Course Guide

# IBM Operations Analytics Log Analysis: Scalable Collection Architecture

Course code TN631 ERC 1.0

IBM Training

## August 2016 edition

# Contents

# About this course

IBM Training

## IBM Operations Analytics Log Analysis: Scalable Collection Architecture

For enterprise-level environments, IBM Operations Analytics Log Analysis needs a reliable and scalable architecture for data collection. This course teaches you how to install and configure all of the components required for a scalable log collection architecture. In this 3-day course, you learn through lecture and hands-on exercises.

To collect log data reliably, IBM Operations Analytics Log Analysis uses several software components to ensure that log messages are not lost. During this course, you learn how to install and configure the following software:

- Filebeat, to collect log messages
- IBM Tivoli Log File Agent, to collect log messages
- HAProxy, a load balancer
- Logstash, an event pipeline
- Kafka, a message queue system
- IBM Operations Analytics Log Analysis, to index log messages and present a user search interface

The lab environment for this course uses the VMware platform.

| | Details |
|---|---|
| **Delivery method** | Classroom or instructor-led online (ILO) |
| **Course level** | ERC 1.0 |
| | This course is a new course. |
| **Product and version** | IBM Operations Analytics Log Analysis version 1.3.3.1 |
| **Recommended duration** | 3 days |
| **Skill level** | Intermediate |

IBM Training

### Course objectives

- Diagram the Scalable collection architecture
- Install and configure Filebeat
- Install and configure HAProxy
- Install and configure Logstash receivers
- Install and configure Kafka
- Configure Logstash senders
- Configure Log Analysis data sources
- Install and configure the Log File Agent
- Support multi-line logs
- Consolidate physical data sources
- Troubleshoot the Scalable collection architecture




*Course objectives*

# Audience

This course is designed for implementers, administrators, technical sales persons, and any others who need IBM Operations Analytics  Log Analysis skills.

# Prerequisites

Before taking this course, make sure that you have Linux administration skills and a working knowledge of IBM Operations Analytics  Log Analysis.

IBM Training IBM

## Agenda

- Unit 1 Overview
- Unit 2 Filebeat, HAProxy, and Logstash receivers
- Unit 3 Kafka
- Unit 4 Logstash senders and Log Analysis core
- Unit 5 Using the Log File Agent
- Unit 6 Multi-line logs
- Unit 7 Log consolidation
- Unit 8 Troubleshooting




*Agenda*

# Course description

The course contains the following units:

1. Overview

   This unit explains the scalable collection design. You also learn about each of the software components that are used for scalable collection.

   This unit has no student exercises.

2. Filebeat, HAProxy, and Logstash receivers

   This unit teaches you how to install and configure Filebeat, HAProxy, and Logstash servers.

   In the exercises for this unit, you install and configure a log collection agent, the HAProxy load balancer, and a Logstash receiver cluster. You configure these components to monitor a web access log from an IBM HTTP Server (IHS).

3. Kafka

   This unit teaches you how to install and configure Kafka.

   Apache Kafka is a high-throughput distributed messaging system. In this scalable collection architecture, Kafka acts as a durable message store with fault tolerance. Kafka requires Apache Zookeeper for coordination between brokers. In these exercises, you install Kafka and Zookeeper. You then configure the Logstash receiver cluster to send messages to Kafka and

verify the data flow. Finally, you install Kafka Manager, which allows you to view your configurations and data that Kafka is processing.

4. Logstash senders and Log Analysis core

   In this unit, you learn how to install and configure Logstash senders. You also learn how to configure the IBM Operations Analytics Log Analysis core software to accept data from a Logstash sender.

   Logstash senders pull messages from Kafka, then send them to the Log Analysis core software. They can also alter the messages before they send them. In these exercises, you configure two Logstash sender instances and verify that they are moving data from Kafka to Log Analysis.

5. Using the Log File Agent

   You can use the IBM Tivoli Log File Agent, which is included with Log Analysis, as a collection agent. This unit teaches you how to install and configure the Log File Agent.

   You can use the Log File Agent (LFA) as a collection agent to capture messages from a target log file. Earlier in this course, you installed and configured Filebeat as a collection agent. In these exercises, you install and configure a second type of log collector: the Log File Agent. You then update the other components in your lab environment to process messages from the Log File Agent.

6. Multiline logs

   In this unit, you learn how to configure Logstash receivers, Logstash senders, and HAProxy load balancers to process multiline logs.

   Many log files are in a multiline format. Multiline log records are single log messages that span multiple lines. To process multiline logs in the scalable collection architecture, you must ensure that all of the lines in a single message are processed together, and in the correct sequence. In the exercises for this unit, you add support for a multiline log file.

7. Log consolidation

   This unit teaches you how to consolidate messages from many logs into just a few Log Analysis data sources.

   In a simple configuration, IBM Log Analysis manages input from a single target log file with a single data source. Environments with many log files to monitor (hundreds or thousands) require just as many data sources. It can be problematic for administrators to create, manage, navigate, and support so many data sources. In the exercises for this unit, you alter your lab environment to consolidate several web server logs into a single Log Analysis data source.

8. Troubleshooting

   In this unit, you learn how to troubleshoot each of the components in the scalable collection architecture.

   This unit has no student exercises.

# *Unit 1* Overview


This unit explains the scalable collection design. You also learn about each of the software components that are used for scalable collection.

IBM Training

## Unit objectives

- Describe the scalable collection architecture
- List the components in the scalable collection architecture




*Unit objectives*

IBM Training

## Scalable collection architecture

In a large environment, the method you use to collect log messages must be reliable and durable.

- The collection method must handle ingestion across hundreds and thousands of end points in a load-balanced and fault-tolerant manner.
- The collection method must be reliable to handle spiky log traffic, for example, several megabytes per second.
- The collection method must prevent data loss when downstream Log Analysis components are slower.
- The collection method should avoid dependence on only the IBM Log File Agent (LFA) for log shipping. Support for additional collection agents such as syslog and Filebeat should be included.



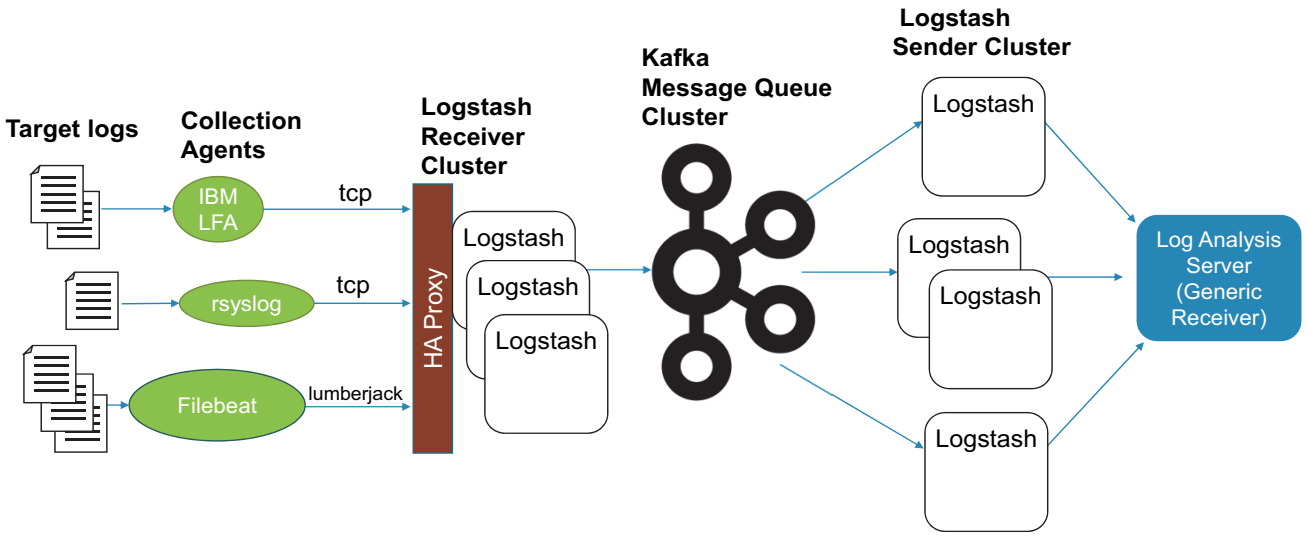

*Scalable collection architecture*

In large IT environments, the volume of log messages can overwhelm a log management solution. To avoid losing any log messages, the collection method that is used to collect logs must be durable, reliable, and resilient.

This course teaches you how to implement a reliable and scalable collection architecture for IBM Operations Analytics Log Analysis.

IBM Training

## Architecture components




Overview



*Architecture components*

This diagram shows the flow of data through the software components that are used to reliably collect log messages.

1. Collection agents such as the Log File Agent and Filebeat collect log messages from target log files and send them to an HAProxy load balancer.
2. The HAProxy load balancer forwards log messages to a cluster of Logstash receiver instances.
3. Logstash receivers perform minimal processing on log messages and send them to the Kafka cluster.
4. The Kafka cluster acts as a durable data store of log messages. After messages arrive in Kafka, the messages are stored on disk.
5. Logstash senders retrieve messages from the Kafka cluster, process them to add or transform the data in the messages and send them to the Log Analysis core software.
6. The Log Analysis core software indexes the interesting text within each log message and presents a search interface to users.

IBM Training

## Collection agents

- **Log File Agent**: Reads log messages from target files and sends data to HAProxy with TCP.
- **Filebeat**: Reads log messages from target files and sends data to HAProxy with lumberjack, a proprietary protocol.
- **Syslog and rsyslog**: Send data to HAProxy with TCP.

For downstream processing, collection agents should add metadata to log messages.




*Collection agents*

Collection agents read messages from target log files and send them to an HAProxy load balancer.

The Log File Agent and Filebeat are included with IBM Log Analysis. Collection agents must be able to add metadata to each log message that they send because the other components in the collection design use that metadata for processing and conditional logic.

IBM Training

## Logstash

- Logstash acts as a log shipping agent or as centralized log server in a standalone or hierarchical architecture
- As a shipping agent, it generates events from input plug-ins, modifies them through filters, and sends them elsewhere based on output plug-ins
- Logstash receivers get messages from the collection agents and send them to Kafka
- Logstash senders pull messages from Kafka, alter the messages if necessary, and send the messages to the Log Analysis server


Overview



*Logstash*

Logstash is open source software that accepts text as input, processes text with filters, and sends text in a specific output format. In this scalable architecture, Logstash servers are used as receivers and senders:

- Receiver Logstash servers accept log messages from collection agents and store them in the Kafka cluster.
- Sender Logstash servers retrieve log messages from the Kafka message queue and send them to the Log Analysis core software.

IBM Training

## HAProxy

- HAProxy is an open source load balancer and proxy server for TCP and HTTP-based applications that spreads requests across multiple servers.
- HAProxy has a reputation for being fast and efficient.
- HAProxy makes nonredundant services redundant.
- Health check
  - HAProxy uses health checks to determine if a back-end server is available to process requests.
  - This avoids having to manually remove a server from the back end if it becomes unavailable.
  - The default health check is to try to establish a TCP connection to the server to verify if the back-end server is listening on the configured IP address and port.
- Sticky sessions
  - Some applications require stickiness between a client and a server: it means all the requests from a client must be sent to the same server
  - Using a dedicated load balancing algorithm:  A hash on the source IP

Overview



*HAProxy*

HAProxy is a software load balancer. In the Log Analysis scalable architecture, its role is to provide redundancy for Logstash receivers.

For Logstash receivers that process single-line messages, HAProxy sends data to each Logstash receiver in turns. For Logstash receivers that process multiline messages, HAProxy sends data to the same Logstash receiver, if the Logstash receiver is running.

IBM Training

## Kafka

Kafka is a persistent, distributed, replicated publish–subscribe messaging system.

- Kafka is fast, scalable, and durable:
  - Fast: A single Kafka broker can handle hundreds of megabytes of reads and writes per second from thousands of clients.
  - Scalable: With Kafka, a single cluster can serve as the central data backbone for a large organization. It can be transparently expanded without downtime.
  - Durable: Messages are persisted on disk and replicated within the cluster to prevent data loss.
- Logstash receivers write data to Kafka topics for various data sources and Logstash senders read from Kafka.
- The basic unit of concurrency in Kafka is a ***partition***.
  A frequent practice is to create a topic for each logical data source and partitions within that topic for each physical data source.




*Kafka*

Kafka is an open source message queue system. In the Log Analysis scalable architecture, its role is to provide reliable storage for log messages.

Kafka is a distributed system. Kafka servers that store data are called ***brokers***.

Apache Zookeeper is used to coordinate communication between different Kafka brokers. When you install Kafka, the Zookeeper software is automatically installed also.

IBM Training

## Software version information

| Component | Version |
|---|---|
| IBM Operations Analytics Log Analysis | 1.3.3 Fix Pack 1 |
| Kafka | 0.8.2.2 |
| Logstash | 2.2.1 |
| HAProxy | 1.5.4 |




*Software version information*

This table lists the version of each software component that is supported with IBM Operations Analytics Log Analysis v1.3.3.1.

IBM Training

## Resources included with Log Analysis

- These directories contain installation files:
  - `<LA_HOME>/logstash-2.2.1`
  - `<LA_HOME>/filebeat`
  - `<LA_HOME>/kafka`
- You can install Logstash and the Log File Agent with the remote installation tool:
  - `<LA_HOME>/remote_install_tool`
- You can find sample configuration files for the following components in the `<LA_HOME>/kafka/test-configs` directory:
  - Filebeat
  - HAProxy
  - Kafka
  - Zookeeper
  - Log File Agent
  - Logstash receivers
  - Logstash senders

Overview



*Resources included with Log Analysis*

When you install IBM Operations Analytics Log Analysis, the following open source software installation files are included:

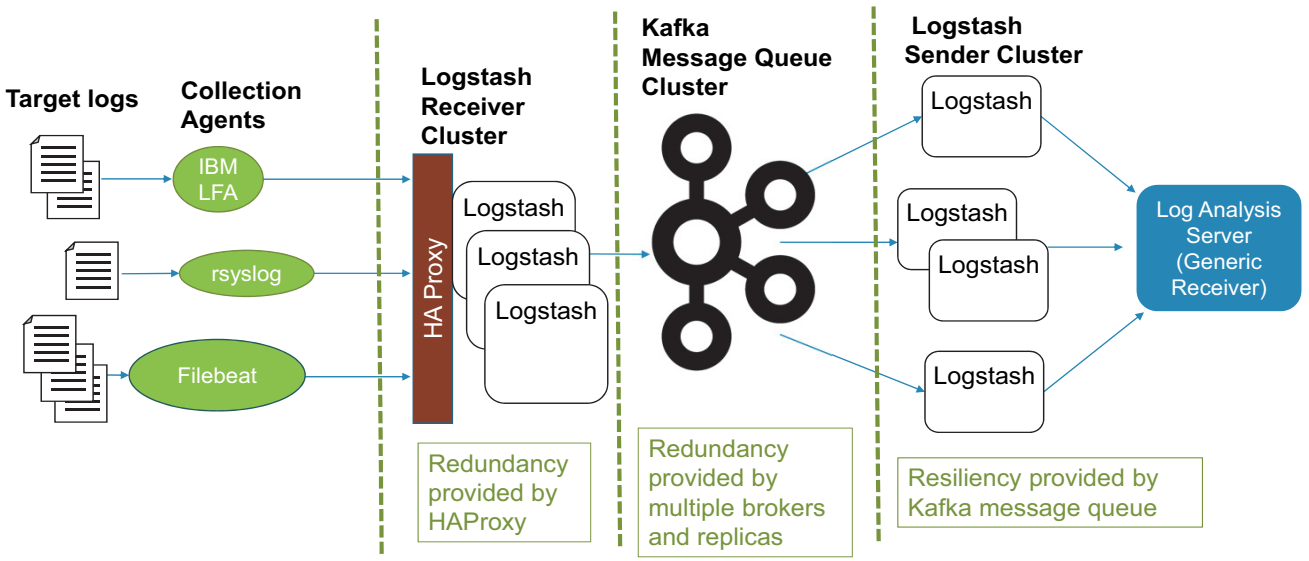
- Logstash
- Filebeat
- Kafka

Sample configuration files for the following components are included when you install IBM Operations Analytics Log Analysis:

- Filebeat
- HAProxy
- Kafka
- Zookeeper
- Log File Agent
- Logstash receivers
- Logstash senders

IBM Training

## Resiliency and redundancy




Overview



*Resiliency and redundancy*

This diagram shows how the scalable collection architecture delivers redundancy and resiliency.

HAProxy ensures that no log messages are lost by adding redundancy to a cluster of Logstash receivers. If one Logstash receiver is down, HAProxy sends log messages to a different Logstash receiver server in the cluster.

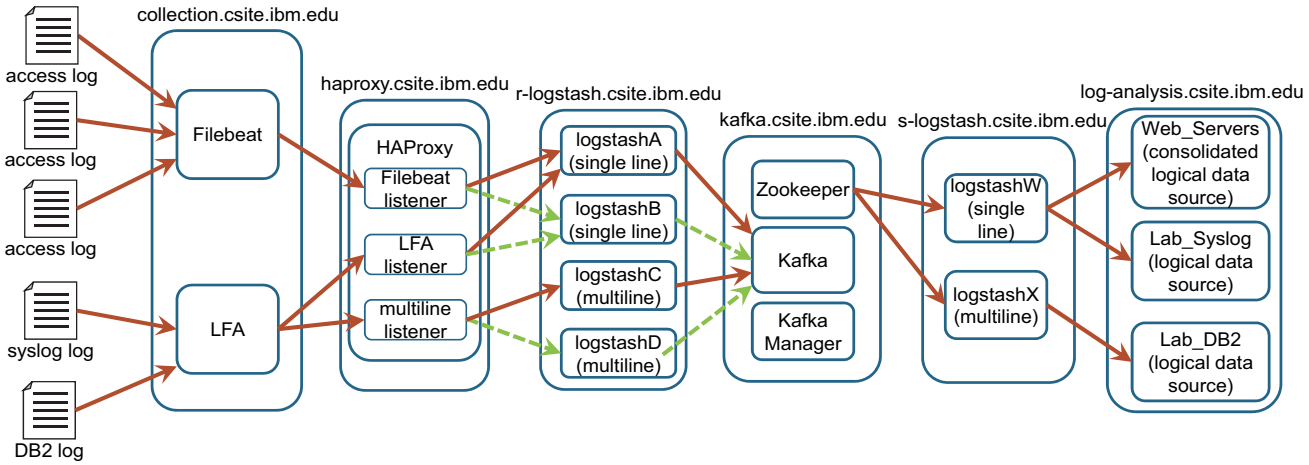
The Kafka cluster provides reliable storage for log messages. If one broker in the Kafka cluster is down, messages can still be read from other brokers in the cluster.

Resiliency for Logstash senders is provided by the Kafka cluster. If a Logstash sender server goes down, it can read messages that are stored in the Kafka message queue when it comes back up.

IBM Training

## About the lab exercises

- Completing the exercises is vital to understanding the scalable collection architecture
- When you are finished with the lab exercises, your environment looks like this:




*About the lab exercises*

In the student exercises for this course, you build an environment for scalable log message collection. You install and configure several software components and connect them together. You also verify data flow between the components.

The student exercises for this course teach you how each component works and the role of each component in the scalable collection design.

