configuration file; since events possessing a certain label are routed directly to a matching `<label>` directive, any other directives that precede the `<label>` directive are ignored.

2a. Organizing pipelines with labels

To begin, add @label parameters to each of the forward `<source>` directives:

```
ubuntu@labsys:~/lab6$ nano lab6.conf && cat $_
```

```
<source>
# WordPress Database
@type forward
port 24000
@label wordpress
</source>

<source>
# WordPress
@type forward
port 24100
@label wordpress
</source>

<source>
# Guestbook Database
@type forward
port 24200
@label guestbook
</source>

<source>
# Guestbook
@type forward
port 24300
@label guestbook
</source>

<match **>
@type stdout
</match>

<label wordpress>
<match **>
@type file
path /tmp/lab6/wordpress-log
<buffer>
timekey 60s
timekey_wait 1m
</buffer>
</match>
</label>

<label guestbook>
<match **>
@type file
path /tmp/lab6/guestbook-log
<buffer>
timekey 60s
timekey_wait 1m
</buffer>
</match>
</label>

ubuntu@labsys:~/lab6$
```

Once a label is established, a matching `<label>` directive **must** be present inside the configuration file. Attempting to start a configuration file using a label while omitting the matching `<label>` directive will prevent Fluentd from starting or reconfiguring. This is one of the few cases outside of a syntax error that Fluentd will not start due to a configuration omission.

Each of these `<label>` directives is configured to send log data to separate logs based on each application: one label will send all WordPress traffic from the database and frontend to `/tmp/wordpress-log` and all Guestbook database and Frontend traffic to `/tmp/guestbook-log`.

For this lab, each of the files will be written out every minute based on the `timekey_wait` option.

Send a `SIGUSR2` to the Fluentd instance to reconfigure it:

```
ubuntu@labsys:~/lab6$ pkill -SIGUSR2 fluentd
```

Check the Fluentd terminal to make sure that it reconfigured correctly:

```
...

2022-06-20 17:42:04 +0000 [info]: adding match in wordpress pattern="*" type="file"
2022-06-20 17:42:04.719798649 +0000 fluent.info: {"pattern":"*","type":"file","message":"adding
match in wordpress pattern=\"*\" type=\"file\""}
2022-06-20 17:42:04.751787989 +0000 fluent.info: {"pattern":"*","type":"file","message":"adding
match in guestbook pattern=\"*\" type=\"file\""}
2022-06-20 17:42:04 +0000 [info]: adding match in guestbook pattern="*" type="file"
2022-06-20 17:42:04.782443714 +0000 fluent.info: {"pattern":"*","type":"stdout","message":"adding
match pattern=\"*\" type=\"stdout\""}
2022-06-20 17:42:04 +0000 [info]: adding match pattern="*" type="stdout"
2022-06-20 17:42:04.807073789 +0000 fluent.info: {"message":"Oj isn't installed, fallback to Yajl as
json parser"}
2022-06-20 17:42:04 +0000 [info]: #0 Oj isn't installed, fallback to Yajl as json parser
2022-06-20 17:42:04.808101578 +0000 fluent.info: {"type":"forward","message":"adding source
type=\"forward\""}
2022-06-20 17:42:04 +0000 [info]: adding source type="forward"
2022-06-20 17:42:04.811695196 +0000 fluent.info: {"type":"forward","message":"adding source
type=\"forward\""}
2022-06-20 17:42:04 +0000 [info]: adding source type="forward"
2022-06-20 17:42:04.814970888 +0000 fluent.info: {"type":"forward","message":"adding source
type=\"forward\""}
2022-06-20 17:42:04 +0000 [info]: adding source type="forward"
2022-06-20 17:42:04.818252747 +0000 fluent.info: {"type":"forward","message":"adding source
type=\"forward\""}
2022-06-20 17:42:04 +0000 [info]: adding source type="forward"
2022-06-20 17:42:04 +0000 [info]: #0 shutting down fluentd worker worker=0
2022-06-20 17:42:04 +0000 [info]: #0 shutting down input plugin type=:forward plugin_id="object:758"
2022-06-20 17:42:04 +0000 [info]: #0 shutting down input plugin type=:forward plugin_id="object:76c"
2022-06-20 17:42:04 +0000 [info]: #0 shutting down input plugin type=:forward plugin_id="object:794"
2022-06-20 17:42:04 +0000 [info]: #0 shutting down input plugin type=:forward plugin_id="object:780"
2022-06-20 17:42:04 +0000 [info]: #0 shutting down output plugin type=:stdout plugin_id="object:730"
2022-06-20 17:42:05 +0000 [info]: #0 restart fluentd worker worker=0
2022-06-20 17:42:05 +0000 [warn]: #0 define <match fluent.*> to capture fluentd logs in top level
is deprecated. Use <label @FLUENT_LOG> instead
2022-06-20 17:42:05 +0000 [info]: #0 listening port port=24300 bind="0.0.0.0"
2022-06-20 17:42:05 +0000 [info]: #0 listening port port=24200 bind="0.0.0.0"
2022-06-20 17:42:05 +0000 [info]: #0 listening port port=24100 bind="0.0.0.0"
2022-06-20 17:42:05 +0000 [info]: #0 listening port port=24000 bind="0.0.0.0"
```

If you see `adding match in wordpress pattern...`, then it succeeded. Since Fluentd was configured to send events over to a set of files rather than STDOUT, it will not be outputting any container-specific data.

Now that Fluentd is reconfigured, check the /tmp directory to see where the logs should be written to:

```
ubuntu@lab6:~/lab6$ ls -l /tmp/lab6

total 8
drwxr-xr-x 2 ubuntu ubuntu 4096 Jun 20 17:42 guestbook-log
drwxr-xr-x 2 ubuntu ubuntu 4096 Jun 20 17:25 wordpress-log

ubuntu@lab6:~/lab6$
```

A pair of directories, guestbook-log and wordpress-log are now present.

- Are those directories where the final log files will be made?

2b. Generating Events from WordPress and the Guestbook:

It's time to create some events for Fluentd to process.

WordPress

In your browser window, click **Continue** in the WordPress Frontend GUI to proceed with the installation:

Welcome

Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

Information needed

Please provide the following information. Don't worry, you can always change these settings later.

Site Title

Username
Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.

Password

Strong

Important: You will need this password to log in. Please store it in a secure location.

Your Email
Double-check your email address before continuing.

Search Engine Visibility ☒ Discourage search engines from indexing this site
It is up to search engines to honor this request.

- Site Title: **LFS242**
- Username: **student**
- Your email: **student@lfs242.linuxfoundation.org**
- Discourage search engines from indexing this site
- Copy the password to your clipboard, or set something simple to remember

Then click **Install WordPress** when all fields are filled out:

Success!

WordPress has been installed. Thank you, and enjoy!

Username

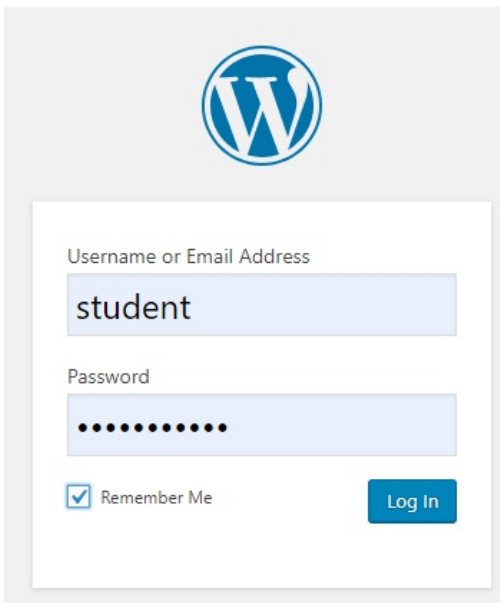
student

Password

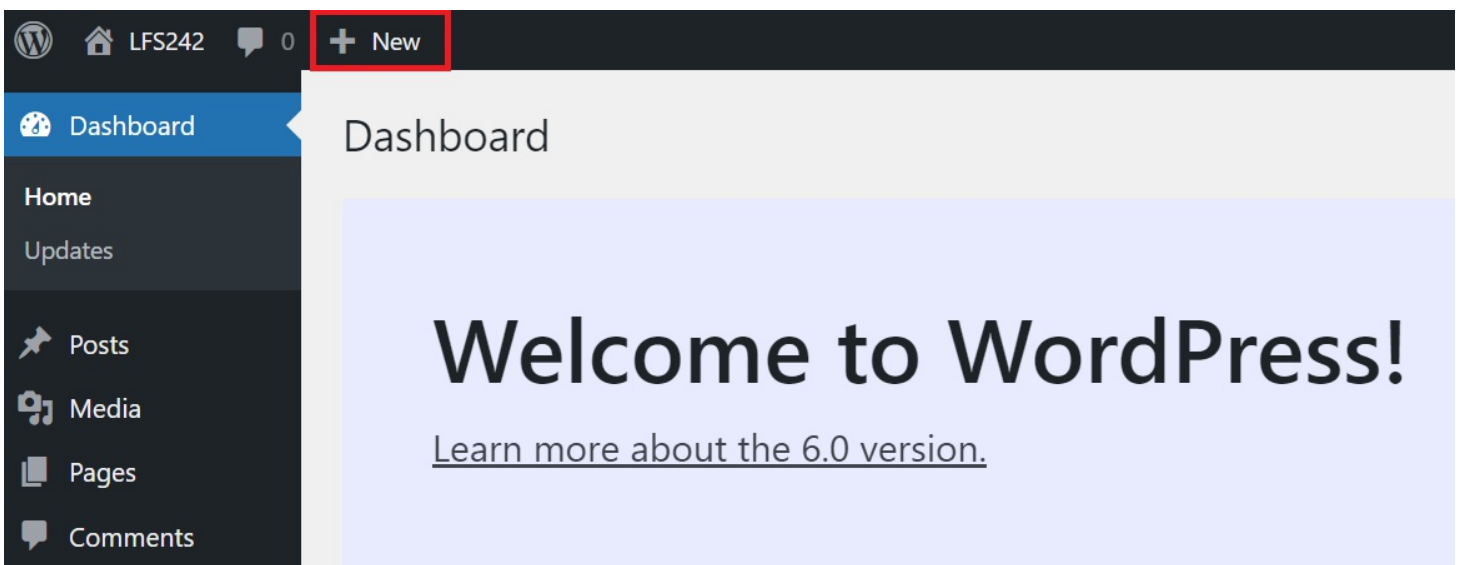
Your chosen password.