IBM Training

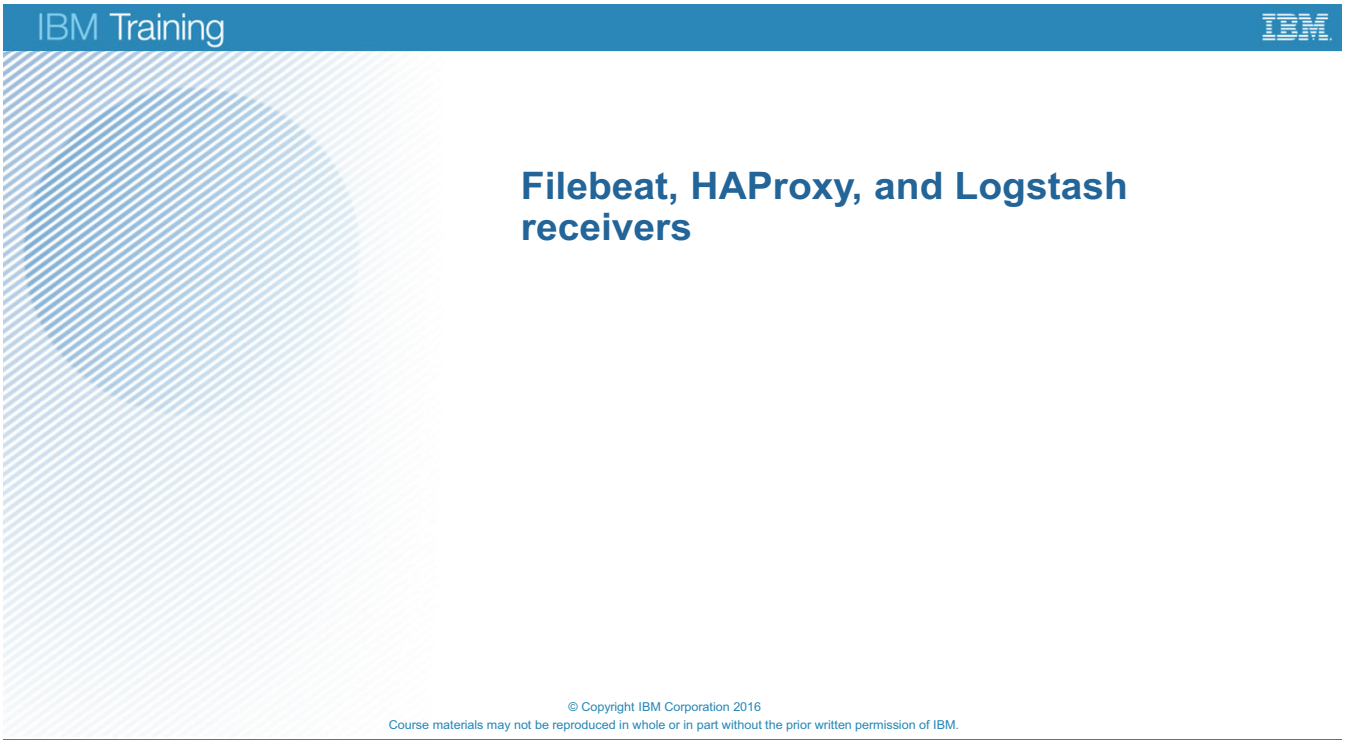
## Unit summary

- Describe the scalable collection architecture
- List the components in the scalable collection architecture




*Unit summary*

# *Unit 2* Filebeat, HAProxy, and Logstash receivers




This unit teaches you how to install and configure Filebeat, HAProxy, and Logstash servers.

IBM Training

## Unit objectives

- Install and configure Filebeat
- Install and configure HAProxy
- Install and configure Logstash receivers




*Unit objectives*

IBM Training

## Installing and configuring Filebeat

- Install Filebeat by decompressing the installation file.
- Configure Filebeat with the `filebeat.yml` file.

`filebeat.yml`

```
filebeat:
   prospectors:
   -
      paths:
       - /opt/IBM/HTTPServer/logs/access.log
      input_type: log
      fields:
         collector: filebeats-collection.csite.ibm.edu
         env: DEV
         module: IBM-HTTP-Server
         type: access-log
         site: DALLAS
         platform: RHEL

output:
   logstash:
      hosts: ["ha-proxy.csite.ibm.edu:20737"]

shipper:

logging:
   to_files: true
   files:
      path: /opt/filebeat-1.1.1-x86_64/
      name: mybeat
      rotateeverybytes: 10485760
   level: info
```




*Installing and configuring Filebeat*

To install a Filebeat server, decompress the installation files. To configure Filebeat, you edit a file named `filebeat.yml`.

In this example, Filebeat is monitoring a log file named `/opt/IBM/HTTPServer/logs/access.log`.

In this example, Filebeat adds the following metadata to each log message:

- collector
- env
- module
- type
- site
- platform

In this example, Filebeat is sending log messages to a host named **ha-proxy.csite.ibm.edu** on port 20737. The messages are sent in a format that Logstash servers can interpret.

In this example, Filebeat is logging to a file named `/opt/filebeat-1.1.1-x86_64/mybeat`.

**Hint:** The indentations in the `filebeat.yml` file are very important. This file is indented with spaces, not tabs.

IBM Training

## Other Filebeat options

- Include or exclude lines from a file
- Exclude files
- Scan frequency
- Multiline
- File as output
- Transport Layer Security




*Other Filebeat options*

You can also configure advanced options for Filebeat servers. The following list describes other Filebeat options:

- You can configure Filebeat to included or exclude individual lines from a log file based on a regular expression match.
- You can exclude files in a directory from monitoring.
- You can increase or decrease the interval that Filebeat checks for new files.
- You can control how Filebeat deals with log messages that span multiple lines.
- You can send log messages that Filebeat collects to a file. This option is useful for troubleshooting.
- You can configure secure transport between Filebeat and Logstash.

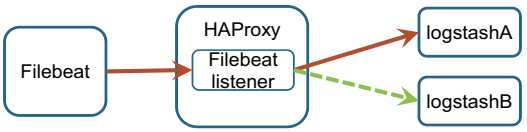
IBM Training

## Installing and configuring HAProxy

There are two options to install HAProxy:

- Install with a package

  `yum install haproxy`
- Install from source




/etc/haproxy/haproxy.cfg

```
...
defaults
    mode                    tcp
    log                     global
    option                  tcplog
    option                  dontlognull
...
## Listener for Filebeat Logstash cluster
## Specify Logstash instances for processing data from Filebeat
listen  Beats_Cluster ha-proxy.csite.ibm.edu:20737
    mode tcp
         balance roundrobin
         server receiver-logstashA logstashA.csite.ibm.edu:18737 check fall 2 rise 3 inter 1000
         server receiver-logstashB logstashB.csite.ibm.edu:18738 check fall 2 rise 3 inter 1000
```




*Installing and configuring HAProxy*

You can install HAProxy as an operating system package, or you can compile it from the source code.

You configure HAProxy by editing a file named `/etc/haproxy/haproxy.cfg`. In this scalable architecture, HAProxy must run in tcp mode.

You must configure a listener for HAProxy to accept data from a log collection agent and forward it to a Logstash receiver cluster. In this example, HAProxy is listening on port 20737. Any data HAProxy receives on port 20737 is forwarded to either `logstashA.csite.ibm.edu:18737` or `logstashB.csite.ibm.edu:18738`.

HAProxy is aware of the state of each server. In this example, HAProxy performs health checks on each Logstash server every 1000 milliseconds. HAProxy considers a Logstash server down if two consecutive health check failures occur (`fall 2`). HAProxy considers a Logstash server up again after three consecutive successful health checks (`rise 3`).

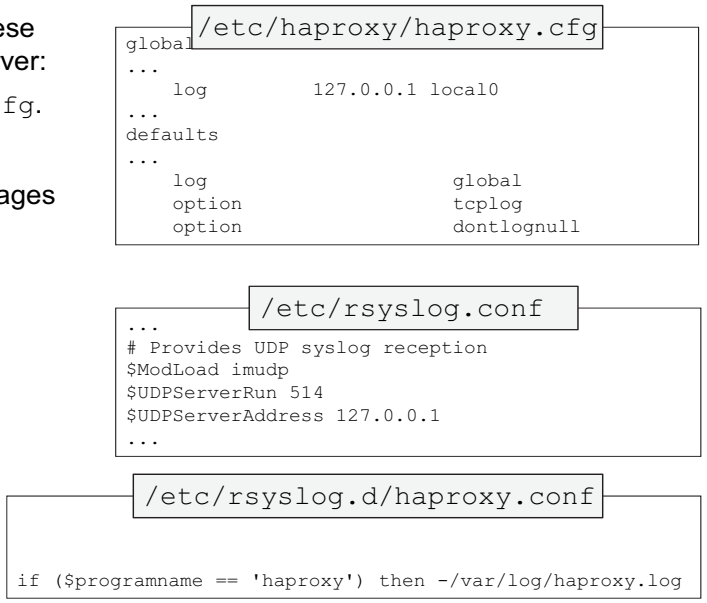
The line `balance roundrobin` configures HAProxy to send data to each Logstash server in turns.

IBM Training

## Configure HAProxy logging

HAProxy logs to syslog or rsyslog. Follow these steps to enable logging for your HAProxy server:

1. Configure logging options in `haproxy.cfg`.
2. Configure rsyslog to receive UDP data.
3. Configure rsyslog to send haproxy messages to a log file.
4. Restart HAProxy
5. Restart rsyslog.




Filebeat, HAProxy, and Logstash receivers

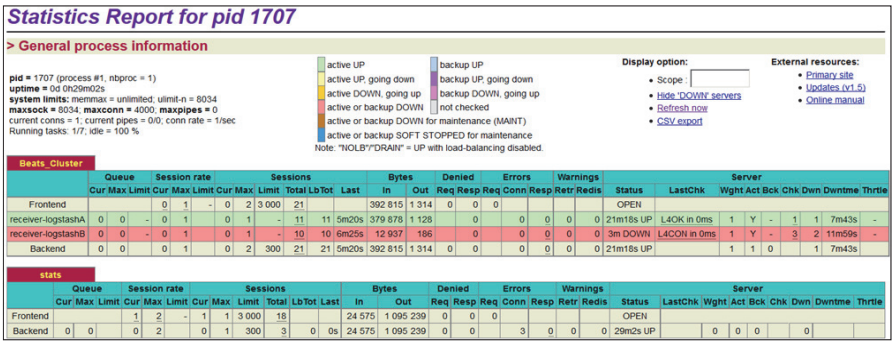


*Configure HAProxy logging*

HAProxy logs to syslog or rsyslog. To enable logging, you must configure HAProxy and your syslog or rsyslog server. In this example, HAProxy is logging to a local rsyslog server.

IBM Training

## HAProxy statistics page




```
 stats socket /var/lib/haproxy/stats
...

listen stats :9000 #Listen on localhost port 9000
    mode http
    stats enable
    stats hide-version
    stats realm Haproxy\ Statistics
    stats uri /haproxy_stats
    stats auth netcool:object00
```

/etc/haproxy/haproxy.cfg




*HAProxy statistics page*

You can configure HAProxy to display a web page that shows the status of each server that it is forwarding data to. The page also shows statistics about each of the endpoints in the cluster, such as the volume of data sent. You configure this web page in the `/etc/haproxy/haproxy.cfg` file.

This slide shows an example of the lines you add to the `/etc/haproxy/haproxy.cfg` file to enable the HAProxy statistics page.

The following table describes each line in the configuration example.

| Configuration option | Description |
|---|---|
| listen stats :9000 | This line configures the statistics page to use port 9000. |
| mode http | This line enables http mode. |
| stats enable | This line enables statistics. |
| stats hide-version | This line hides the version of HAProxy so it is not visible on the web page. |
| stats realm Haproxy\ Statistics | This text is what users see when they are prompted to log in to the page. |
| stats uri /haproxy_stats | The URI of the web page, for example, `http://localhost:9000/haproxy_stats` |
| stats auth netcool:object00 | This line enables authentication. It also sets a user name and password. In this example, the user name is `netcool` and the password is `object00`. |

IBM Training

## Logstash defined

- Logstash acts as a log shipping agent or as centralized log server in a stand-alone or hierarchical architecture.
- As a shipping agent, it generates events from input plug-ins, modifies them through filters, and sends them elsewhere based on output plug-ins.





*Logstash defined*

Logstash is open source software that accepts text as input, process text with filters, and sends text in a specific output format. In this scalable architecture, Logstash servers are used as receivers and senders.

Receiver Logstash servers accept log messages from collection agents and store them in the Kafka cluster.

Logstash uses plug-ins for inputs, filters, and outputs.

IBM Training

## Key Logstash functions

- Logstash is an event pipeline:
  - Inputs → filters/codecs → outputs
  - Inputs capture events.
  - Codecs and filters modify them.
  - Outputs send them somewhere.
- Inputs can declare a type for each stream (for example, type=WAS)
  - Inputs are used mainly for filter activation.
  - They persist with the event.
- Tags can be added throughout event processing:
  - Use for debugging purposes
  - Can be used to specify an order for event processing
- Conditional logic allows events to be processed in many ways.




*Key Logstash functions*

Logstash processes messages in an event pipeline. Each log message is treated as an individual event.

Logstash accepts log messages with input plug-ins. After Logstash receives a message, it can alter the text within that message with filter plug-ins. These filters can modify the message itself, or it can add and alter metadata that arrived with the log message.

Logstash sends the text in log messages to a destination with output plug-ins.

At any step in Logstash processing, you can add tags to the original log message. You can use these tags, along with metadata in the log message, to add conditional logic to the text processing pipeline. For example, you can alter the time stamp for one type of log message, while the time stamp for a different type of log message remains unchanged.

IBM Training

## Included Logstash plug-ins





*Included Logstash plug-ins*

This slide is a partial list of Logstash plug-ins of inputs, filters, and outputs. Codecs are also listed on this slide. As open source software, plug-ins are published and maintained by a community. If necessary, you can create your own Logstash plug-ins.

IBM Training

## Important plug-ins for Log Analysis

- Inputs
  - **file** reads text in a file.
  - **tcp** is used for accepting text over a TCP socket.
  - **kafka** is used to retrieve data from Kafka.
  - **beats** is used to accept messages from Filebeat.
- Filters
  - **drop** deletes text that is not useful.
  - **date** converts the time stamp.
  - **grok** annotates data in unstructured text.
  - **mutate** adds, modifies, or deletes fields and the data in those fields.
- Outputs
  - **file** used for testing and troubleshooting.
  - **kafka** sends data to a Kafka broker.
  - **scala** publishes data in the format that Log Analysis expects.




*Important plug-ins for Log Analysis*

This slide lists Logstash plug-ins that you use for Logstash receiver and Logstash sender servers. The following list provides examples of how you might use these plug-ins.

- Inputs:
  - Logstash receivers accept data from collection agents. Use the **tcp** input plug-in to accept data from a Log File Agent. Use the **beats** input plug-in to accept data from a Filebeat server.
  - Logstash senders pull data from a Kafka cluster. Use the *kafka* input plug-in to retrieve data from a Kafka cluster.
- Filters:
  - Logstash receivers must construct a topic ID for the kafka output. Use the *mutate* filter plug-in to build a topic ID from metadata that is attached to each log message.
  - Logstash senders must send a value for host and path to the Log Analysis server. Use the *mutate* filter plug-in to build values for host and path from metadata that is attached to each log message.
- Outputs:
  - Logstash receivers send messages to a Kafka cluster. Use the *kafka* output plug-in to send data to a list of Kafka brokers.
  - Logstash senders send messages to the Log Analysis server. Use the *scala* output plug-in to send data to Log Analysis in the correct format.
  - Use the **file** output plug-in to troubleshoot Logstash receivers and senders.

IBM Training

## Installing Logstash

There are three options to install Logstash:

- Decompress installation files (included with Log Analysis)
- Install with Log Analysis remote installer tool (included with Log Analysis)
- Install from RPM




*Installing Logstash*

You can install Logstash with the methods listed on this slide.

## *Installing Logstash by decompressing the installation file*

If you install Logstash by decompressing the installation file, you must start and stop manually. The user who decompressed the installation file owns Logstash. You can create the Logstash configuration file anywhere the owner of Logstash has read and write permissions.

## *Installing Logstash with the Log Analysis remote installer tool (included with Log Analysis)*

Use the following steps to install Logstash with the Log Analysis remote installer tool:

1. Change to the `<LA_HOME>/remote_install_tool` directory
2. Edit the `ssh-config.properties` file in the `config` subdirectory
3. Start the remote installation tool with the command `install.sh` and follow the prompts to install Logstash.

After you install Logstash with the Log Analysis remote installer tool, your Logstash instance has the following attributes:

- You must start and stop Logstash with the logstash-util.sh utility.

  `<LS_DIR>/utilities/logstash-util.sh`

- The configuration file is in the following directory:.

  `<LS_DIR>/Logstash/logstash-2.2.1/logstash-scala/logstash/config/logstash-scala.conf`

- Log files are in the following directories.

  `/opt/logstashA/Logstash/logs/ls-console.log`
  `/opt/logstashA/Logstash/logs/scala_logstash.log`

## *Installing Logstash from an RPM*

Use the following steps to install Logstash from an RPM Package Manager file (RPM):

1. Switch to the root user.
2. Run the following command to install Logstash.

   `rpm -i logstash-2.2.1-1.noarch.rpm`

After you install Logstash from an RPM, your Logstash instance has the following attributes:

- Logstash is installed in the `/opt/logstash/` directory
- A user and a group named logstash is created
- You must start and stop Logstash with a script in the `/etc/init.d` directory

  `/etc/init.d/logstash`
- Your configuration file must be in the following directory:

  `/etc/logstash/conf.d/`
- Log files are in the /`var/log/logstash/` directory.

IBM Training

## Logstash configuration

Logstash is configured with a `.conf` file.

- Logstash will not start without a valid configuration file.
- The configuration file must contain the following sections:
  - input
  - filter
  - output
- After you change the configuration, you must restart Logstash.




*Logstash configuration*

You configure Logstash with a text file. The configuration file name must have the extension .conf.

A Logstash configuration file must have an input section, a filter section, and an output section. You learn how to create a configuration file during this course.

IBM Training

## About Logstash receivers

- Logstash receivers accept text from collection agents such as Filebeat or Log File Agent.
- Logstash receivers perform minimal processing on each message.
- Logstash receivers send messages to Kafka brokers.




*About Logstash receivers*

In the Log Analysis scalable architecture, the role of a Logstash receiver instance is to accept log messages from collection agents and send the messages to Kafka. Multiple Logstash receivers provide redundancy: if one Logstash receiver is down, other Logstash receivers can continue to accept data.

Logstash receivers should perform only minimal processing and alteration of each log message, so they can quickly send messages to Kafka.

IBM Training

## Example of Logstash receiver configuration

```
input {
    beats {
             port => 18737
    } #end beats input
} #end input section

filter {
    if [fields][collector] == "filebeats-collection.csite.ibm.edu" {
        mutate {
          add_field => [ "datasource", "%{[fields][env]}_%{[fields][module]}_%{[fields][type]}" ]
          add_field => [ "resourceID", "%{[beat][hostname]}_%{source}_1" ]
          add_tag => "mutate_filebeat"
        }# end mutate
    } #end filebeat condition
} #end filter section

output {
    if ("mutate_filebeat" in [tags]) and ! ("_grokparsefailure" in [tags]) {
        kafka {
            bootstrap_servers =>"kafka.csite.ibm.edu:17991"
            topic_id => "%{datasource}"
            message_key => "%{resourceID}"
        } #end Kafka output
    } #end Kafka condition
} # end output section
```

Filebeat, HAProxy, and Logstash receivers



*Example of Logstash receiver configuration*

In this example configuration, a Logstash receiver server is accepting data from a Filebeat server with the *beats* input plug-in. The Logstash receiver adds metadata to each message with the *mutate* filter plug-in. Finally, the Logstash receiver sends log messages to a Kafka broker with the *kafka* output plug-in.

### Key fields in the input section

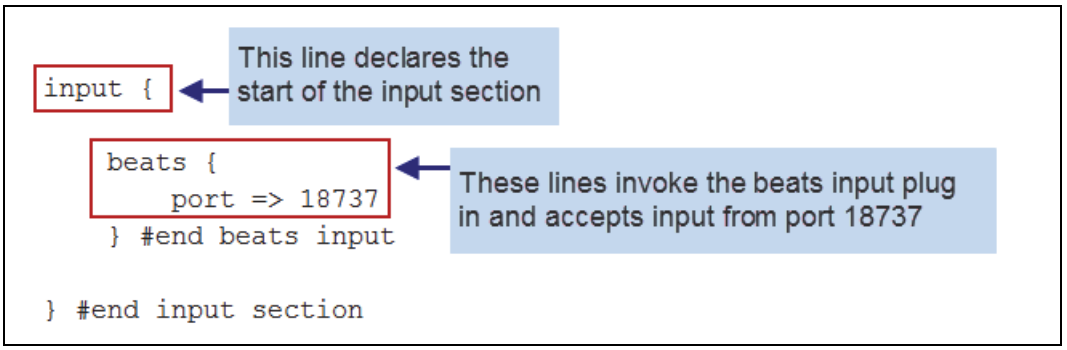



*Figure 1 Key fields in the input section*

## Key fields in the filter section

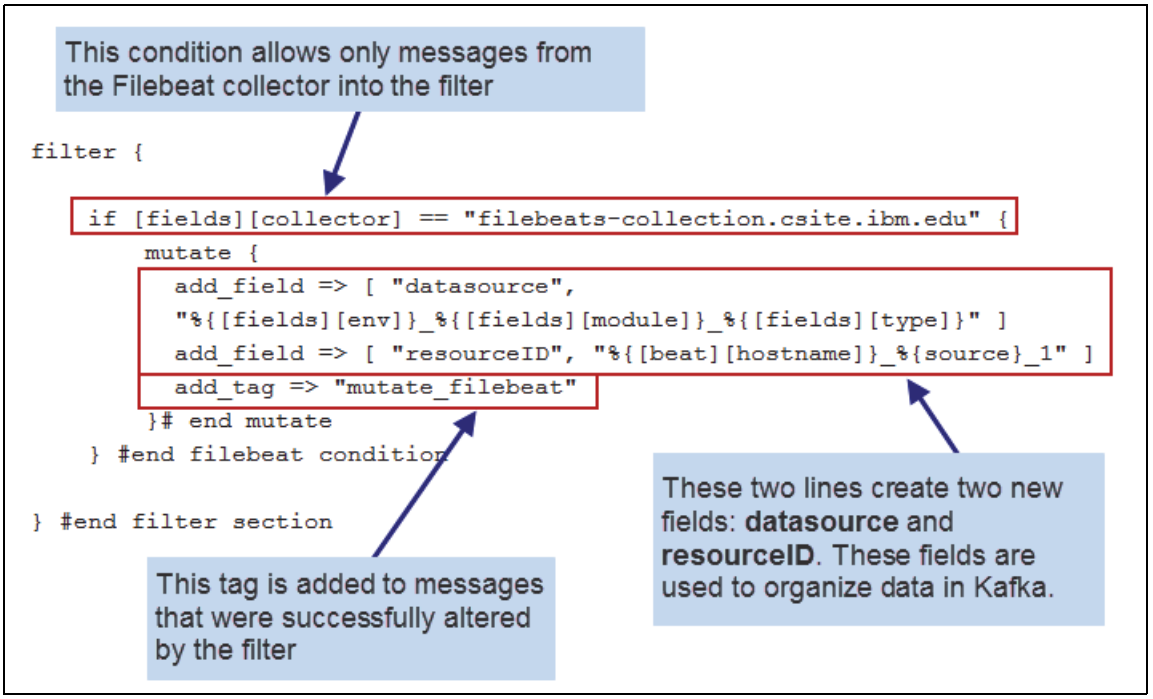



*Figure 2 Key fields in the filter section*

## Key fields in the output section

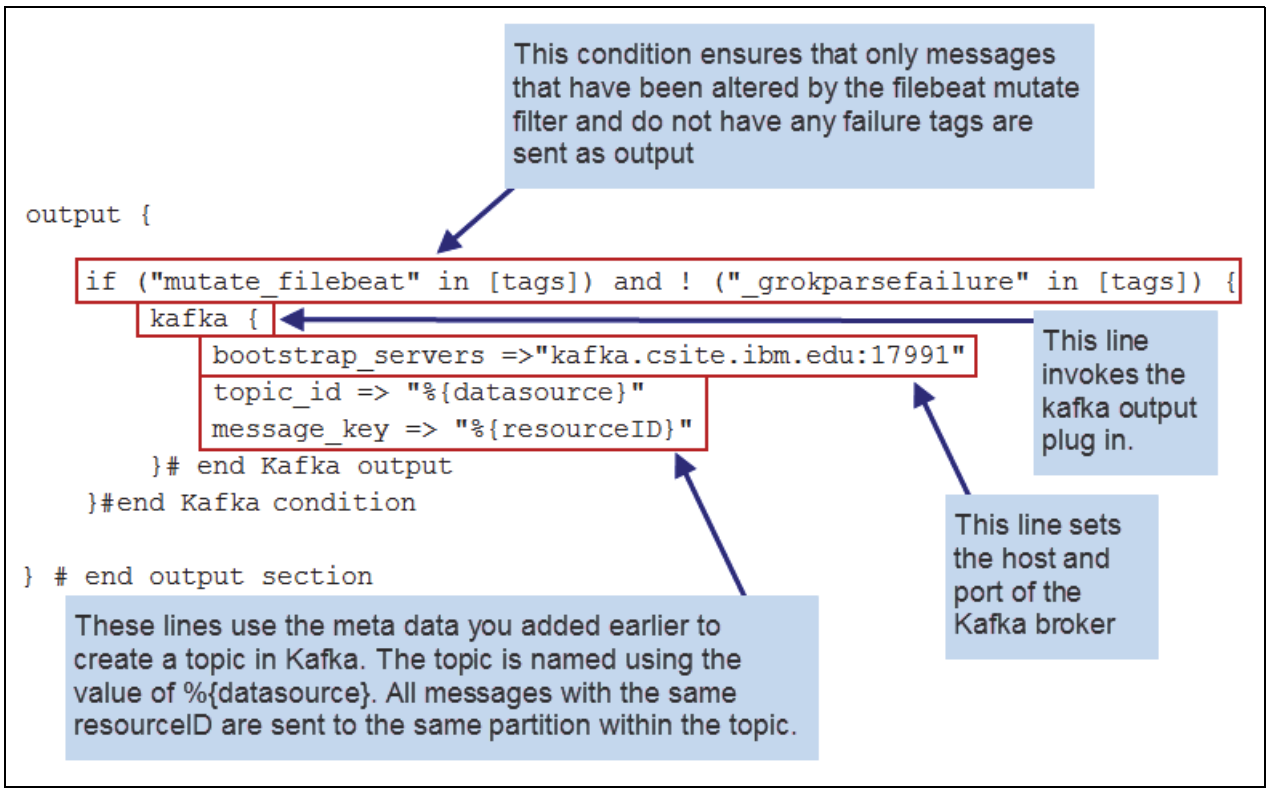



*Figure 3 Key fields in the output section*

IBM Training

## Metadata for Logstash receivers

- Logstash receivers create topics in Kafka
- The filter section of a Logstash receiver uses data from the collection agent to set the Kafka topic ID and message key
- In this example:
  - The topic ID is **DEV_IBM-HTTP-Server_access-log**
  - The message key is **collection.csite.ibm.edu_/opt/IBM/HTTPServer/logs/access.log_1**

```
filebeat:
   prospectors:
   -
      paths:
       - /opt/IBM/HTTPServer/logs/access.log
      input_type: log
      fields:
         collector: filebeats-collection.csite.ibm.edu
         env: DEV
         module: IBM-HTTP-Server
         type: access-log
         site: DALLAS
         platform: RHEL
...
```

filebeat.yml

```
...
filter {
    if [fields][collector] == "filebeats-collection.csite.ibm.edu" {
        mutate {
          add_field => [ "datasource", "%{[fields][env]}_%{[fields][module]}_%{[fields][type]}" ]
          add_field => [ "resourceID", "%{[beat][hostname]}_%{source}_1" ]
          add_tag => "mutate_filebeat"
        }# end mutate
    } #end filebeat condition
} #end filter section

output {
    if ("mutate_filebeat" in [tags]) and ! ("_grokparsefailure" in [tags]) {
        kafka {
            bootstrap_servers =>"kafka.csite.ibm.edu:17991"
            topic_id => "%{datasource}"
            message_key => "%{resourceID}"
        } #end Kafka output
    } #end Kafka condition
} # end output section
```

logstashreceiver.conf

Filebeat, HAProxy, and Logstash receivers



*Metadata for Logstash receivers*

In the example on this slide, the Logstash receiver is using metadata that was added to each log message by the Filebeat collection agent.

The mutate filter adds a field named **datasource**. The value of **datasource** is constructed by using the value of the Filebeat field **env**, followed by an underscore character, followed by the value of the Filebeat field module, followed by an underscore character, followed by the value of the Filebeat field **type**. In this example, the value of **datasource** is `DEV_IBM-HTTP-Server_access-log`.

The mutate filter also adds a field named **resourceID**. The value of resourceID is constructed by using the value of the Filebeat metadata keyword **hostname**, followed by an underscore character, followed by the value of the metadata keyword `source`, followed by the string `_1`. In this example, the value of **resourceID** is `collection.csite.ibm.edu_/opt/IBM/HTTPServer/logs/access.log_1`.

The two fields that were created by the mutate filter, **datasource** and **resourceID**, are used in the configuration of the kafka output plug-in.

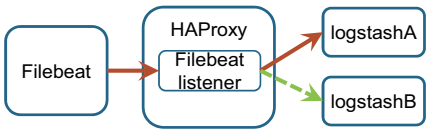
- The value of **datasource** is the name of the Kafka topic that this Logstash receiver writes messages to. Notice that the value of **datasource** is used as the **topic_id**.
- Any messages with the same **message_key** are written to the same partition within the topic. Notice that the value of **resourceID** is used as the **message_key**.

IBM Training

## Logstash receiver redundancy

filebeat.yml

```
...
output:
   logstash:
      hosts: ["ha-proxy.csite.ibm.edu:20737"]
...
```




haproxy.cfg

```
listen  Beats_Cluster ha-proxy.csite.ibm.edu:20737
    mode tcp
         balance roundrobin
         server receiver-logstashA logstashA.csite.ibm.edu:18737 check fall 2 rise 3 inter 1000
         server receiver-logstashB logstashB.csite.ibm.edu:18738 check fall 2 rise 3 inter 1000
```

logstashReceiverA.conf

```
input {
    beats {
             port => 18737
    } #end beats input
} #end input section
...
output {
        kafka {
            bootstrap_servers =>"kafka.csite.ibm.edu:17991"
            topic_id => "%{datasource}"
            message_key => "%{resourceID}"
        } #end Kafka output
} # end output section
```

logstashReceiverB.conf

```
input {
    beats {
             port => 18738
    } #end beats input
} #end input section
...
output {
        kafka {
            bootstrap_servers =>"kafka.csite.ibm.edu:17991"
            topic_id => "%{datasource}"
            message_key => "%{resourceID}"
        } #end Kafka output
} # end output section
```




*Logstash receiver redundancy*

In a scalable collection environment, you should install multiple Logstash receiver servers. You use HAProxy to make your Logstash receiver servers run as a cluster for failover and redundancy.

In this example, the Filebeat collection agent sends messages from the target log file to ha-proxy.csite.ibm.edu:20737.

HAProxy listens for traffic on port 20737. HAProxy forwards traffic from the Filebeat server on port 20737 to either logstashA.csite.ibm.edu:18737 or logstashB.csite.ibm.edu:18738.

Both Logstash receiver servers have identical configurations for the kafka output plug-in. Both Logstash receiver servers send data to the same list of Kafka brokers (kafka.csite.ibm.edu:17991). This means if one Logstash receiver is down, HAProxy sends data to the other Logstash receiver, and the data is still written to the Kafka cluster.

IBM Training

## Unit summary

- Install and configure Filebeat
- Install and configure HAProxy
- Install and configure Logstash receivers




*Unit summary*

IBM Training

# Filebeat, HAProxy, and Logstash receivers exercises

## Exercise 1 Installing and configuring Filebeat
## Exercise 2 Installing and configuring HAProxy
## Exercise 3 Installing and configuring the first Logstash receiver
## Exercise 4 Verifying data flow
## Exercise 5 Adding metadata to the log message
## Exercise 6 Installing and configuring the second Logstash receiver

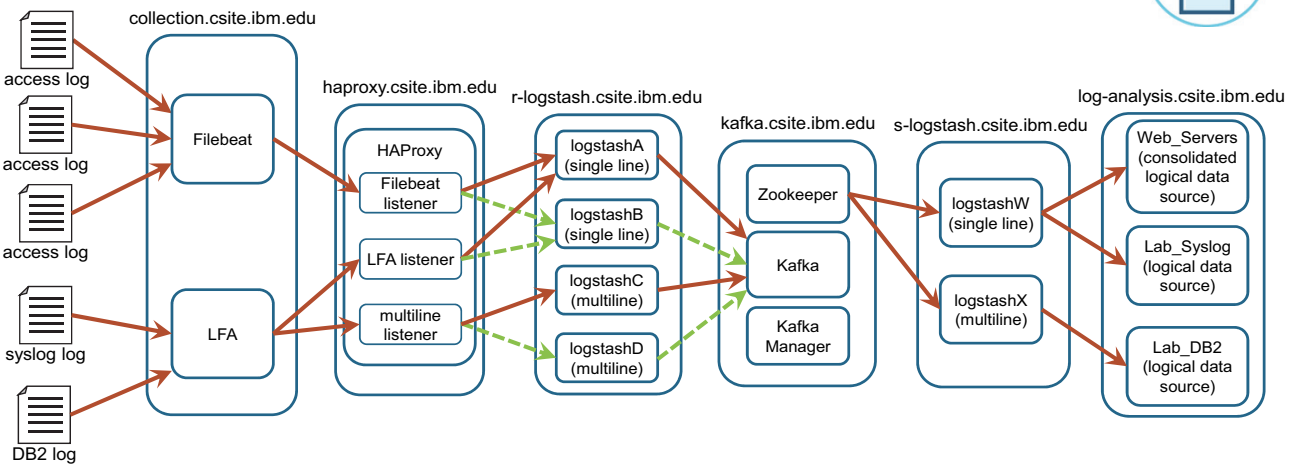



*Student exercises*

Perform the exercises for this unit.

IBM Training

## Exercise introduction: Finished lab environment





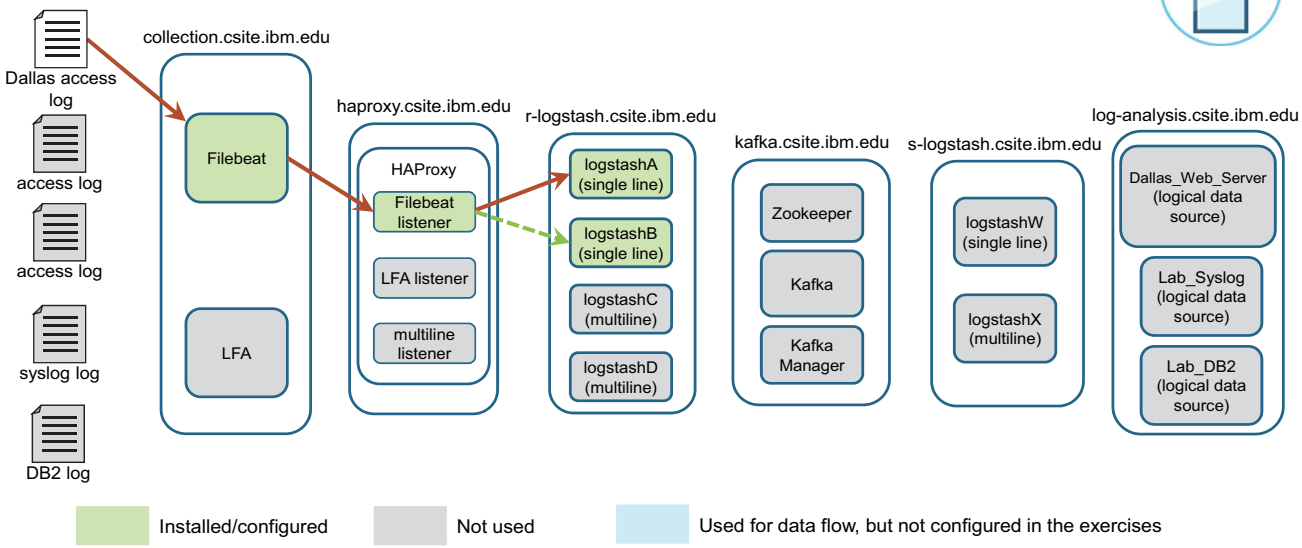


*Exercise introduction: Finished lab environment*

At the end of this course, your lab environment will look like this diagram. Refer to this diagram as you complete the steps in the student exercises to help you understand what you are installing and configuring.

IBM Training

## Exercise introduction: Lab environment in Unit 2 exercises







*Exercise introduction: Lab environment in Unit 2 exercises*

In the exercises for this unit, you install and configure the following components:

- A Filebeat server, to monitor a web server access log
- An HAProxy load balancer, with a listener for traffic from the Filebeat server
- Two Logstash receiver servers, to accept messages from the Filebeat server

As you install and configure these components, you also verify the data flow between them.

# *Unit 3* Kafka


This unit teaches you how to install and configure Kafka.

IBM Training

## Unit objectives

- Install and configure Kafka
- Diagram data flow from a Logstash receiver server to Kafka
- Install and configure Kafka Manager




*Unit objectives*

IBM Training

## Kafka overview

- Kafka is a distributed, high-throughput message broker.
- All Kafka messages are organized into ***topics*.**
- To send a message to Kafka, you send it to a specific topic.
- To read a message from Kafka, you read it from a specific topic.
- ***Producers*** send messages to Kafka, for example, a receiver Logstash server.
- ***Consumers*** pull messages from Kafka, for example, a sender Logstash server.
- Topics are divided ***partitions***. Partitions allow for messages within each topic to be read in parallel.
- Data is stored in transaction logs. Producers append to logs sequentially. Consumers read a range from these logs using an offset.
- Kafka, as a distributed system, runs in a cluster. Each node in the cluster is called a ***broker***.
- Kafka requires Zookeeper for coordination.




*Kafka overview*

In the Log Analysis scalable architecture, the role of the Kafka cluster is a reliable and durable data store for log messages.

Kafka is a distributed system. Each Kafka server that stores data is called a broker. When a Logstash receiver sends output to Kafka, it sends data to a list of brokers.

Apache Zookeeper is used in the Kafka cluster to coordinate data replication and communication between brokers.

Applications that send data to Kafka are called producers. In the Log Analysis scalable architecture, the producers are Logstash receivers.

Applications that pull data out of Kafka are called consumers. In the Log Analysis scalable architecture, the producers are Logstash senders.

The following diagram shows the relationship between brokers, topics, and partitions.

- There are two brokers in this example: Broker 0 and Broker 1.
- There are two topics in this example: ABC and XYZ. Topics are the message queues. Topics are logical collections of physical files. Partitions are the physical files where data is saved.
- The data in each topic is saved across three partitions. Partitions are the physical files where data is saved.

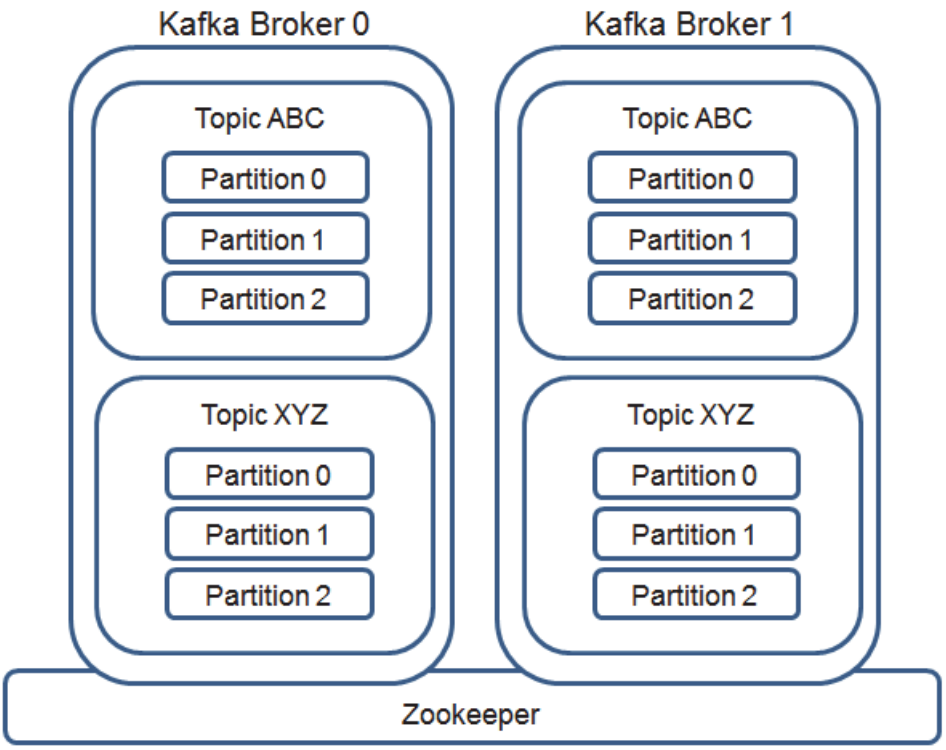



*Figure 1 Brokers, topics, and partitions*

IBM Training IBM

## Installing and configuring Kafka

- Install Kafka by decompressing the installation file.
- Zookeeper installs with Kafka.
- Configure Zookeeper with the `zookeeper.properties` file.
- Configure Kafka with the `server.properties` file.

zookeeper.properties

```
dataDir=/tmp/zookeeper
clientPort=17981
maxClientCnxns=0
```

server.properties

```
broker.id=0
port=17991
log.dirs=/tmp/kafka-logs-server-0
num.partitions=3
zookeeper.connect=kafka.csite.ibm.edu:17981
```




*Installing and configuring Kafka*

Decompress the installation file to install a Kafka broker and Zookeeper.