Log In

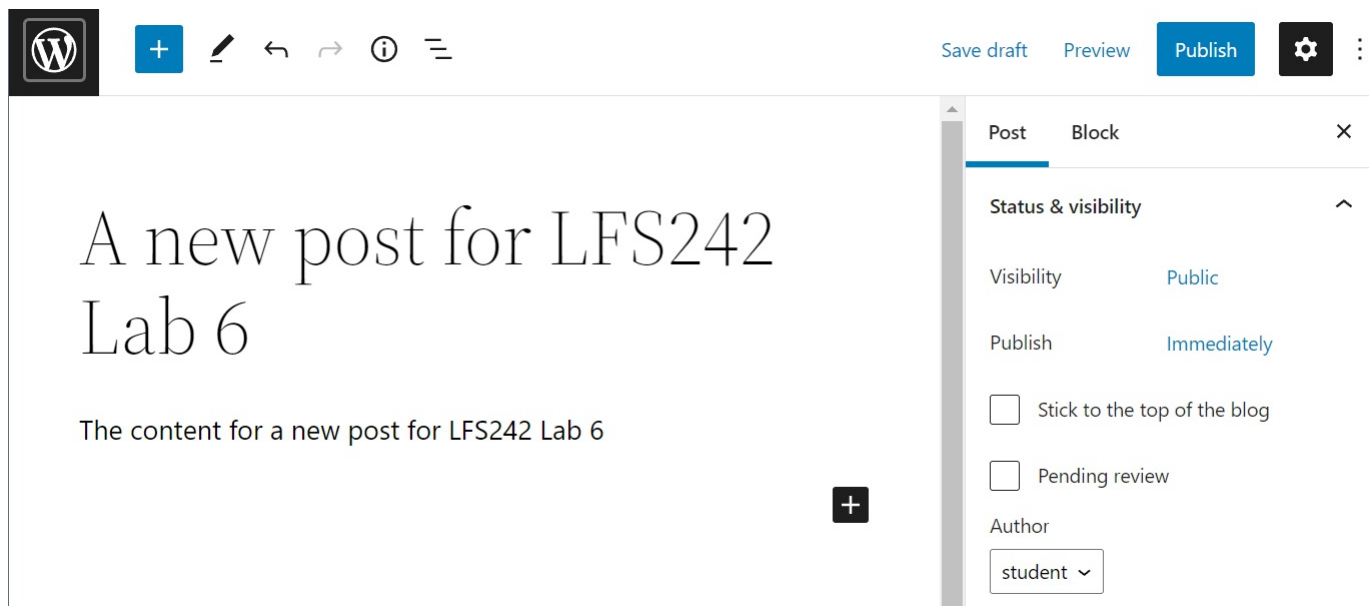
Login to the WordPress dashboard using the username and password you set:

A screenshot of the WordPress login form. At the top is the WordPress logo. Below it is a white box containing the login fields. The first field is labeled "Username or Email Address" and contains the text "student". The second field is labeled "Password" and contains a series of dots. Below the password field is a checkbox labeled "Remember Me" which is checked. To the right of the checkbox is a blue "Log In" button.

On the WordPress dashboard, click "+ New" at the top bar:

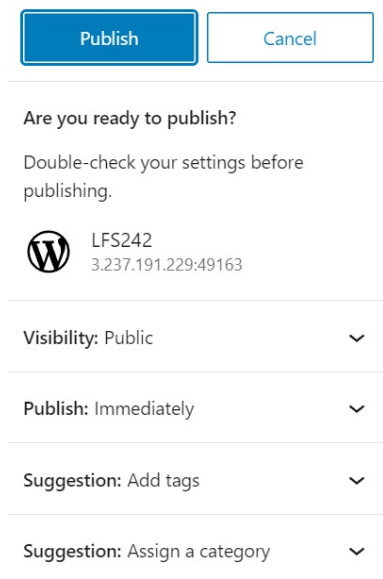
A screenshot of the WordPress dashboard. The top bar is dark grey and contains the WordPress logo, a home icon, the text "LFS242", a comment icon with "0", and a "+ New" button which is highlighted with a red box. The left sidebar is dark grey and contains a "Dashboard" button (highlighted with a blue bar), "Home", "Updates", "Posts", "Media", "Pages", and "Comments". The main content area is light grey and contains the text "Dashboard" and a large blue box with the text "Welcome to WordPress!" and a link "[Learn more about the 6.0 version.](#)".

Close any dialogues that pop up, then add a title to the new post with some content:



The image shows the WordPress post editor interface. At the top, there is a toolbar with icons for adding blocks, editing, undo, redo, help, and a menu. To the right of the toolbar are buttons for 'Save draft', 'Preview', and 'Publish', along with a settings gear icon and a vertical ellipsis. The main content area displays the title 'A new post for LFS242 Lab 6' and the text 'The content for a new post for LFS242 Lab 6'. On the right side, there is a sidebar with tabs for 'Post' and 'Block'. The 'Post' tab is active, showing settings for 'Status & visibility'. Under this section, 'Visibility' is set to 'Public', 'Publish' is set to 'Immediately', and there are checkboxes for 'Stick to the top of the blog' and 'Pending review'. The 'Author' section shows 'student' with a dropdown arrow.

When finished, click Publish... in the top right corner:



The image shows the 'Publish' confirmation dialog. At the top, there are two buttons: 'Publish' and 'Cancel'. Below them, the text asks 'Are you ready to publish?' and advises to 'Double-check your settings before publishing.' The user's profile is shown as 'LFS242' with the ID '3,237,191,229:49163'. Below this, there are four settings with dropdown arrows: 'Visibility: Public', 'Publish: Immediately', 'Suggestion: Add tags', and 'Suggestion: Assign a category'.

And finally, click Publish to create your post:

A new post for LFS242 Lab 6 is now live.

What's next?

Post address

http://3.237.191.229:49163...

Copy

View Post

Add New Post

Guestbook

In the text box on the Guestbook page, type a message:

Guestbook

I am a guestbook entry!

Submit

Click Submit:

Guestbook

Messages

Submit

I am a guestbook entry!

Before you check the logs in the /tmp/lab6 directory, you may need to trigger more activity by performing actions inside the frontend Browser instances.

You can also force a buffer flush by sending a **SIGUSR1** to Fluentd, then check the directory:

```
ubuntu@lab6sys:~/lab6$ pkill -SIGUSR1 fluentd
```

```
ubuntu@lab6sys:~/lab6$ ls -l /tmp/lab6
```

```
total 96
```

```
drwxr-xr-x 2 ubuntu ubuntu 4096 Jun 20 17:45 guestbook-log
-rw-r--r-- 1 ubuntu ubuntu 472 Jun 20 17:45 guestbook-log.202206201745_0.log
drwxr-xr-x 2 ubuntu ubuntu 4096 Jun 20 17:45 wordpress-log
-rw-r--r-- 1 ubuntu ubuntu 33197 Jun 20 17:45 wordpress-log.202206201743_0.log
-rw-r--r-- 1 ubuntu ubuntu 37301 Jun 20 17:45 wordpress-log.202206201744_0.log
-rw-r--r-- 1 ubuntu ubuntu 4732 Jun 20 17:45 wordpress-log.202206201745_0.log
```

```
ubuntu@labsys:~/lab6$
```

Some new `.log` files have been written. Since the `file` output plugin is a buffered plugin, events are sent to a buffer that will periodically write out to the actual files after a given time or size requirement is met, which is configured by the `<match>` directive's `<buffer>` subdirective.

Inspect the new logs using `tail`, replacing the `log.*.log` with one printed in the previous step:

```
ubuntu@labsys:~/lab6$ tail -2 /tmp/lab6/wordpress-log.202206201743_0.log

2022-06-20T17:43:42+00:00      wordpress      {"log":"99.149.248.237 - - [20/Jun/2022:17:43:42
+0000] \"GET /wp-admin/admin-ajax.php?action=dashboard-
widgets&widget=dashboard_primary&pagenow=dashboard HTTP/1.1\" 200 981
\"http://3.237.191.229:49163/wp-admin/\" \"Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.0.0
Safari/537.36\"\", \"container_id\":\"439b40dd421f1322072553c44871824813b1d2331365cf80195d1e779a79e926\", \"
container_name\":\"/wordpress\", \"source\":\"stdout\"}
2022-06-20T17:43:50+00:00      wordpress      {"source":"stdout", "log":"127.0.0.1 - -
[20/Jun/2022:17:43:50 +0000] \"OPTIONS * HTTP/1.0\" 200 126 \"-\" \"Apache/2.4.53 (Debian)
PHP/7.4.30 (internal dummy
connection)\"\", \"container_id\":\"439b40dd421f1322072553c44871824813b1d2331365cf80195d1e779a79e926\", \"co
ntainer_name\":\"/wordpress\"}

ubuntu@labsys:~/lab6$
```

```
ubuntu@labsys:~/lab6$ tail -2 /tmp/lab6/guestbook-log.202206201745_0.log

2022-06-20T17:45:38+00:00      guestbook      {"log":"99.149.248.237 - - [20/Jun/2022:17:45:38
+0000] \"GET /guestbook.php?cmd=set&key=messages&value=,I%20am%20a%20guestbook%20entry! HTTP/1.1\"
200 255 \"http://3.237.191.229:49165/\" \"Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.0.0
Safari/537.36\"\", \"container_id\":\"92afd9985ebb087d956c42fd7207d67f451e3851dea45e7fdfb50f3717c6ee89\", \"
container_name\":\"/guestbook\", \"source\":\"stdout\"}

ubuntu@labsys:~/lab6$
```

Excellent! Each application is now having its logs sent to a unified log of their own. Fluentd is routing them based on the user-configurable label rather than the tag.

2c. Changing Labels with Relabel

Using the `relabel` output plugin, events can be assigned another tag in the middle of a processing pipeline without rewriting or removing the tag from the pipeline. This can be useful for changing the routing of labeled events, or aggregating logs that initially came under different tags.