Edit the zookeeper.properties file to configure the Zookeeper server. The following list explains some key properties in the zookeeper.properties file:

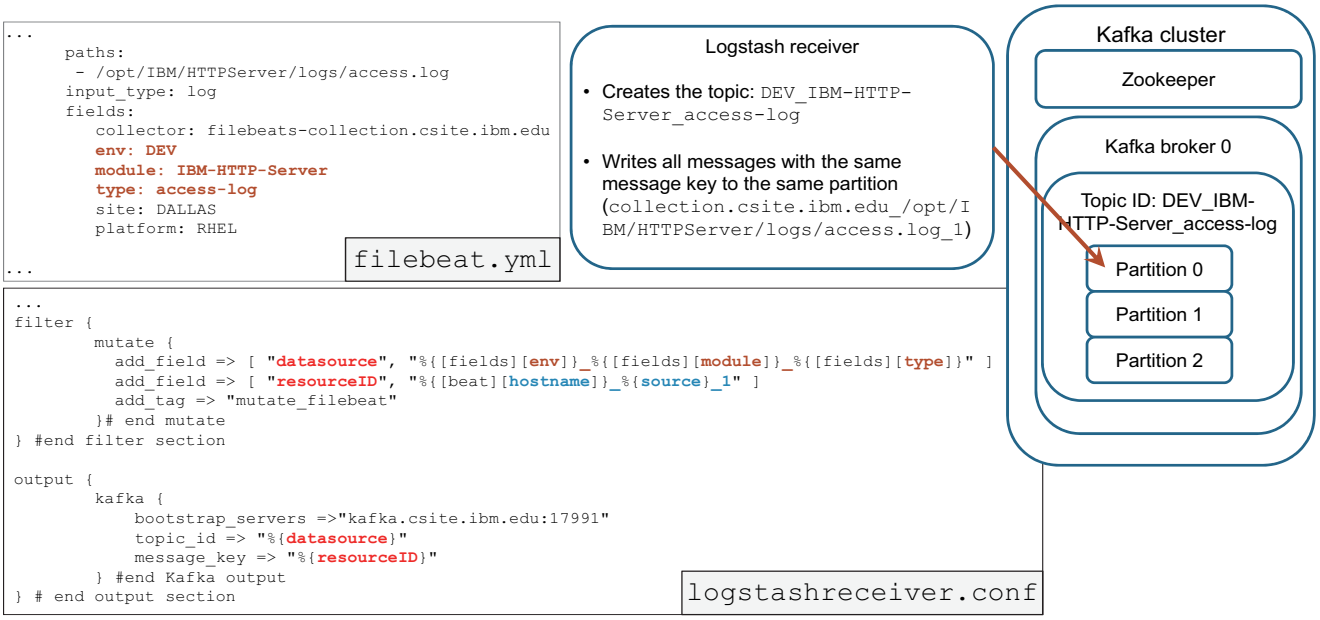
- **dataDir**: The directory where Zookeeper saves state information
- **clientPort**: The port number that Kafka brokers and Logstash senders use to connect to the Zookeeper server
- **maxClientCnxns**: The number of concurrent connections that a client can make to Zookeeper

Edit the `server.properties` file to configure the Kafka broker. The following list explains some key properties in the `server.properties` file:

- **broker.id**: The unique identifier for the broker.
- **port**: The port used to connect to the broker.
- **log.dirs**: A comma-delimited list of directories where the broker saves data. Partitions are created in the directories you enter in this property.
- **num.partitions**: The number of partitions that the broker will create for each topic.
- **zookeeper.connect**: The host and port of the Zookeeper server.

IBM Training

## Kafka in the scalable collection architecture




Kafka



*Kafka in the scalable collection architecture*

In the example on this slide, the Logstash receiver is using metadata that was added to each log messages by the Filebeat collection agent.

The Logstash receiver mutate filter adds a new field named **datasource**. The value of `datasource` is constructed using the value of the Filebeat field **env**, followed by an underscore character, followed by the value of the Filebeat field **module**, followed by an underscore character, followed by the value of the Filebeat field **type**. In this example, the value of **datasource** is `DEV_IBM-HTTP-Server_access-log`.

The mutate filter also adds a new field named **resourceID**. The value of **resourceID** is constructed using the value of the Filebeat metadata keyword **hostname**, followed by an underscore character, followed by the value of the metadata keyword `source`, followed by the string `_1`. In this example, the value of **resourceID** is `collection.csite.ibm.edu_/opt/IBM/HTTPServer/logs/access.log_1`.

The two fields that were created by the mutate filter, **datasource** and **resourceID**, are used in the configuration of the kafka output plug-in.

- The value of **datasource** is the name of the Kafka topic that this Logstash receiver writes messages to. Notice that the value of **datasource** is used as the **topic_id**.
- Any messages with the same **message_key** are written to the same partition within the topic. Notice that the value of **resourceID** is used as the **message_key**.

The Logstash receiver kafka output plug-in is configured to send data to the Kafka broker at kafka.csite.ibm.edu:17991. The Logstash receiver is a Kafka producer. The Logstash receiver automatically creates a topic named `DEV_IBM-HTTP-Server_access-log` in the Kafka broker.

All data is saved to the same partition in the topic, because the **message_key** option is configured in the kafka output plug-in.

IBM Training

## Viewing data in Kafka

Kafka includes tools to list all topics and to show the data in each topic.

To list all topics, issue this command:

```
kafka-topics.sh --list --zookeeper kafka.csite.ibm.edu:17981
```

```
DEV_IBM-HTTP-Server_access-log
LabDB2Instance_db2diag
labInstance_syslog
```

To view data in a topic, issue this command:

```
kafka-console-consumer.sh --zookeeper kafka.csite.ibm.edu:17981 --topic DEV_IBM-HTTP-
Server_access-log --from-beginning
```

```
{"message":"Apache/IHS,192.168.2.94,-,-,[10/May/2016:15:41:37 -0400],GET,\"GET
/daytrader/app?action=portfolio HTTP/1.1\",200,3409,1308,\"-\",\"curl/7.21.3 (x86_64-unknown-
linux-gnu) libcurl/7.21.3 OpenSSL/1.0.0 zlib/1.2.3\"","@version":"1","@timestamp":
...
```




*Viewing data in Kafka*

You can use command-line tools to show data that is saved in the Kafka cluster. These tools are in the `<KAFKA_HOME>/bin` directory.

Use the `kafka-topics.sh` tool to list all topics in the Kafka cluster.

Use the `kafka-console-consumer.sh` tool to view the data in a topic. If you use the `--from-beginning` option, the tool shows all data in a topic. If you do not use the `--from-beginning` option, the tool shows only data that is currently being written to the topic.

You can also use the kafka-topics.sh tool to show details about a topic. In the following example, the tools shows information about the topic named DEV_IBM-HTTP-Server_access-log:

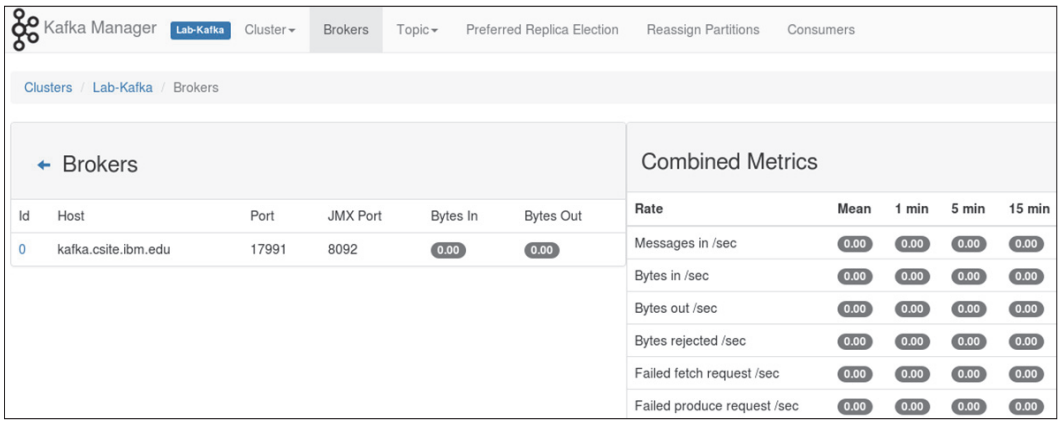
```
./kafka-topics.sh --describe --zookeeper kafka.csite.ibm.edu:17981 --topic
DEV_IBM-HTTP-Server_access-log

Topic:DEV_IBM-HTTP-Server_access-logPartitionCount:3ReplicationFactor:1Configs:
   Topic: DEV_IBM-HTTP-Server_access-logPartition: 0Leader: 0Replicas: 0Isr: 0
```

IBM Training

## Kafka Manager

Kafka Manager is an optional web-based user interface that manages your Kafka cluster.







*Kafka Manager*

Kafka Manager is a web-based user interface that shows the configuration of your cluster. You can also use Kafka Manager to view data and modify your cluster. Although Kafka Manager is not required, it can be useful for troubleshooting and maintenance.

IBM Training

## Installing and using Kafka Manager

Follow these steps to install Kafka Manager:

1. Create an installation package with Simple Build Tool (sbt).
2. Decompress the installation package to install Kafka Manager.

Kafka Manager uses Java Management Extensions (JMX) to connect to Kafka brokers.

You must set the JMX port before you start your Kafka broker, for example:

```
export JMX_PORT=8092
/opt/kafka_2.9.1-0.8.2.2/bin/kafka-server-start.sh -daemon /opt/kafka_2.9.1-0.8.2.2/config/server.properties
```



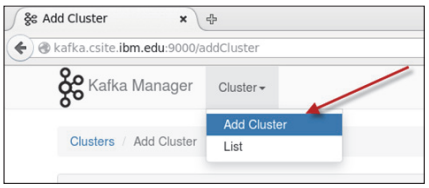

*Installing and using Kafka Manager*

To install Kafka Manager, you must compile it from source code with the Simple Build Tool. The Simple Build Tool creates an archive file that you use to install Kafka Manager.

Kafka Manager uses Java Management Extensions (JMX) to connect to Kafka brokers. If you want Kafka Manager to communicate with your Kafka cluster, you must set the JMX port before you start each of your Kafka brokers.

IBM Training

## Configuring Kafka Manager





Click **Cluster > Add Cluster** to configure Kafka Manager to monitor your Kafka cluster.




*Configuring Kafka Manager*

You must configure Kafka Manager before you can use it to manager your cluster. To add your cluster, open a browser and go to http://<*kafka-manager_host*>:9000. Click **Cluster > Add Cluster**. The following table explains how to configure the required fields on the Add Cluster page.

| Field | Usage |
|---|---|
| Cluster Name | A name for your cluster. This name is used only to display the cluster in Kafka Manager. |
| Cluster Zookeeper Hosts | A comma-delimited list of zookeeper hosts and port numbers. |
| Kafka Version | Select your version of Kafka, for example 0.8.2.2. |
| Enable JMX Polling | Set this to true. |
| brokerViewThreadPoolSize | Set this to a minimum of 2. |
| offsetCacheThreadPoolSize | Set this to a minimum of 2. |
| kafkaAdminClientThreadPoolSize | Set this to a minimum of 2. |

IBM Training

## Unit summary

- Install and configure Kafka
- Diagram data flow from a Logstash receiver server to Kafka
- Install and  configure Kafka Manager




*Unit summary*

IBM Training

## Kafka exercises

**Exercise 1 Installing and configuring Kafka**
**Exercise 2 Configuring Logstash receiver output**
**Exercise 3 Verifying data flow**
**Exercise 4 Kafka Manager**

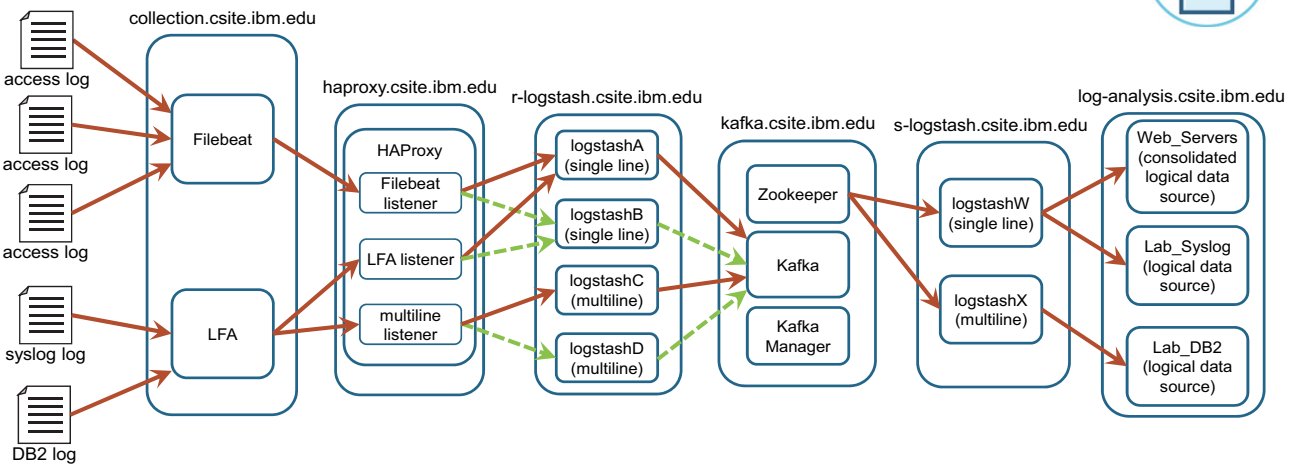



*Student exercises*

Perform the exercises for this unit.

IBM Training

## Exercise introduction: Finished lab environment

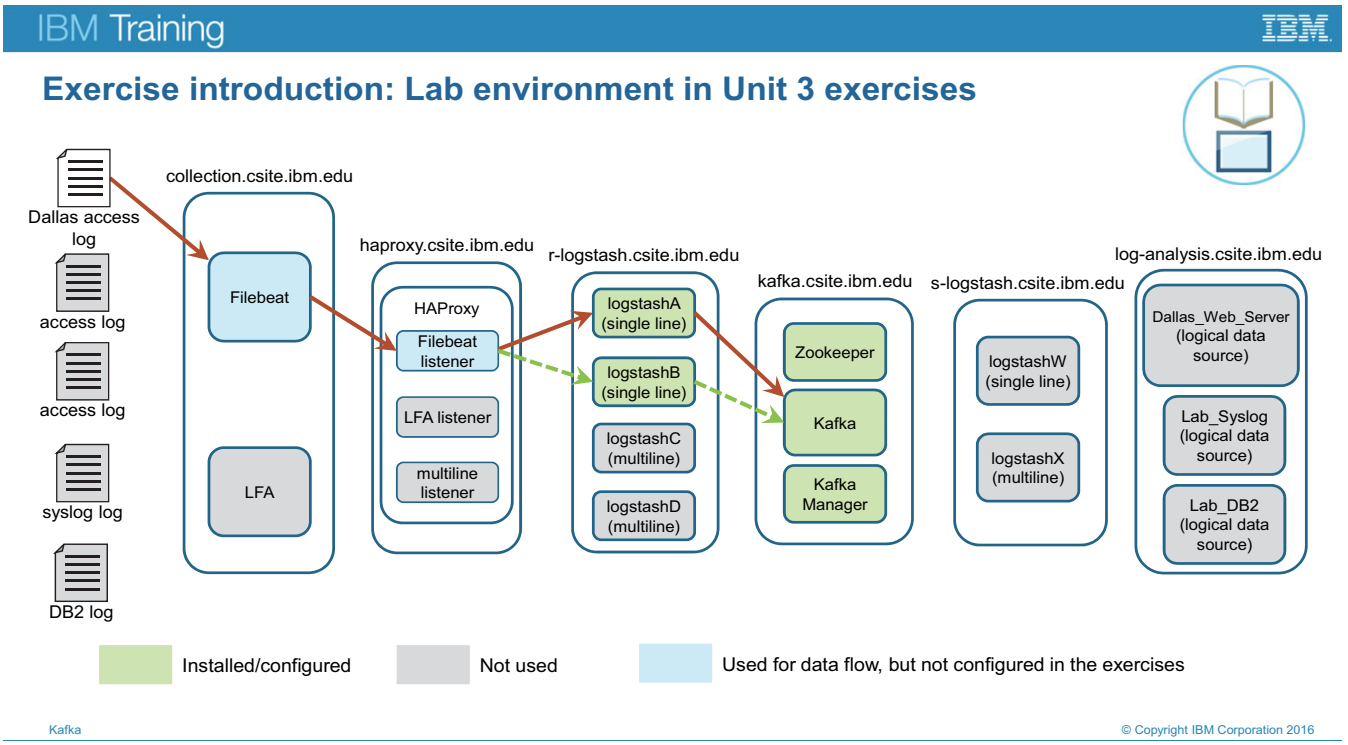



Kafka



*Exercise introduction: Finished lab environment*

At the end of this course, your lab environment will look like this diagram. Refer to this diagram as you complete the steps in the student exercises to help you understand what you are installing and configuring.