In this example configuration, use the `relabel` plugin to separate the logs by purpose - by frontend and database:

```
ubuntu@labsys:~/lab6$ nano lab6.conf && cat $_
<source>
```

```

# WordPress Database
@type forward
port 24000
@label wordpress
</source>

<source>
# WordPress
@type forward
port 24100
@label wordpress
</source>

<source>
# Guestbook Database
@type forward
port 24200
@label guestbook
</source>

<source>
# Guestbook
@type forward
port 24300
@label guestbook
</source>

<label wordpress>
  <match wordpress>
    @type relabel
    @label frontend
  </match>
  <match wordpress-db>
    @type relabel
    @label db
  </match>
</label>

<label guestbook>
  <match guestbook>
    @type relabel
    @label frontend
  </match>
  <match guestbook-db>
    @type relabel
    @label db
  </match>
</label>

<label frontend>
  <match **>
    @type file
    path /tmp/lab6/frontend-log
    <buffer>
      timekey 60s
      timekey_wait 1m
    </buffer>
  </match>
</label>

<label db>
  <match **>

```

```
@type file
path /tmp/lab6/database-log
<buffer>
  timekey 60s
  timekey_wait 1m
</buffer>
</match>
</label>

ubuntu@labsys:~/lab6$
```

The `relabel` plugin outputs an event to another label and does not remove that event from the processing stream. Like the regular label parameter, it must be paired with a valid `<label>` directive once it is established inside a configuration file in order for Fluentd to reload correctly.

Reconfigure Fluentd with a `SIGUSR2`. Restart the `wordpress-db` and `guestbook-db` containers to ensure they reconnect and send events:

```
ubuntu@labsys:~/lab6$ kill -SIGUSR2 fluentd

ubuntu@labsys:~/lab6$ sudo docker restart wordpress-db guestbook-db

wordpress-db
guestbook-db

ubuntu@labsys:~/lab6$
```

```
...

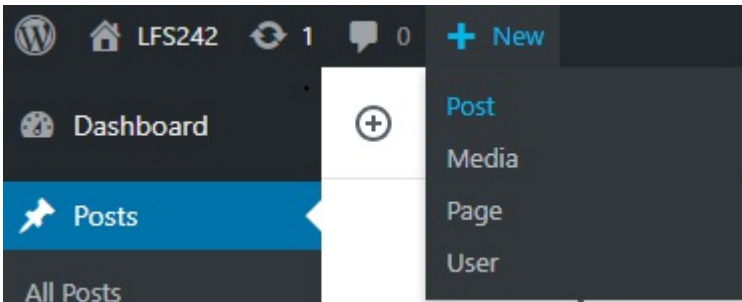
2022-06-20 17:56:44 +0000 [info]: adding match in wordpress pattern="wordpress" type="relabel"
2022-06-20 17:56:44 +0000 [info]: adding match in wordpress pattern="wordpress-db" type="relabel"
2022-06-20 17:56:44 +0000 [info]: adding match in guestbook pattern="guestbook" type="relabel"
2022-06-20 17:56:44 +0000 [info]: adding match in guestbook pattern="guestbook-db" type="relabel"
2022-06-20 17:56:44 +0000 [info]: adding match in frontend pattern="*" type="file"
2022-06-20 17:56:44 +0000 [info]: adding match in db pattern="*" type="file"
2022-06-20 17:56:44 +0000 [info]: adding source type="forward"
2022-06-20 17:56:44 +0000 [info]: adding source type="forward"
2022-06-20 17:56:44 +0000 [info]: adding source type="forward"
2022-06-20 17:56:44 +0000 [info]: adding source type="forward"
2022-06-20 17:56:44 +0000 [info]: #0 shutting down fluentd worker worker=0
2022-06-20 17:56:44 +0000 [info]: #0 shutting down input plugin type=:forward plugin_id="object:9d8"
2022-06-20 17:56:44 +0000 [info]: #0 shutting down input plugin type=:forward plugin_id="object:9ec"
2022-06-20 17:56:44 +0000 [info]: #0 shutting down input plugin type=:forward plugin_id="object:a00"
2022-06-20 17:56:44 +0000 [info]: #0 shutting down input plugin type=:forward plugin_id="object:a14"
2022-06-20 17:56:44 +0000 [info]: #0 shutting down output plugin type=:file plugin_id="object:938"
2022-06-20 17:56:44 +0000 [info]: #0 shutting down output plugin type=:file plugin_id="object:974"
2022-06-20 17:56:44 +0000 [info]: #0 shutting down output plugin type=:stdout plugin_id="object:9b0"
2022-06-20 17:56:45 +0000 [info]: #0 restart fluentd worker worker=0
2022-06-20 17:56:45 +0000 [info]: #0 listening port port=24300 bind="0.0.0.0"
2022-06-20 17:56:45 +0000 [info]: #0 listening port port=24200 bind="0.0.0.0"
2022-06-20 17:56:45 +0000 [info]: #0 listening port port=24100 bind="0.0.0.0"
2022-06-20 17:56:45 +0000 [info]: #0 listening port port=24000 bind="0.0.0.0"
```

Confirm that you see `type=relabel` after Fluentd has started before proceeding. To test the new configuration, perform some activity on the WordPress and Guestbook frontend instances.

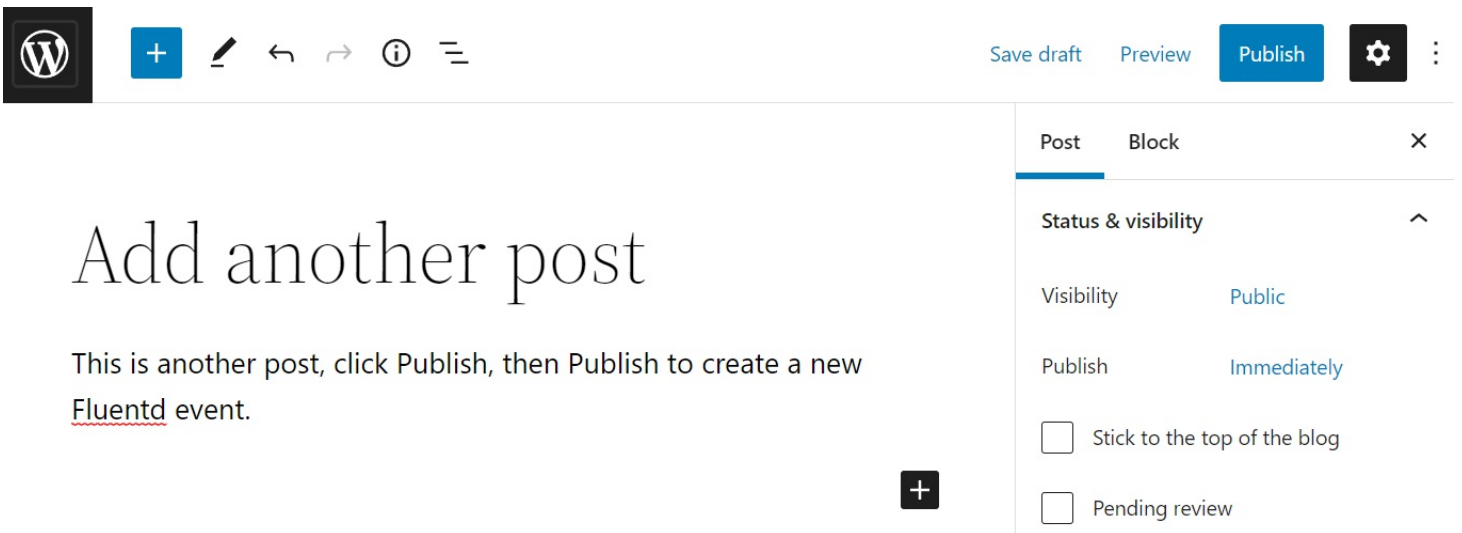
In WordPress, click the WordPress logo to return to the WordPress dashboard:



Click **+ New** in the top dashboard and click "Post":



Add a title, content, then publish a new post:



In the Guestbook, add an event by typing text in the box and clicking "Submit":

The next step needed activity!

Guestbook

Messages

Submit

I am a guestbook entry!

The next step needed activity!

Now flush the Fluentd buffers so ensure the events have been written.

```
ubuntu@labsys:~/lab6$ pkill -SIGUSR1 fluentd
```


Check the log directory, /tmp/lab6/ now:

```
ubuntu@labsys:~/lab6$ ls -l /tmp/lab6

total 164
drwxr-xr-x 2 ubuntu ubuntu 4096 Jun 20 17:58 database-log
-rw-r--r-- 1 ubuntu ubuntu 16458 Jun 20 17:58 database-log.202206201756_0.log
drwxr-xr-x 2 ubuntu ubuntu 4096 Jun 20 17:58 frontend-log
-rw-r--r-- 1 ubuntu ubuntu 9729 Jun 20 17:58 frontend-log.202206201757_0.log
-rw-r--r-- 1 ubuntu ubuntu 1743 Jun 20 17:58 frontend-log.202206201758_0.log
drwxr-xr-x 2 ubuntu ubuntu 4096 Jun 20 17:55 guestbook-log
-rw-r--r-- 1 ubuntu ubuntu 472 Jun 20 17:45 guestbook-log.202206201745_0.log
-rw-r--r-- 1 ubuntu ubuntu 1215 Jun 20 17:55 guestbook-log.202206201753_0.log
drwxr-xr-x 2 ubuntu ubuntu 4096 Jun 20 17:56 wordpress-log
-rw-r--r-- 1 ubuntu ubuntu 33197 Jun 20 17:45 wordpress-log.202206201743_0.log
-rw-r--r-- 1 ubuntu ubuntu 37301 Jun 20 17:45 wordpress-log.202206201744_0.log
-rw-r--r-- 1 ubuntu ubuntu 4732 Jun 20 17:45 wordpress-log.202206201745_0.log
-rw-r--r-- 1 ubuntu ubuntu 459 Jun 20 17:48 wordpress-log.202206201746_0.log
-rw-r--r-- 1 ubuntu ubuntu 459 Jun 20 17:50 wordpress-log.202206201748_0.log
-rw-r--r-- 1 ubuntu ubuntu 459 Jun 20 17:52 wordpress-log.202206201750_0.log
-rw-r--r-- 1 ubuntu ubuntu 459 Jun 20 17:54 wordpress-log.202206201752_0.log
-rw-r--r-- 1 ubuntu ubuntu 459 Jun 20 17:56 wordpress-log.202206201754_0.log

ubuntu@labsys:~/lab6$
```

There should now be `database-log` and `frontend-log` directories, which is where the buffers for those logs will be placed before they are written to actual log files.