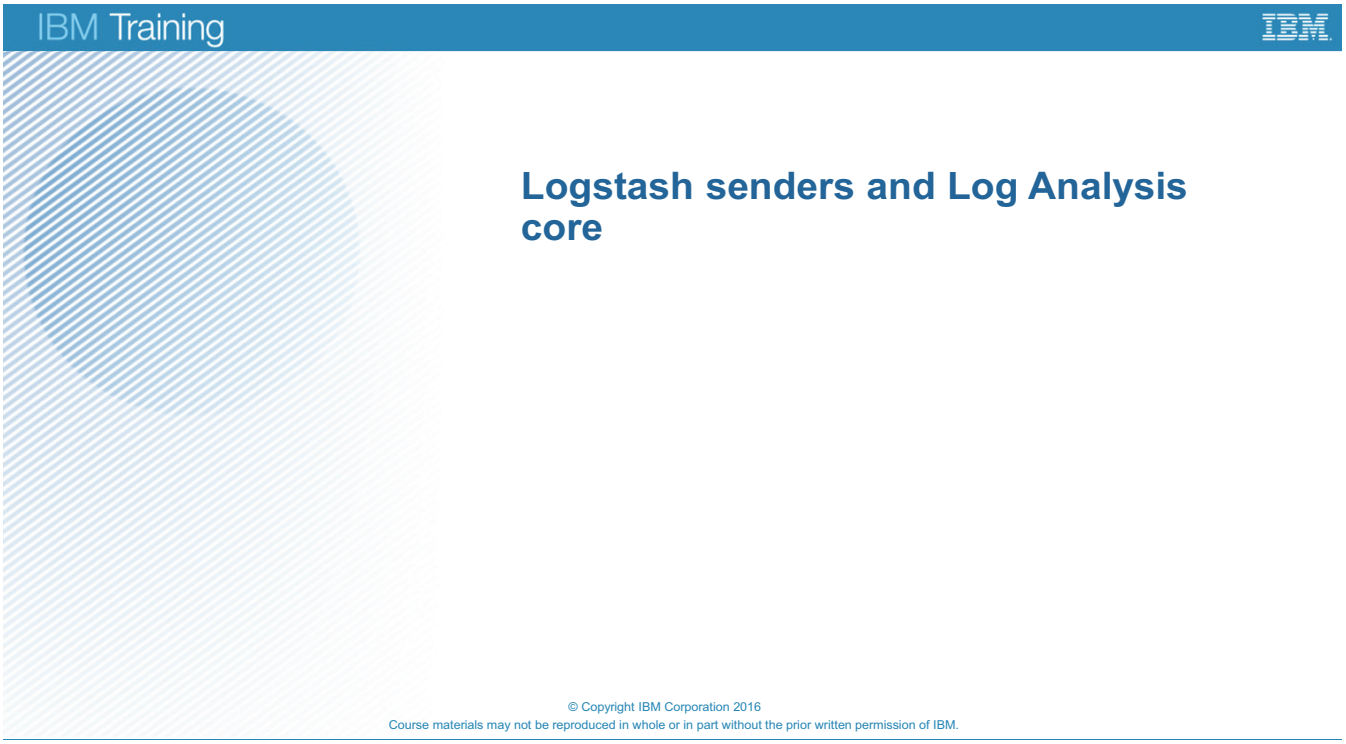

*Exercise introduction: Lab environment in Unit 3 exercises*

In the exercises for this unit, you install and configure the following components:

- A Kafka broker, to store messages
- A Zookeeper server, to coordinate Kafka communications
- Kafka Manager, to view data in your Kafka cluster
- The output section of your logstashA and logstashB receivers, to send data to Kafka (configuration only)

As you install and configure these components, you also verify the data flow between them.

# *Unit 4* Logstash senders and Log Analysis core




In this unit, you learn how to install and configure Logstash senders. You also learn how to configure the IBM Operations Analytics Log Analysis core software to accept data from a Logstash sender.

IBM Training

## Unit objectives

- Configure Logstash senders
- Configure Log Analysis data sources
- Diagram data flow from Kafka to Logstash senders to Log Analysis




*Unit objectives*

IBM Training

## About Logstash senders

- Logstash senders pull messages from a Kafka topic.
- Use the Logstash sender filter section for any text transformation or annotation operations.
- Use the Logstash sender filter section to set a *host* and *path*.
- Logstash senders use a custom output plug-in named ***scala*** to send data to Log Analysis.




*About Logstash senders*

In the Log Analysis scalable architecture, the role of a Logstash sender instance is to retrieve log messages from a Kafka broker and send those messages to the Log Analysis server.

You use the input section of a Logstash sender configuration to connect to Kafka and pull messages from a specific topic.

You use the filter section of a Logstash sender configuration to alter the log message or metadata that is attached to the log message. At minimum, Logstash senders create two metadata fields for each message: **host** and **path**.

You use the output section of a Logstash sender configuration to send log messages to the Log Analysis server.

IBM Training

## Logstash sender input section

```
input {
    kafka {
        zk_connect => "kafka.csite.ibm.edu:17981"
        group_id => "DEV_IBM-HTTP-Server_access-log"
        topic_id => "DEV_IBM-HTTP-Server_access-log"
        consumer_threads => 1
        consumer_restart_on_error => true
        consumer_restart_sleep_ms => 100
        decorate_events => true
    }# end HTTP server log Kafka input
    kafka {
        zk_connect => "kafka.csite.ibm.edu:17981"
        group_id => "labInstance_syslog"
        topic_id => "labInstance_syslog"
        consumer_threads => 1
        consumer_restart_on_error => true
        consumer_restart_sleep_ms => 100
        decorate_events => true
    }# end syslog server log Kafka input
}# end input section
```




*Logstash sender input section*

Logstash senders use the kafka input plug-in to pull messages from a topic in the Kafka cluster. In this example, the Logstash sender server is pulling messages from two different topics: `DEV_IBM-HTTP-Server_access-log` and `labInstance_syslog`. Notice that the kafka input plug-in is invoked twice: once for each topic.

The following table describes each line in the kafka input plug-in configuration example.

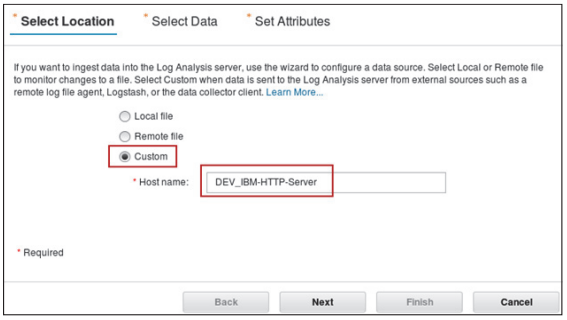
| Configuration option | Description |
|---|---|
| zk_connect | The host and port number of the Zookeeper server. |
| group_id | A name for the consumer group that identifies the Logstash server in Kafka. Use the same value for topic_id and group_id. |
| topic_id | The name of the Kafka topic you want to pull data from. |
| consumer_threads | The number or threads to concurrently pull data from topic. This is typically set to the number of partitions for the topic.<br><br>The consumer_threads parameter also specifies the number of consumers that are created in a consumer group. Each thread, or consumer, maps to a partition for the specified topic. This ensures that data is processed concurrently. |
| consumer_restart_on_error | If true, consumer threads will restart on errors. |

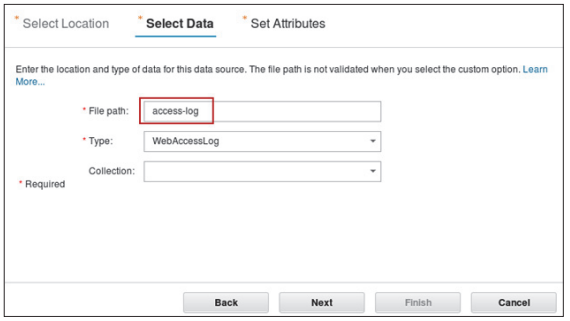
| Configuration option | Description |
|---|---|
| consumer_restart_sleep_ms | The number milliseconds to wait for consumer threads to restart after an error. |
| decorate_events | If true, metadata about Kafka, such as `topic` or `msg_size` is added to each log message. |

IBM Training

## Logstash sender filter section

```
filter {
    if "mutate_filebeat" in [tags] {
        mutate {
          add_field => [ "path", "%{[fields][type]}" ]
          replace => { "host" => "%{[fields][env]}_%{[fields][module]}" }
        } # end mutate
    } # end filebeat mutate condition
}# end filter section
```







*Logstash sender filter section*

You use the filter section of a Logstash sender configuration to alter the log message or metadata that is attached to the log message. At minimum, Logstash senders create two metadata fields for each message: **host** and **path**. You use the **host** and **path** fields when you configure the data source in the Log Analysis server.

In this example, the mutate filter in a Logstash sender configuration is creating a metadata field named **path**. The value for **path** is set with metadata that was added by the Filebeat server (**type**). The mutate filter is also replacing the existing value for **host** with metadata that was added by the Filebeat server (**env_module**). The final value for **path** is **access-log**. The final value for **host** is `DEV_IBM-HTTP-Server`.

At the bottom of the slide are pictures of the Log Analysis data source wizard. Notice that you use the values of **host** and **path** that were set by the Logstash sender when you create the data source.

IBM Training IBM

## Logstash sender output section

```
output {

    scala {
        scala_url => "https://log-analysis.csite.ibm.edu:9987/Unity/DataCollector"
        scala_user => "unityadmin"
        scala_password => "object00"
        scala_keystore_path => ""
        batch_size => 500000
        idle_flush_time => 5
        sequential_flush => true
        num_concurrent_writers => 20
        use_structured_api => false
        disk_cache_path => "/opt/logstashW/training/cache/basecache"
        date_format_string => "yyyy-MM-dd'T'HH:mm:ssX"
        log_file => "/opt/logstashW/logstash-2.2.1/log/scala_logstashW.log"
        log_level => "info"
    }#end scala output

}# end output section
```




*Logstash sender output section*

You use the output section of a Logstash sender configuration to send log messages to the Log Analysis server.

Use the scala output plug-in to send data to the Log Analysis server. The following table describes each line in the scala output plug-in configuration example.

| Configuration option | Description |
|---|---|
| scala_url | The REST endpoint for the Log Analysis Generic Receiver REST API. |
| scala_user | The Log Analysis user name. |
| scala_password | The Log Analysis user password. |
| scala_keystore_path | The full path to the directory where the keystore file is saved. Enter a value if you want to encode the plain text scala_password. |
| batch_size | The maximum number of bytes that can be buffered for each data source before Logstash sends data to the Log Analysis server. |
| idle_flush_time | The maximum time between successive data transmissions for each data source. |
| sequential_flush | Defines whether batches for each data source are sent sequentially. If true, batches are sent sequentially. |
| num_concurrent_writers | The number of threads that concurrently transmit batches of data to the Log Analysis server. |

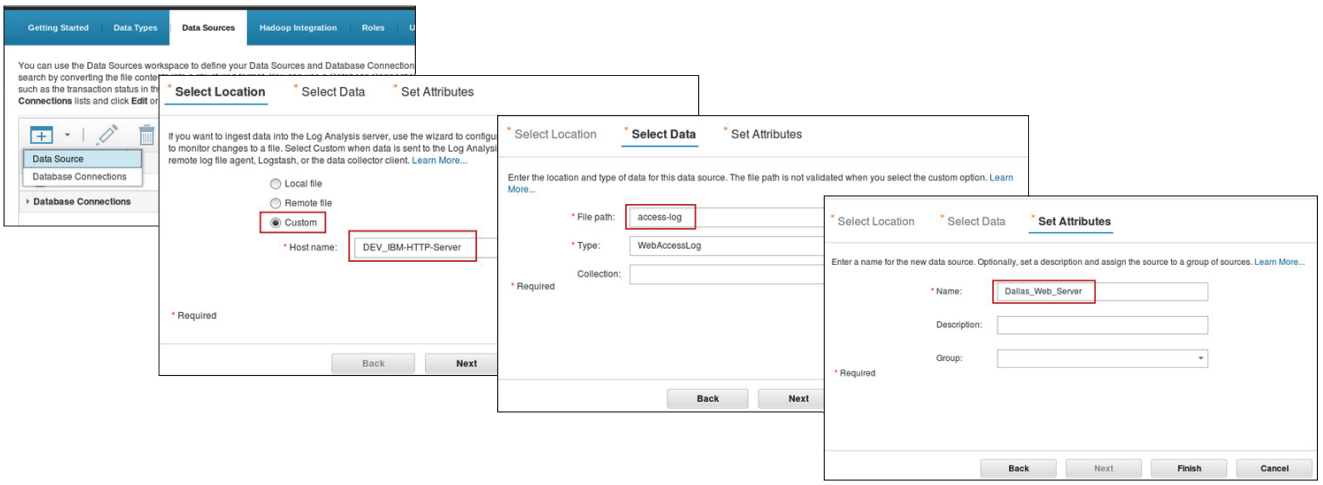
| Configuration option | Description |
|---|---|
| use_structured_api | Determines whether data is transmitted to the Log Analysis server in JSON format. If true, data is sent in JSON format. |
| disk_cache_path | The path on the file system that temporarily buffers data. The scala output plug-in writes data to this path before sending data to the Log Analysis server. |
| date_format_string | The string format that all date fields are transformed to before transmission to the Log Analysis server. |
| log_file | The file that is used for logging information from the scala output plug-in. |
| log_level | The level of logging information for the scala output plug-in. The supported levels are: fatal, error, warn, info, and debug. |

**Note:** If you installed Logstash by decompressing the installation file or with an RPM Package Manager file (RPM), you must manually copy the scala output plug-in files to the Logstash server. For example, copy the files to the following directory:
`<LOGSTASH_HOME>/vendor/bundle/jruby/1.9/gems/logstash-core-2.2.1-java/lib/logstash/outputs/`

IBM Training

## Configuring Log Analysis

- Use the Log Analysis administrator user interface to add a data source
- Use the host and path that were set by the Logstash sender, not the actual path of the target log file




Logstash senders and Log Analysis core



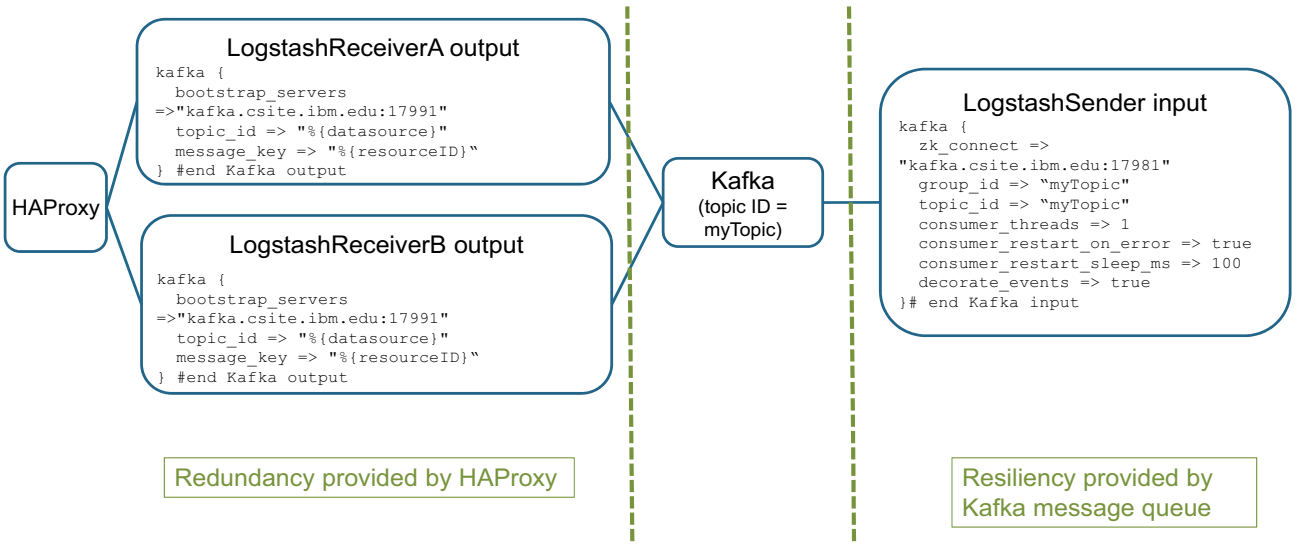
*Configuring Log Analysis*

To configure Log Analysis to accept data from the target log file, you must create a data source. Use the data source wizard in the Log Analysis administrator user interface to add a data source.

Use the host and path that were set by the Logstash sender filter section, not the actual path of the target log file.

IBM Training

## Logstash sender resiliency




Logstash senders and Log Analysis core



*Logstash sender resiliency*

Resiliency for Logstash senders is provided by the Kafka cluster. If a Logstash sender server goes down, it can read messages that are stored in the Kafka message queue when it comes back up.

IBM Training

## Unit summary

- Configure Logstash senders
- Configure Log Analysis data sources
- Diagram data flow from Kafka to Logstash senders to Log Analysis




*Unit summary*

IBM Training

## Logstash senders and Log Analysis core exercises

### Exercise 1 Installing and configuring the Logstash sender
### Exercise 2 Sending data to Log Analysis
### Exercise 3 Verifying Logstash sender resiliency

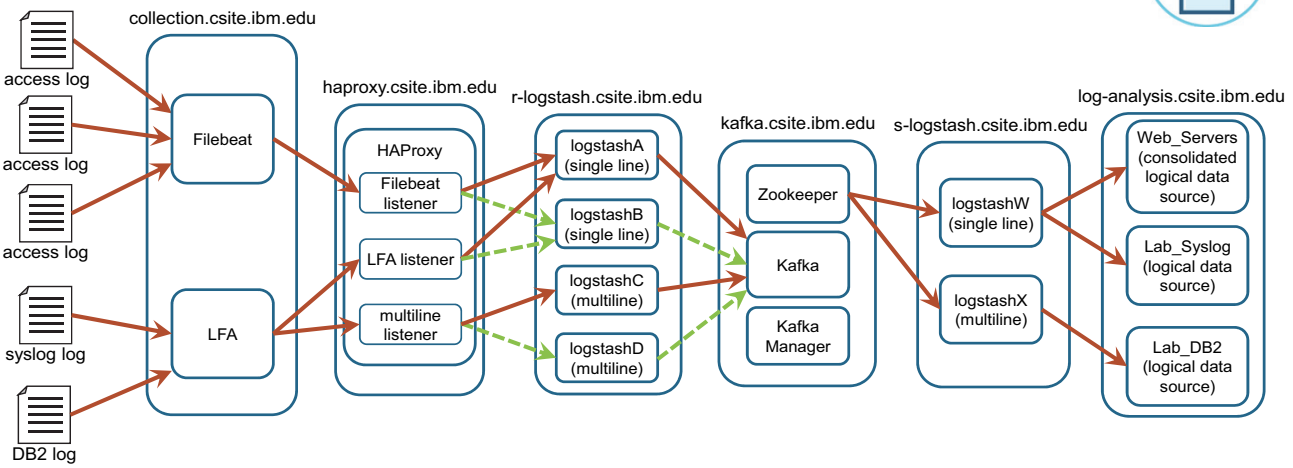



*Student exercises*

Perform the exercises for this unit.

IBM Training

## Exercise introduction: Finished lab environment

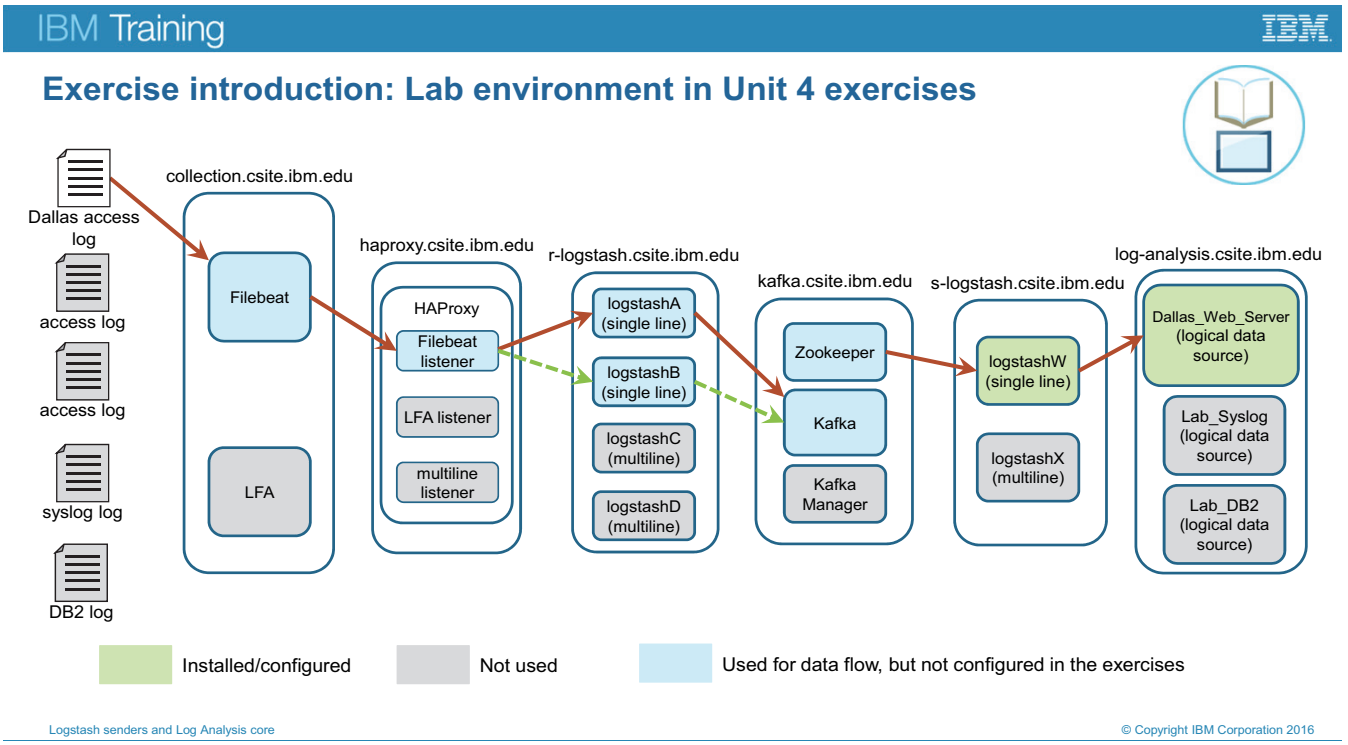



Logstash senders and Log Analysis core



*Exercise introduction: Finished lab environment*

At the end of this course, your lab environment will look like this diagram. Refer to this diagram as you complete the steps in the student exercises to help you understand what you are installing and configuring.



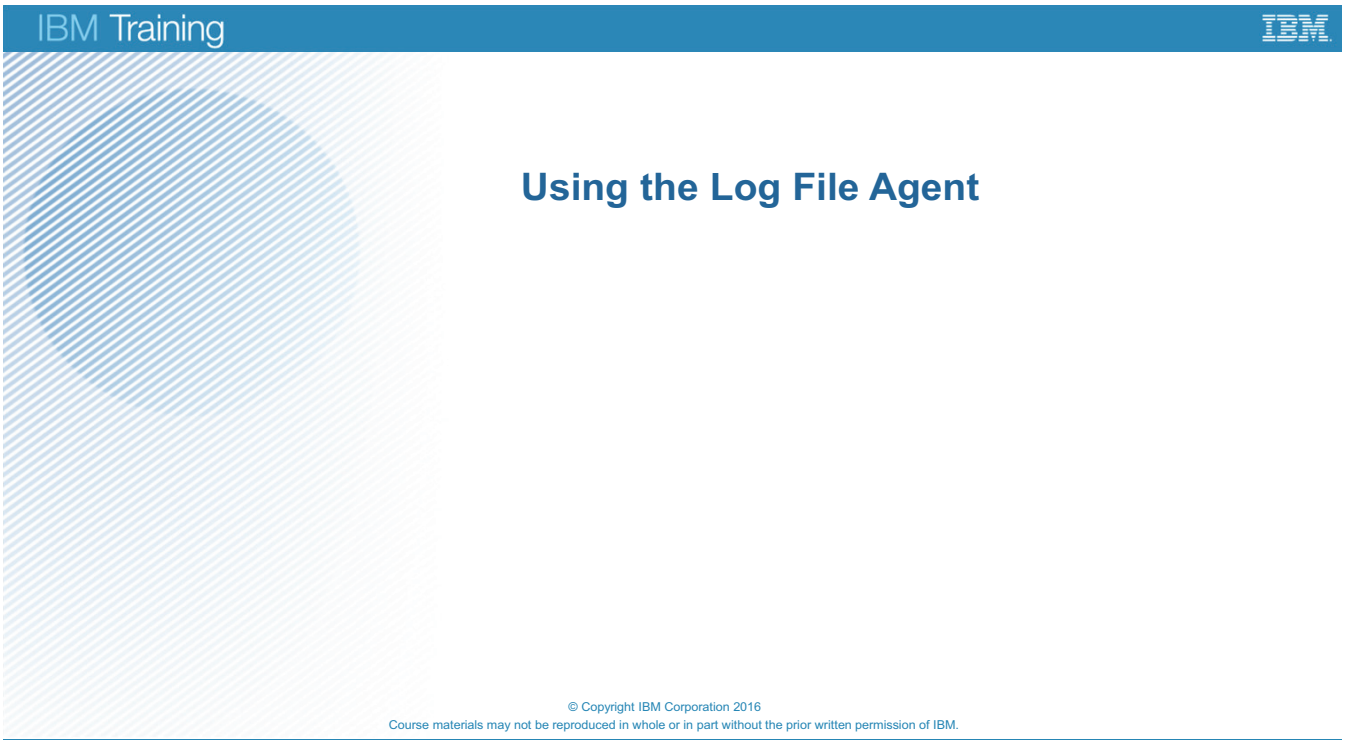

*Exercise introduction: Lab environment in Unit 4 exercises*

In the exercises for this unit, you install and configure the following components:

- A Logstash sender, to pull messages from Kafka and send them to the Log Analysis server
- A data source in Log Analysis, to accept data from the target log file (configuration only)

As you install and configure these components, you also verify the data flow between them.

# *Unit 5* Using the Log File Agent




You can use the IBM Tivoli Log File Agent, which is included with Log Analysis, as a collection agent. This unit teaches you how to install and configure the Log File Agent.

IBM Training

## Unit objectives

- Install the Log File Agent
- Configure the Log File Agent
- Configure Logstash receivers to process Log File Agent messages




*Unit objectives*

IBM Training

## Installing the Log File Agent

Use the Log Analysis remote installation tool to install the Log File Agent (LFA):

1. Edit the `ssh-config.properties` file with details about the host that you want to install to.
2. Run the install.sh tool.

The remote installation tool is in the `<LA_HOME>/remote_install_tool` directory.




*Installing the Log File Agent*

You can use the Log Analysis remote installation tool to install the Log File Agent (LFA). The following steps show you how to install the Log File Agent.

1. Change to the remote installation tool directory of the Log Analysis server.

   ```
   cd <LA_HOME>/remote_install_tool
   ```

2. Open the **ssh-config.properties** file in a text editor. This example uses vi.

   ```
   vi config/ssh-config.properties
   ```

3. Change the following values in bold typeface. These values are the details about how to connect to the host where you want to install the Log File Agent.

   ```
   REMOTE_HOST=log_file_agent_target_host
   PORT=ssh_port_number
   TIME_OUT=60000
   USER=lfa_user
   #PASSWORD can be commented while using Public key based authentication
   PASSWORD=lfa_user_password
   ```

4. Save and close the file when you are finished.

5. Run the following command to start the remote installation tool.

   ```
   <LA_HOME>/remote_install_tool/install.sh
   ```

6. Enter the directory where you want to install the Log File Agent. In this example, the installation directory is `/opt/LFA`.

   ```
   Enter Remote Top Level Installation Directory absolute path:
   [/home/netcool/LogAnalysis]
   ```

   **/opt/LFA**

7. Enter **n** when you are prompted to install EIF Receiver instances.

   ```
   Install EIF Receiver Instances (y|Y|n|N):[y]
   ```

   **n**

8. Enter **y** when you are prompted to install Log File Agent (LFA) 6.3.

   ```
   Install LFA 6.3 (y|Y|n|N):[y]
   ```

   **y**

9. Enter **n** when you are prompted to install Logstash. The installation starts.

   ```
   Install logstash 2.2.1 (y|Y|n|N):[y]
   ```

   **n**

The installation takes several minutes to complete. The following message confirms that the installation was successful.

```
Response:================================================
Response:    COMPONENT PIDSTATUS
Response:================================================
Response:  Log File Agent 9784  UP
Response:================================================
End Time:Thu May 26 16:06:54 UTC 2016
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Total install duration (seconds):109

++++++++++++++++++++ Installation Ends ++++++++++++++++++++
```

IBM Training

## Configuring the Log File Agent

You use two files to configure the Log File Agent to monitor a target log:

- A .conf file
- A .fmt file

```
LogSources=/var/log/messages.log
BufEvtPath=/opt/LFA/IBM-LFA-6.30/logs/lab-syslog.cache
FileComparisonMode=CompareByAllMatches
ServerLocation=ha-proxy.csite.ibm.edu
ServerPort=5530
FQDomain=yes
BufferEvents=YES
BufEvtMaxSize=102400
ConnectionMode=CO
PollInterval=3
NumEventsToCatchUp=-1
ServerSSL=NO
```

lab-syslog.conf

```
REGEX AllRecords
(.*)
hostname LABEL
-file FILENAME
RemoteHost DEFAULT
logpath PRINTF("%s",file)
type syslog
instance labInstance
cluster NONE
module syslog
env DEV
functional NONE
site NONE
text $1
END
```

lab-syslog.fmt

Using the Log File Agent



*Configuring the Log File Agent*

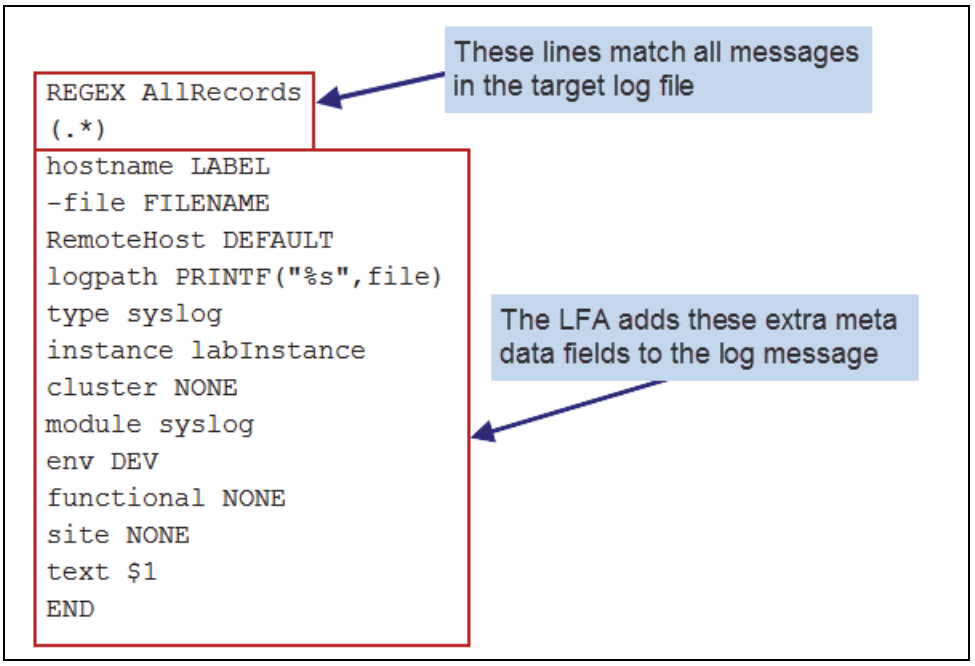
You configure the Log File Agent by creating two files: a .conf file and a .fmt file. You create these files in the `<`*LFA_HOME*`>/IBM-LFA-6.30/config/lo/` directory.

The following table describes each line in the Log File Agent conf file example.

| Option | Description |
|---|---|
| LogSources | Specifies the text log files to poll for messages. The complete path to each file must be specified, and file names must be separated by commas. |
| BufEvtPath | Specifies the full path name of the agent cache file. |
| FileComparisonMode | Specifies which log files are monitored when more than one matches a wildcard pattern. If set to CompareByAllMatches, all files that match the wildcard pattern that is specified in LogSources are monitored. |
| ServerLocation | Specifies the host name or IP address where log messages are sent. |
| ServerPort | Specifies the port number to use when sending log messages. |
| FQDomain | Specifies how and if the agent sets a domain name.<br>• If set to yes, the agent determines the system domain name itself.<br>• If set to no, the agent does not set a domain name. The host name attribute is assigned a blank string.<br>• If set so that it does not contain a yes or no value, the domain name is accepted as the value and it is appended to the host name. |

| Option | Description |
|---|---|
| BufferEvents | Specifies how event buffering is enabled. The possible values are:<br>• YES: Stores events in the file that is specified by the BufEvtPath option.<br>• MEMORY_ONLY: Buffers events in memory.<br>• NO: Does not store or buffer events. |
| BufEvtMaxSize | Specifies the maximum size, in KB, of the agent cache file. |
| EventMaxSize | Specifies in bytes, the maximum size of a generated log message event. |
| ConnectionMode | Specifies whether the connection is connection-oriented or connectionless. |
| PollInterval | Specifies the frequency, in seconds, to poll each log file that is listed in the LogSources option for new messages |
| NumEventsToCatchUp | Specifies the event in the log that the agent starts with. When set to -1, the agent saves its place in the file that is being monitored. When set to 0, the agent starts with the next event in the logs. |
| ServerSSL | Specifies whether to use Secure Sockets Layer when sending data. |

The following diagram describes key fields in the .fmt file example.



IBM Training

## Adding a HAProxy listener for Log File Agents

```
## Listener for LFA Logstash cluster
## Specify Logstash instances for processing data from LFA
listen  LFA_Cluster ha-proxy.csite.ibm.edu:5530
        mode tcp
        balance roundrobin
        server receiver-logstashA logstashA.csite.ibm.edu:5540 check fall 2 rise 3 inter 1000
        server receiver-logstashB logstashB.csite.ibm.edu:5541 check fall 2 rise 3 inter 1000
```




*Adding a HAProxy listener for Log File Agents*

You configure HAProxy by editing a file named `/etc/haproxy/haproxy.cfg`. In this scalable architecture, HAProxy must run in tcp mode.

You must configure a listener for HAProxy to accept data from a log collection agent and forward it to a Logstash receiver cluster. In this example, HAProxy is listening on port 5530. Any data HAProxy receives on port 5530 is forwarded to either logstashA.csite.ibm.edu:5540 or logstashB.csite.ibm.edu:5541.

HAProxy is aware of the state of each server. In this example, HAProxy performs health checks on each Logstash server every 1000 milliseconds. HAProxy considers a Logstash server down if two consecutive health check failures occur (`fall 2`). HAProxy considers a Logstash server up again after three consecutive successful health checks (`rise 3`).

The line `balance roundrobin` configures HAProxy to send data to each Logstash server in turns.

IBM Training

## Logstash receiver input section

```
input {

    tcp {
        port => 5540
        type => "lfa"
        codec => line { charset => "US-ASCII" }
    } #end lfa input

} #end input section
```




*Logstash receiver input section*

In the Log Analysis scalable architecture, the role of a Logstash receiver instance is to accept log messages from collection agents and send the messages to Kafka.

Use the **tcp** input plug-in to accept data from the Log File Agent. Use the **type** option to add a field named **type** to each log message. Setting a type for messages from the LFA makes it easier to alter these messages with the filter section.

Use the codec option to set the character encoding.

IBM Training IBM

## Extra text in Log File Agent messages

```
"message" =>
"\\u0001<START>>\\u0000\\u0000\\u0000\\u0000\\u0000\\u0000\\u0000\\u0000\\u0
000\\u0000\\u0000\\u0000\\u0000\\u0000\\u0000\\u0000\\u0000\\u0000\\u0000\\u
0000\\u0000\\u0000\\u0001\\xBD\\u0000\\u0000\\u0000\\u0004\\u0003AllRecords;
type='syslog';text='2016-05-12T23:39:50.764527+00:00
host=analysishost,relayHost=analysishost, tag=kernel:,
programName=kernel,procid=-, facility=kern, sev=info,appName=kernel,
msg=Reserving 129MB of memory at 48MB for crashkernel (System RAM:
9216MB)';RemoteHost='';site='NONE';instance='labInstance';hostname='collecti
on';cluster='NONE';module='syslog';env='DEV';logpath='/var/log/messages.log'
;functional='NONE';END",
      "@version" => "1",
    "@timestamp" => "2016-05-26T18:20:08.149Z",
          "host" => "192.168.100.176",
          "port" => 45437,
          "type" => "lfa"
```

Using the Log File Agent 

*Extra text in Log File Agent messages*

The Log File Agent sends extra text with each log message. You use the filter section of the Logstash receiver to remove this unnecessary text.

IBM Training IBM

## Removing the extra text with the Logstash receiver

```
filter {

    if [type] == "lfa" {
        grok {
            patterns_dir => "/opt/logstashA/logstash-2.2.1/patterns"
            match => [ "message", "%{LFAMESSAGE}" ]
            add_tag => ["grok_lfa"]
        } #end initial LFA grok
    } #end initial LFA condition

    if "grok_lfa" in [tags] {
        mutate {
            replace => ["message","%{LFA_ORIG_MSG}"]
            add_field => [ "datasource", "%{LFA_INSTANCE}_%{LFA_MODULE}"]
            add_field => [ "resourceID", "%{LFA_HOSTNAME}_%{LFA_LOGNAME}_1"]
        }# end mutate
    }# end grok_lfa condition

} #end filter section
```




*Removing the extra text with the Logstash receiver*

Use the grok filter to remove the unnecessary text from the Log File Agent. The grok filter uses regular expression objects called patterns. The grok filter uses these patterns to match meaningful strings of text and identify them as fields within a log message. You can use a file that contains a grok pattern to match the unnecessary text from the Log File Agent. This patterns file is included with Log Analysis. You can find it in the following location:

`<LA_HOME>/kafka/test-configs/logstash-configs/SCALAPATTERNS`

Copy the SCALAPATTERNS file to the Logstash receiver server.

## *Grok filter configuration example*

In the following example, line numbers have been added to the grok filter configuration for clarity.

```
1 if [type] == "lfa" {
2     grok {
3         patterns_dir => "/opt/logstashA/logstash-2.2.1/patterns"
4         match => [ "message", "%{LFAMESSAGE}" ]
5         add_tag => ["grok_lfa"]
6     } #end initial LFA grok
7 } #end initial LFA condition
```

- Line 1 is a condition. Only messages where the type is **lfa** are processed by the grok filter.
- Line 2 invokes the grok filter plug-in.
- Line 3 points to the directory where you copied the SCALAPATTERNS file.

- Line 4 uses a pattern named LFAMESSAGE to match meaningful strings of text in the data that was received from the Log File Agent.
- Line 5 adds a tag to messages that were successfully matched by the grok filter.
- Line 6 closes the grok filter configuration.
- Line 7 closes the condition.

## *Mutate filter configuration example*

In the following example, line numbers have been added to the mutate filter configuration for clarity.

```
1 if "grok_lfa" in [tags] {
2     mutate {
3         replace => ["message","%{LFA_ORIG_MSG}"]
4         add_field => [ "datasource", "%{LFA_INSTANCE}_%{LFA_MODULE}"]
5         add_field => [ "resourceID", "%{LFA_HOSTNAME}_%{LFA_LOGNAME}_1"]
6     }# end mutate
7 }# end grok_lfa condition
```

- Line 1 is a condition. Only messages that were successfully matched by the preceding grok filter are processed by the mutate filter.
- Line 2 invokes the mutate filter plug-in.
- Line 3 replaces the text in the **message** field with text that was identified as LFA_ORIG_MSG by the preceding grok filter. The text that was identified as LFA_ORIG_MSG is the actual log message from the target log file. This line removes the unnecessary text that was added by the Log File Agent.
- Line 4 adds a field named **datasource** and sets the value of **datasource** with metadata that was added by the Log File Agent. The **datasource** field is used in the Logstash receiver output section.
- Line 5 adds a field named **resourceID** and sets the value of **resourceID** with metadata that was added by the Log File Agent. The **resourceID** field is used in the Logstash receiver output section.
- Line 6 closes the mutate filter configuration.
- Line 7 closes the condition.

IBM Training

## Metadata from the Log File Agent

```
"message" =>
"\\u0001<START>>\\u0000\\u0000\\u0000\\u0000\\u0000
0\\u0000\\u0000\\u0000\\u0000\\u0000\\u0000\\u0000\\u0000
\\u0000\\u0000\\u0000\\u0000\\u0000\\u0000\\u0000\\u0000\
\u0000\\u0000\\u0000\\u0001\\xBD\\u0000\\u0000\\u0
000\\u0004\\u0003AllRecords;type='syslog';text='20
16-05-12T23:39:50.764527+00:00
host=analysishost,relayHost=analysishost,
tag=kernel:, programName=kernel,procid=-,
facility=kern, sev=info,appName=kernel,
msg=Reserving 129MB of memory at 48MB for
crashkernel (System RAM:
9216MB)';RemoteHost='';site='NONE';instance='labIn
stance';hostname='collection';cluster='NONE';modul
e='syslog';env='DEV';logpath='/var/log/messages.lo
g';functional='NONE';END",
      "@version" => "1",
    "@timestamp" => "2016-05-26T18:20:08.149Z",
          "host" => "192.168.100.176",
          "port" => 45437,
          "type" => "lfa"
```

Message from the LFA

```
REGEX AllRecords
(.*)
hostname LABEL
-file FILENAME
RemoteHost DEFAULT
logpath PRINTF("%s",file)
type syslog
instance labInstance
cluster NONE
module syslog
env DEV
functional NONE
site NONE
text $1
END
```

lab-syslog.fmt

```
filter {

  if [type] == "lfa" {
      grok {
          patterns_dir => "/opt/logstashA/logstash-2.2.1/patterns"
          match => [ "message", "%{LFAMESSAGE}" ]
          add_tag => ["grok_lfa"]
      } #end initial LFA grok
  } #end initial LFA condition

  if "grok_lfa" in [tags] {
      mutate {
          replace => ["message","%{LFA_ORIG_MSG}"]
          add_field => [ "datasource", "%{LFA_INSTANCE}_%{LFA_MODULE}"]
          add_field => [ "resourceID", "%{LFA_HOSTNAME}_%{LFA_LOGNAME}_1"]
      }# end mutate
  }# end grok_lfa condition

} #end filter section
```

logstashreceiver.conf

Using the Log File Agent



*Metadata from the Log File Agent*

You configure the Log File Agent to add metadata to each log message with the. fmt file. In this example, the Log File Agent is adding metadata such as **hostname**, **instance**, and **module**.