Inspect one of the resulting log files (try using the latest one):

```
ubuntu@labsys:~/lab6$ tail -2 /tmp/lab6/frontend-log.202206201758_0.log

...

2022-06-20T17:58:47+00:00      guestbook
{"container_id":"92afd9985ebb087d956c42fd7207d67f451e3851dea45e7fdfb50f3717c6ee89","container_name":
"/guestbook","source":"stdout","log":"99.149.248.237 - - [20/Jun/2022:17:58:47 +0000] \"GET
/guestbook.php?
cmd=set&key=messages&value=,I%20am%20a%20guestbook%20entry!,The%20next%20step%20needed%20activity!
HTTP/1.1\" 200 255 \"http://3.237.191.229:49165/\" \"Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.0.0 Safari/537.36\""}
2022-06-20T17:58:56+00:00      wordpress
{"container_id":"439b40dd421f1322072553c44871824813b1d2331365cf80195d1e779a79e926","container_name":
"/wordpress","source":"stdout","log":"99.149.248.237 - - [20/Jun/2022:17:58:56 +0000] \"POST /wp-
admin/admin-ajax.php HTTP/1.1\" 200 563 \"http://3.237.191.229:49163/wp-admin/post-new.php\"
\"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.0.0
Safari/537.36\""}
2022-06-20T17:58:56+00:00      wordpress
{"container_name":"/wordpress","source":"stdout","log":"99.149.248.237 - - [20/Jun/2022:17:58:56
+0000] \"POST /wp-json/wp/v2/posts/8/autosaves?_locale=user HTTP/1.1\" 200 1518
\"http://3.237.191.229:49163/wp-admin/post-new.php\" \"Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.0.0
Safari/537.36\"","container_id":"439b40dd421f1322072553c44871824813b1d2331365cf80195d1e779a79e926"}

ubuntu@labsys:~/lab6$
```

```
ubuntu@labsys:~/lab6$ tail -5 /tmp/lab6/database-log.202206201756_0.log

...
```

```

2022-06-20T17:56:58+00:00      guestbook-db
{"container_id":"43cfc54a187d86384b591cefc0cdad66e258e33a59d7fe3a7d3a9400cfb656db","container_name":
"/guestbook-db","source":"stdout","log":"[1] 20 Jun 17:56:58.056 * The server is now ready to accept
connections on port 6379"}
2022-06-20T17:56:58+00:00      wordpress-db      {"log":"2022-06-20 17:56:58 0 [Note] InnoDB: 128
rollback segments are
active.", "container_id":"b9be61b57968443cd3edfc2708eb58b36147ca57d7ff216d8475baa9b44c6bd6", "containe
r_name":"/wordpress-db", "source":"stderr"}

...

ubuntu@labsys:~/lab6$

```

Looks like those are the latest logs being written to - no other logs have been written to either the previous guestbook-log or wordpress-log files from the previous configuration. The combination of using labels and relabeling allow the user to fully influence the way events are routed in addition to setting tags.

Try the following:

- Reconfigure the database-log output to send events in msgpack
 - Add an event to the Guestbook to trigger activity here
- Using a filter, try to print only the log portion of events being sent to the frontend-log file
 - Interact with either the WordPress or guestbook frontends to generate events

3. Unifying separate configuration files using Include statements

@include statements can be used to load directives from separate configuration files. In this step, the Fluentd configuration made in this lab will be divided into separate files. By doing this, the known-good configurations listed in them can be reused at a later time. It will also greatly simplify the process of reconfiguring Fluentd, allowing new configuration files to be loaded without having to terminate the instance.

Create a file containing all of the sources:

```

ubuntu@labsys:~/lab6$ nano sources.conf && cat $_

<source>
  # WordPress Database
  @type forward
  port 24000
  @label wordpress
</source>

<source>
  # WordPress
  @type forward
  port 24100
  @label wordpress
</source>

<source>
  # Guestbook Database
  @type forward
  port 24200
  @label guestbook
</source>

<source>
  # Guestbook

```

```
@type forward
port 24300
@label guestbook
</source>

ubuntu@labsys:~/lab6$
```

Note that the source file is still utilizing labels, so one of the configuration files in the intended chain of source files needs to include matching `<label>` directives for each of those sources.

Create different configuration files based on each label:

```
ubuntu@labsys:~/lab6$ nano wordpress.conf && cat $_

<label wordpress>
  <match wordpress>
    @type relabel
    @label frontend
  </match>
  <match wordpress-db>
    @type relabel
    @label db
  </match>
</label>

ubuntu@labsys:~/lab6$
```

```
ubuntu@labsys:~/lab6$ nano guestbook.conf && cat $_

<label guestbook>
  <match guestbook>
    @type relabel
    @label frontend
  </match>
  <match guestbook-db>
    @type relabel
    @label db
  </match>
</label>

ubuntu@labsys:~/lab6$
```

In a previous step, it was shown that Fluentd can be started with configuration files that lack complete pipelines as long as all syntax requirements are met.

It is time to unify each of the files created above.

Modify the original configuration file to use the `@include` parameter to load all of these configuration files together:

```
ubuntu@labsys:~/lab6$ cp lab6.conf lab6-1.conf

ubuntu@labsys:~/lab6$ nano lab6.conf && cat $_

@include /home/ubuntu/lab6/sources.conf

@include /home/ubuntu/lab6/wordpress.conf
@include /home/ubuntu/lab6/guestbook.conf
```

```

<label frontend>
  <match **>
    @type file
    path /tmp/lab6/frontend-log
    <buffer>
      timekey 60s
      timekey_wait 1m
    </buffer>
  </match>
</label>

<label db>
  <match **>
    @type file
    path /tmp/lab6/database-log
    <buffer>
      timekey 60s
      timekey_wait 1m
    </buffer>
  </match>
</label>

ubuntu@labsys:~/lab6$

```

Before proceeding with the reconfiguration, answer the following questions:

- Why are a pair of `<label>` directives still inside the "master" configuration file?
- If those `<label>` directives were removed, would Fluentd still correctly reconfigure? Why or why not?