The grok filter in the Logstash receiver filter section uses a pattern named LFAMESSAGE to match the metadata in the text from the Log File Agent. Logstash stores the metadata in fields named **LFA_HOSTNAME, LFA_INSTANCE**, **LFA_MODULE**, and so on.

The mutate filter in the Logstash receiver filter section adds a field named **datasource**. The value of **datasource** is constructed by using the fields **LFA_INSTANCE** and **LFA_MODULE**. These two fields were matched by the grok filter. In this example, the value of **datasource** is **labInstance_syslog**.

The mutate filter adds another field named **resourceID**. The value of **resourceID** is constructed by using the fields **LFA_HOSTNAME** and **LFA_LOGNAME**. These two fields were matched by the grok filter. In this example, the value of **resourceID** is `collection_/software/log_samples/messages.log_1`.

IBM Training IBM

## Logstash receiver output section

```
output {

    if ("grok_lfa" in [tags]) and ! ("_grokparsefailure" in [tags]) {
        kafka {
            bootstrap_servers =>"kafka.csite.ibm.edu:17991"
            topic_id => "%{datasource}"
            message_key => "%{resourceID}"
        } #end Kafka output
    } #end Kafka condition

} # end output section
```




*Logstash receiver output section*

The two fields that were created by the mutate filter, **datasource** and **resourceID**, are used in the configuration of the kafka output plug-in.

The value of **datasource** is the name of the Kafka topic that this Logstash receiver writes messages to. Notice that the value of **datasource** is used as the **topic_id**.

Any messages with the same **message_key** are written to the same partition within the topic. Notice that the value of **resourceID** is used as the **message_key**.

IBM Training

## Logstash sender input section

```
input {
    kafka {
        zk_connect => "kafka.csite.ibm.edu:17981"
        group_id => "DEV_IBM-HTTP-Server_access-log"
        topic_id => "DEV_IBM-HTTP-Server_access-log"
        consumer_threads => 1
        consumer_restart_on_error => true
        consumer_restart_sleep_ms => 100
        decorate_events => true
    }# end HTTP server log Kafka input

    kafka {
        zk_connect => "kafka.csite.ibm.edu:17981"
        group_id => "labInstance_syslog"
        topic_id => "labInstance_syslog"
        consumer_threads => 1
        consumer_restart_on_error => true
        consumer_restart_sleep_ms => 100
        decorate_events => true
    }# end syslog server log Kafka input
}# end input section
```

Using the Log File Agent



*Logstash sender input section*

In the Log Analysis scalable architecture, the role of a Logstash sender instance is to retrieve log messages from a Kafka broker and send those messages to the Log Analysis server.

Logstash senders use the kafka input plug-in to pull messages from a topic in the Kafka cluster. In this example, the Logstash sender server is pulling messages from two different topics: DEV_IBM-HTTP-Server_access-log and labInstance_syslog. Notice that the kafka input plug-in is invoked twice, once for each topic.

The following table describes each line in the kafka input plug-in configuration example.

| Configuration option | Description |
|---|---|
| zk_connect | The host and port number of the Zookeeper server. |
| group_id | A name for the consumer group that identifies the Logstash server in Kafka. Use the same value for topic_id and group_id. |
| topic_id | The name of the Kafka topic you want to pull data from. |
| consumer_threads | The number or threads to concurrently pull data from topic. This is typically set to the number of partitions for the topic.<br><br>The consumer_threads parameter also specifies the number of consumers that are created in a consumer group. Each thread, or consumer, maps to a partition for the specified topic. This ensures that data is processed concurrently. |
| consumer_restart_on_error | If true, consumer threads will restart on errors. |

| Configuration option | Description |
|---|---|
| consumer_restart_sleep_ms | The number milliseconds to wait for consumer threads to restart after an error. |
| decorate_events | If true, metadata about Kafka, such as **topic** or **msg_size** is added to each log message. |

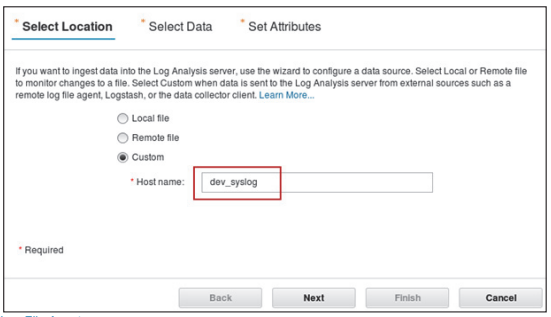
IBM Training

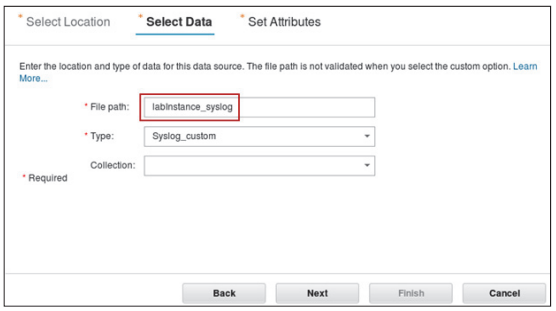
## Logstash sender filter section

```
filter {

    if "grok_lfa" in [tags] {
        mutate {
          add_field => [ "path", "%{LFA_INSTANCE}_%{LFA_MODULE}" ]
          replace => ["host", "%{LFA_ENVIRONMENTNAME}_%{LFA_MODULE}"]
        } # end mutate
    } # end lfa mutate condition

}# end filter section
```







*Logstash sender filter section*

You use the filter section of a Logstash sender configuration to alter the log message or metadata that is attached to the log message. At minimum, Logstash senders create two metadata fields for each message: **host** and **path**. You use the **host** and **path** fields when you configure the data source in the Log Analysis server.

In this example, the mutate filter in a Logstash sender configuration is creating a metadata field named **path**. The value for **path** is set with metadata that was added by the Log File Agent (**LFA_INSTANCE** and **LFA_MODULE**). The mutate filter is also replacing the existing value for **host** with metadata that was added by the Log File Agent (**LFA_ENVIRONMENTNAME** and **LFA_MODULE**). The final value for **path** is **labInstance_syslog**. The final value for **host** is **DEV_syslog**.

At the bottom of the slide are pictures of the Log Analysis data source wizard. Notice that you use the values of **host** and **path** that were set by the Logstash sender when you create the data source.

IBM Training

## Logstash sender output section

```
output {

    scala {
        scala_url => "https://log-analysis.csite.ibm.edu:9987/Unity/DataCollector"
        scala_user => "unityadmin"
        scala_password => "object00"
        scala_keystore_path => ""
        batch_size => 500000
        idle_flush_time => 5
        sequential_flush => true
        num_concurrent_writers => 20
        use_structured_api => false
        disk_cache_path => "/opt/logstashW/training/cache/basecache"
        date_format_string => "yyyy-MM-dd'T'HH:mm:ssX"
        log_file => "/opt/logstashW/logstash-2.2.1/log/scala_logstashW.log"
        log_level => "info"
    }#end scala output

}# end output section
```




*Logstash sender output section*

You use the output section of a Logstash sender configuration to send log messages to the Log Analysis server.

Use the scala output plug-in to send data to the Log Analysis server. The following table describes each line in the scala output plug-in configuration example.

| Configuration option | Description |
|---|---|
| scala_url | The REST endpoint for the Log Analysis Generic Receiver REST API. |
| scala_user | The Log Analysis user name. |
| scala_password | The Log Analysis user password. |
| scala_keystore_path | The full path to the directory where the keystore file is saved. Enter a value if you want to encode the plain text scala_password. |
| batch_size | The maximum number of bytes that can be buffered for each data source before Logstash sends data to the Log Analysis server. |
| idle_flush_time | The maximum time between successive data transmissions for each data source. |
| sequential_flush | Defines whether batches for each data source are sent sequentially. If true, batches are sent sequentially. |
| num_concurrent_writers | The number of threads that concurrently transmit batches of data to the Log Analysis server. |

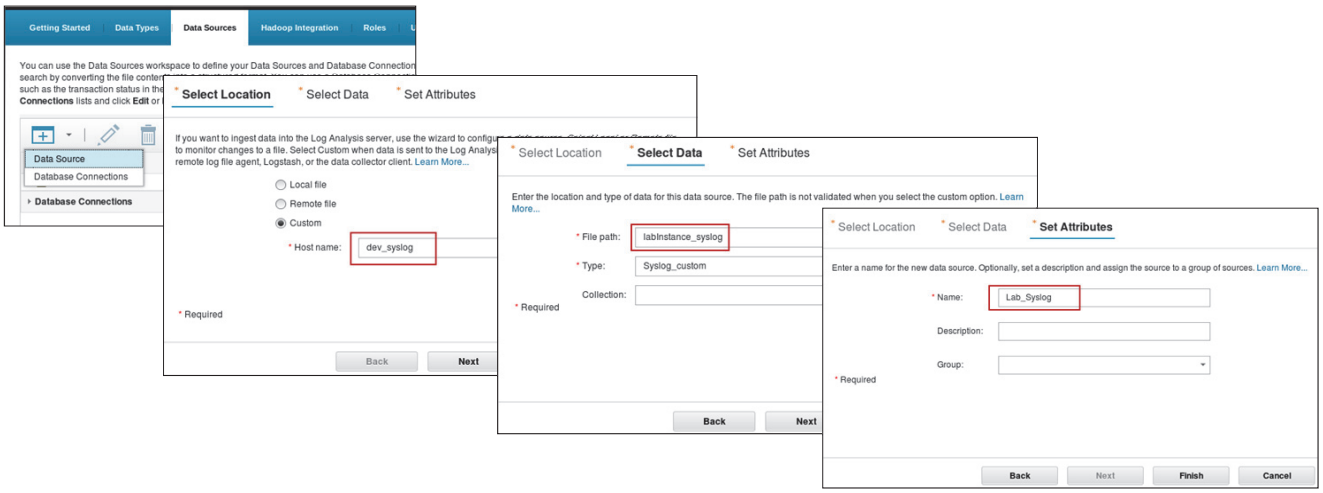
| Configuration option | Description |
|---|---|
| use_structured_api | Determines whether data is transmitted to the Log Analysis server in JSON format. If true, data is sent in JSON format. |
| disk_cache_path | The path on the file system that temporarily buffers data. The scala output plug-in writes data to this path before sending data to the Log Analysis server. |
| date_format_string | The string format that all date fields are transformed to before transmission to the Log Analysis server. |
| log_file | The file that is used for logging information from the scala output plug-in. |
| log_level | The level of logging information for the scala output plug-in. The supported levels are: fatal, error, warn, info, and debug. |

**Note:** If you installed Logstash by decompressing the installation file or with an RPM Package Manager file (RPM), you must manually copy the scala output plug-in files to the Logstash server. For example, copy the files to the following directory:
`<LOGSTASH_HOME>/vendor/bundle/jruby/1.9/gems/logstash-core-2.2.1-java/lib/logstash/outputs/`

IBM Training

## Configuring Log Analysis

- Use the Log Analysis administrator user interface to add a data source.
- Use the host and path that were set by the Logstash sender, not the actual path of the target log file.




Using the Log File Agent



*Configuring Log Analysis*

To configure Log Analysis to accept data from the target log file, you must create a data source. Use the data source wizard in the Log Analysis administrator user interface to add a data source.

Use the host and path that were set by the Logstash sender filter section, not the actual path of the target log file.

IBM Training

## Unit summary

- Install the Log File Agent
- Configure the Log File Agent
- Configure Logstash receivers to process Log File Agent messages




*Unit summary*

IBM Training

## Using the Log File Agent exercises

**Exercise 1 Installing and configuring the Log File Agent**
**Exercise 2 Configuring HAProxy**
**Exercise 3 Configuring the logstashA receiver**
**Exercise 4 Configuring the logstashB receiver**
**Configuring the Logstash sender and the Log Analysis data source**

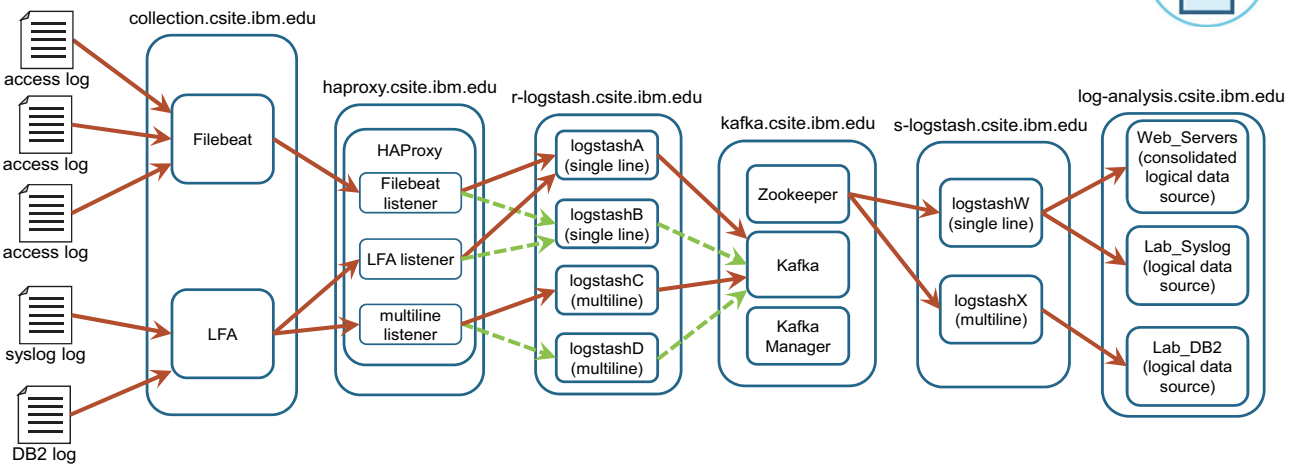



*Student exercises*

Perform the exercises for this unit.

IBM Training

## Exercise introduction: Finished lab environment

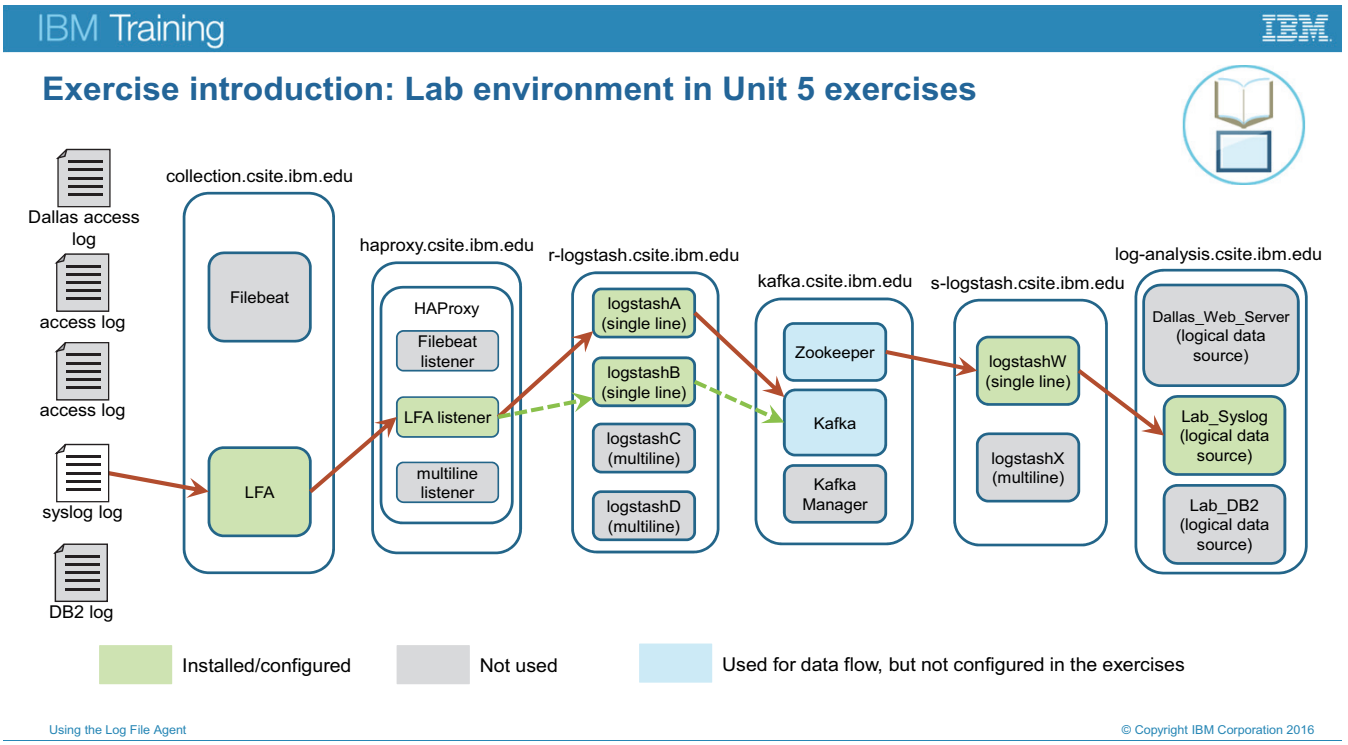



Using the Log File Agent



*Exercise introduction: Finished lab environment*

At the end of this course, your lab environment will look like this diagram. Refer to this diagram as you complete the steps in the student exercises to help you understand what you are installing and configuring.