Send a `SIGUSR2` to Fluentd to have it reload the configuration file:

```

ubuntu@labsys:~/lab6$ pkill -SIGUSR2 fluentd

ubuntu@labsys:~/lab6$

```

```

ubuntu@labsys:~/lab6$ fluentd -c lab6.conf

2022-06-20 18:07:18 +0000 [info]: Reloading new config
2022-06-20 18:07:18 +0000 [info]: using configuration file: <ROOT>
<source>
  @type forward
  port 24000
  @label wordpress
</source>
<source>
  @type forward
  port 24100
  @label wordpress
</source>
<source>
  @type forward
  port 24200
  @label guestbook
</source>
<source>
  @type forward
  port 24300
  @label guestbook
</source>

```

```

<label wordpress>
  <match wordpress>
    @type relabel
    @label frontend
  </match>
  <match wordpress-db>
    @type relabel
    @label db
  </match>
</label>
<label guestbook>
  <match guestbook>
    @type relabel
    @label frontend
  </match>
  <match guestbook-db>
    @type relabel
    @label db
  </match>
</label>
<label frontend>
  <match **>
    @type file
    path "/tmp/lab6/frontend-log"
    <buffer>
      timekey 60s
      timekey_wait 1m
      path "/tmp/lab6/frontend-log"
    </buffer>
  </match>
</label>
<label db>
  <match **>
    @type file
    path "/tmp/lab6/database-log"
    <buffer>
      timekey 60s
      timekey_wait 1m
      path "/tmp/lab6/database-log"
    </buffer>
  </match>
</label>
</ROOT>

...

```

When Fluentd parses the @include parameters in a configuration file, it will load all of the directives into memory at runtime.

Restart the wordpress-db and guestbook-db containers to force them to reconnect to Fluentd:

```

ubuntu@labsys:~/lab6$ sudo docker restart guestbook-db wordpress-db

guestbook-db
wordpress-db

ubuntu@labsys:~/lab6$

```

Add a new blog post to WordPress and another guestbook entry, then flush the logs:

```

ubuntu@labsys:~/lab6$ pkill -sigusr1 fluentd

```

```
ubuntu@labsys:~/lab6$ tail -2 /tmp/lab6/database-log.202206201807_0.log

2022-06-20T18:07:59+00:00      wordpress-db      {"source":"stderr","log":"Version: '10.7.4-MariaDB-1:10.7.4+maria~focal' socket: '/run/mysqld/mysqld.sock' port: 3306 mariadb.org binary distribution","container_id":"b9be61b57968443cd3edfc2708eb58b36147ca57d7ff216d8475baa9b44c6bd6","container_name":"/wordpress-db"}
2022-06-20T18:07:59+00:00      wordpress-db      {"container_id":"b9be61b57968443cd3edfc2708eb58b36147ca57d7ff216d8475baa9b44c6bd6","container_name":"/wordpress-db","source":"stderr","log":"2022-06-20 18:07:59 0 [Note] InnoDB: Buffer pool(s) load completed at 220620 18:07:59"}

ubuntu@labsys:~/lab6$ tail -1 /tmp/lab6/frontend-log.202206201808_0.log

2022-06-20T18:08:56+00:00      wordpress
{"container_id":"439b40dd421f1322072553c44871824813b1d2331365cf80195d1e779a79e926","container_name":"/wordpress","source":"stdout","log":"99.149.248.237 - - [20/Jun/2022:18:08:56 +0000] \"GET /wp-json/wp/v2/templates/twentytwentytwo//single?context=edit&_locale=user HTTP/1.1\" 200 3880 \"http://3.237.191.229:49163/wp-admin/post.php?post=10&action=edit\" \"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.0.0 Safari/537.36\""}

ubuntu@labsys:~/lab6$
```

This setup, which is made of separate files imported by `@include` parameter, should function identically to the previous configuration. By separating configurations into many separate files, easier troubleshooting and configuration modularity can be achieved.

The `@include` parameter can also be used against web URLs, so anything uploaded to a network registry or even GitHub can be reused if the Fluentd instance attempting to use those files can communicate with the remote server or host. Incorporating separate, established Fluentd configurations with the `@include` parameter makes the process of maintaining and reusing them much easier.

Now that you have worked with both labels and include in this lab, try to perform more complex configurations:

- Try using a wildcard `*` to simplify the primary Fluentd configuration file

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:~/lab6$ sudo docker container rm \
$(sudo docker container stop wordpress-db wordpress guestbook-db guestbook)

wordpress-db
wordpress
guestbook-db
guestbook

ubuntu@labsys:~/lab6$
```

Use `CTRL C` to terminate any locally running instances of Fluentd:

```
^C
...

2022-06-20 18:11:00 +0000 [info]: Worker 0 finished with status 0

ubuntu@labsys:~/lab6$ cd
```

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

Congratulations, you have completed the lab!