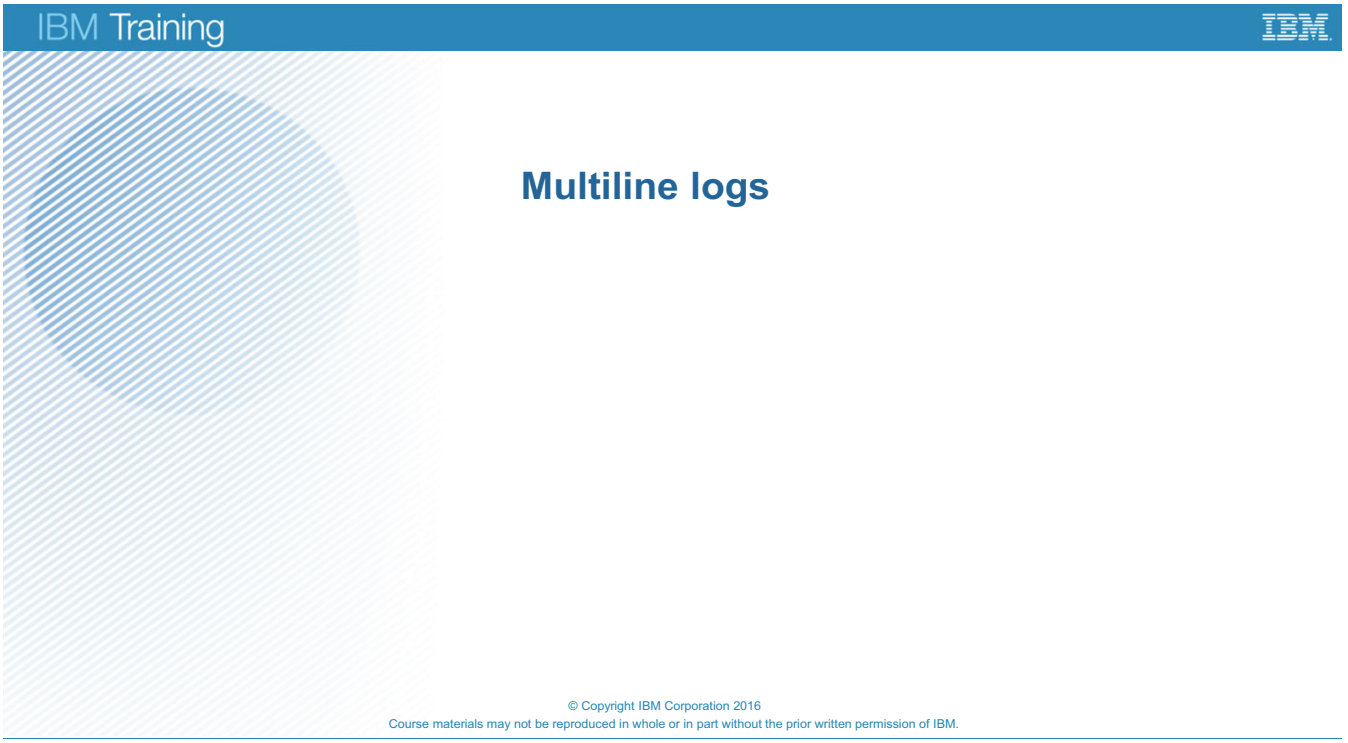

*Exercise introduction: Lab environment in Unit 5 exercises*

In the exercises for this unit, you install and configure the following components:

- A Log File Agent, to monitor a syslog file
- An HAProxy listener for traffic from the Log File Agent (configuration only)
- Your logstashA and logstashB receivers, to accept messages from the Log File Agent (configuration only)
- Your logstashW sender, to pull messages from Kafka and send them to the Log Analysis server (configuration only)
- A data source in Log Analysis, to accept data from the target log file (configuration only)

As you install and configure these components, you also verify the data flow between them.

# *Unit 6* Multiline logs




In this unit, you learn how to configure Logstash receivers, Logstash senders, and HAProxy load balancers to process multiline logs.

IBM Training

## Unit objectives

- Configure Logstash for multiline logs
- Configure HAProxy for multiline logs
- Configure Logstash senders to send messages from the same log file together
- Diagram data flow for multiline logs




*Unit objectives*

IBM Training

## Multiline log example

```
2016-05-16-13.23.45.628104+000 E166427E271         LEVEL: Severe
PID     : 5731                 TID  : 139822525732608PROC : db2star2
INSTANCE: db2inst1             NODE : 000
FUNCTION: DB2 UDB, oper system services, sqloKAAnalyze, probe:10
DATA #1 : Codepath, 8 bytes
3

2016-05-16-13.23.45.858552+000 I166699E491         LEVEL: Warning
PID     : 5731                 TID  : 139822525732608PROC : db2star2
INSTANCE: db2inst1             NODE : 000
FUNCTION: DB2 UDB, SQO Memory Management, SqloMemController::calculateAutomaticSiz,
probe:130
MESSAGE : AUTOMATIC instance_memory setting decreased due to licensing
          restrictions.
DATA #1 : Size in bytes, PD_TYPE_MEM_SIZE, 8 bytes
6900779035
DATA #2 : Size in bytes, PD_TYPE_MEM_SIZE, 8 bytes
4294967296
```

Multiline logs



*Multiline log example*

Many log files are in a multiline format. Multiline log records are single log messages that span multiple lines. To process multiline logs in the scalable collection architecture, you must ensure that all of the lines in a single message are processed together and in the correct sequence.

IBM Training

## Logstash and multiline logs

- Logstash instances that processes multiline logs must run in single-threaded mode, which forces Logstash to processes each line of a single message in the correct sequence.
- A good practice is to use different Logstash instances for single-line and multiline logs. This suggestion applies to Logstash receivers and senders.
- Use the `-w 1` option when you start Logstash to run it in single-threaded mode, for example:

```
/opt/logstash-2.2.1/bin/logstash -w 1 -f /opt/logstash-2.2.1/conf/logstash-
multi.conf -l /opt/logstash-2.2.1/log/logstash-multi-debug.log &
```




*Logstash and multiline logs*

Multiline Logstash receivers must run in single-threaded mode so that they process each line of a single message in the correct sequence. This configuration helps to ensure that the lines in the log file are processed sequentially, maintaining order between the records.

So be sure to install Logstash receivers and Logstash senders that are dedicated to multiline messages.

IBM Training

## HAProxy and multiline logs

Configure HAProxy with affinity for collection agents based on their IP information.

- The `balance source` and `hash-type consistent` lines ensure that all traffic from a collection agent is sent to the same Logstash instance, as long as that Logstash instance is up.
- This is important for multiline logs because you want all lines from the same log message to go to the same Logstash instance.

haproxy.cfg

```
## Listener for LFA Multiline cluster
## Specify Logstash instances for processing multiline messages from LFA
listen  Multiline_LFA_Cluster ha-proxy.csite.ibm.edu:5980
        mode tcp
        balance source
        hash-type consistent
        server receiver-logstashC logstashC.csite.ibm.edu:5990 check fall 2 rise 3 inter 1000
        server receiver-logstashD logstashD.csite.ibm.edu:5991 check fall 2 rise 3 inter 1000
```




*HAProxy and multiline logs*

For multiline log files, configure HAProxy with affinity for log collection agents based on their IP information.

IBM Training

## Sending data to Log Analysis with Logstash senders

Use the `metadata_fields` option in the scala output plug-in to use a unique identifier for each target log file.

Use a value for `field_names` and `field_paths` that uniquely identifies the physical log data source within your entire environment.

This sample configuration ensures that messages with the same `resourceID` are sent from the Logstash sender to the Log Analysis server together in the same batches.

The `host@path` line specifies which logical data source within Log Analysis messages with the same `resourceID` are mapped to.

```
output {

  scala {
      scala_url =>
      "https://log-analysis.csite.ibm.edu:9987/Unity/DataCollector"
      scala_user => "unityadmin"
      scala_password => "object00"
      scala_keystore_path => ""
      batch_size => 500000
      idle_flush_time => 5
      sequential_flush => true
      num_concurrent_writers => 20
      use_structured_api => false
      disk_cache_path => "/opt/logstashX/training/cache/basecache"
      date_format_string => "yyyy-MM-dd'T'HH:mm:ssX"
      log_file => "/opt/logstashX/logstash-2.2.1/log/scala_logstashX.log"
      log_level => "info"
       metadata_fields => {
         "TEST_db2diag@LabDB2Instance_db2diag" => {
           "field_names" => "resourceID"
           "field_paths" => "resourceID"
         } # end db2diag meta data
       } # end meta data fields

  }#end scala output

}# end output section
```

Multiline logs



*Sending data to Log Analysis with Logstash senders*

Logstash senders pull messages from Kafka, process the log messages, and send them to the Log Analysis core software.

Logstash senders that process multiline messages must use the **metadata_fields** option in the scala output plug-in.

Notice the following lines in this example scala output plug-in configuration.

```
metadata_fields => {
  "TEST_db2diag@LabDB2Instance_db2diag" => {
    "field_names" => "resourceID"
    "field_paths" => "resourceID"
```

The **metadata_fields** option in the Logstash sender ensures that messages with the same **resourceID** are sent from the Logstash sender to the Log Analysis server together in the same batches. This is important for multiline logs, because multiple lines from the same log message must arrive in sequence and only with other lines from the same actual log file.

Configure the value of **resourceID** to be a string that uniquely identifies the physical log source within your entire environment.

IBM Training

## Metadata and multiline messages

In this example, the value of resourceID is `collection_/opt/db2inst1/sqllib/db2dump/db2diag.log_1`

**From a multiline Logstash receiver filter section:**

```
mutate {
  replace => ["message","%{LFA_ORIG_MSG}"]
  add_field => [ "datasource", "%{LFA_INSTANCE}_%{LFA_MODULE}"]
  add_field => [ "resourceID", "%{LFA_HOSTNAME}_%{LFA_LOGNAME}_1"]
```

**From a multiline Logstash sender filter section:**

```
if "grok_lfa" in [tags] {
  mutate {
    add_field => [ "path", "%{LFA_INSTANCE}_%{LFA_MODULE}" ]
    replace => ["host", "%{LFA_ENVIRONMENTNAME}_%{LFA_MODULE}"]
  } # end mutate
} # end lfa mutate condition
```

**From a multiline Logstash sender output section:**

```
metadata_fields => {
        "TEST_db2diag@LabDB2Instance_db2diag" => {
          "field_names" => "resourceID"
          "field_paths" => "resourceID"
        } # end db2diag meta data
      } # end meta data fields
```

Multiline logs



*Metadata and multiline messages*

Look at the following lines in the Logstash sender output section. Notice that **field_names** and **field_paths** use the value of **resourceID**.

```
metadata_fields => {
  "TEST_db2diag@LabDB2Instance_db2diag" => {
    "field_names" => "resourceID"
    "field_paths" => "resourceID"
```

The values for host (`TEST_db2diag`) and path (`LabDB2Instance_db2diag`) are set by the Logstash sender filter section.

The value of **resourceID** is set with a filter in the multiline Logstash receiver configuration.

Example multiline Logstash receiver filter configuration:

```
mutate {
    replace => ["message","%{LFA_ORIG_MSG}"]
    add_field => [ "datasource", "%{LFA_INSTANCE}_%{LFA_MODULE}"]
    add_field => [ "resourceID", "%{LFA_HOSTNAME}_%{LFA_LOGNAME}_1"]
```

In this example, the value of **resourceID** is:
`collection_/opt/db2inst1/sqllib/db2dump/db2diag.log_1`.

You also use **resourceID** as the Kafka message key in your Logstash receiver output configuration.

Example of a multiline Logstash receiver output configuration:

```
kafka {
    bootstrap_servers =>"kafka.csite.ibm.edu:17991"
    topic_id => "%{datasource}"
    message_key => "%{resourceID}"
} #end Kafka output
```

The message key ensures that the Logstash sender can identify messages from Kafka for this specific physical data source, even if messages from other physical data sources are in the same Kafka topic and partition.

Configure the value of `resourceID` in the Logstash receiver filter section to be a string that uniquely identifies the physical log source within your entire environment.

IBM Training

## Unit summary

- Configure Logstash for multiline logs
- Configure HAProxy for multiline logs
- Configure Logstash senders to send messages from the same log file together
- Diagram data flow for multiline logs




*Unit summary*

IBM Training

## Multiline logs exercises

**Exercise 1 Configuring the Log File Agent**
**Exercise 2 Configuring HAProxy**
**Exercise 3 Installing and configuring the first multiline Logstash receiver**
**Exercise 4 Installing and configuring the second multiline Logstash receiver**
**Exercise 5 Installing and configuring the multiline Logstash sender**
**Exercise 6 Sending data to Log Analysis**

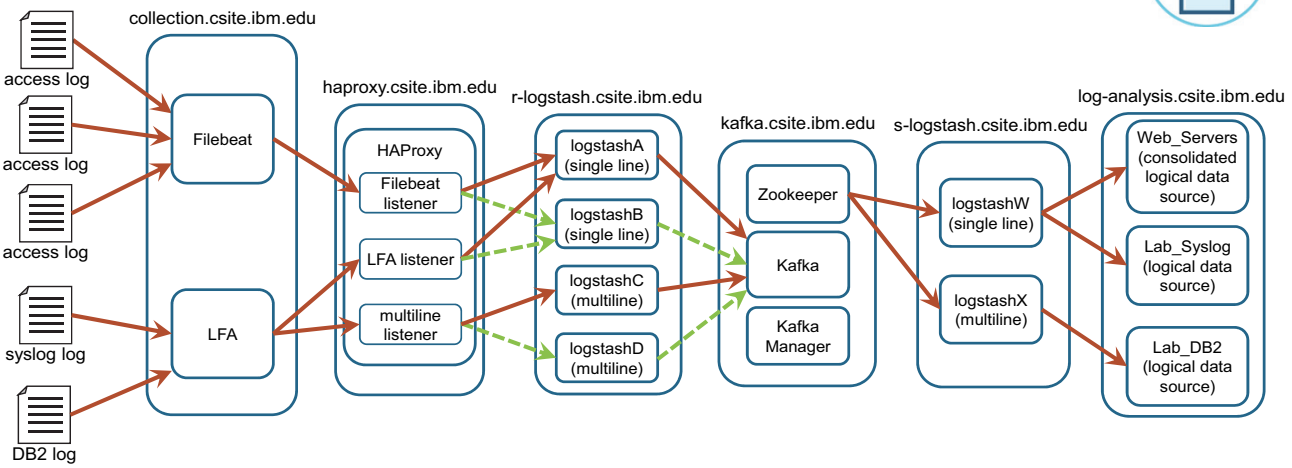



*Student exercises*

Perform the exercises for this unit.

IBM Training

## Exercise introduction: Finished lab environment

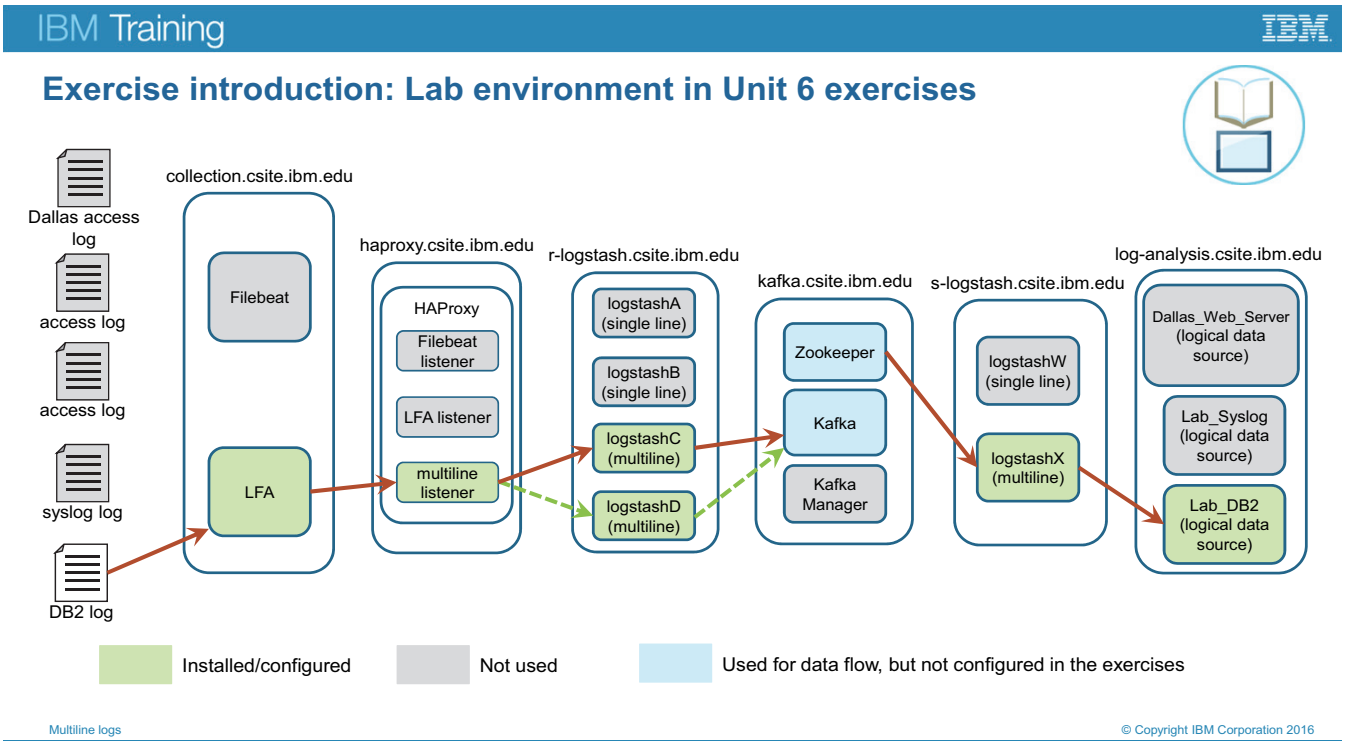



Multiline logs



*Exercise introduction: Finished lab environment*

At the end of this course, your lab environment will look like this diagram. Refer to this diagram as you complete the steps in the student exercises to help you understand what you are installing and configuring.




*Exercise introduction: Lab environment in Unit 6 exercises*

In the exercises for this unit, you install and configure the following components:

- A Log File Agent, to monitor a DB2 log (configuration only)
- An HAProxy listener for traffic from the Log File Agent (configuration only)
- Two multiline Logstash receivers, to accept messages from the Log File Agent
- A multiline Logstash sender, to pull messages from Kafka and send them to the Log Analysis server
- A data source in Log Analysis, to accept data from the target log file (configuration only)

As you install and configure these components, you also verify the data flow between them.

# *Unit 7* Log consolidation


This unit teaches you how to consolidate messages from many logs into just a few Log Analysis data sources.

IBM Training

## Unit objectives

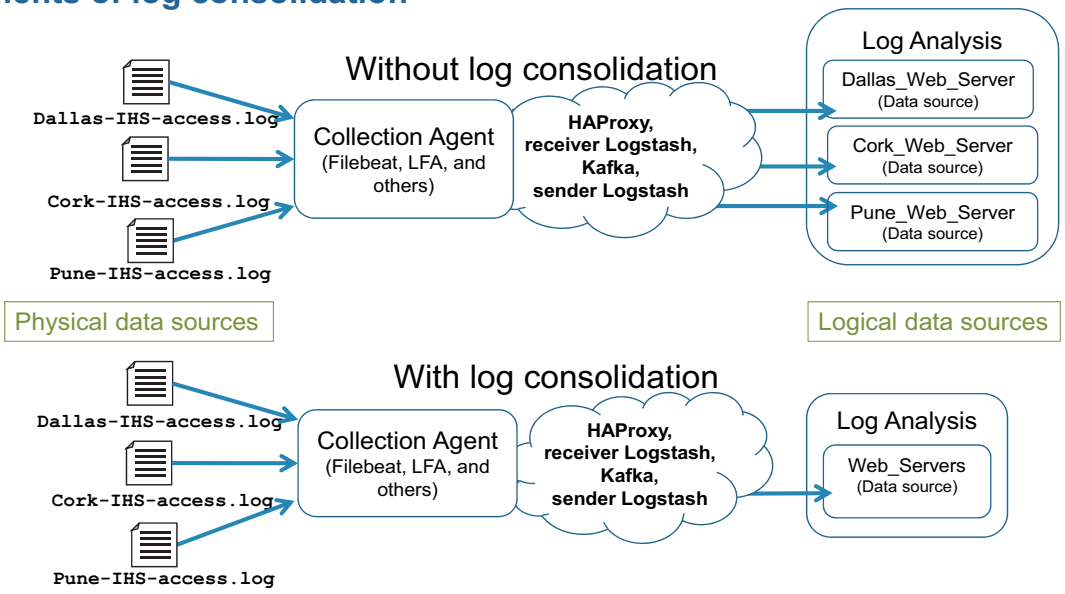
- Diagram the data flow for consolidated logs
- Configure a custom source type and index configuration
- Configure log collection agents to add metadata
- Configure Logstash senders to add metadata




*Unit objectives*

IBM Training

## Benefits of log consolidation




Log consolidation



*Benefits of log consolidation*

In a simple configuration, IBM Log Analysis manages input from a single target log file with a single data source. Environments with many log files to monitor (hundreds or thousands) require just as many data sources. It can be problematic for administrators to create, manage, navigate, and support so many data sources.

With log collection agents and Logstash, you can consolidate logs of the same type in to a single Log Analysis data source. Consolidation makes many tasks easier for several roles:

- Administrators can more easily add and manage data sources.
- Users can search through multiple log files simultaneously.

This diagram shows two different configurations: an environment without log consolidation and an environment where log consolidation is implemented.

In the environment without log consolidation, three different Log Analysis data sources are necessary to accept messages from three different web server access log files.

In the environment where log consolidation is implemented, a single Log Analysis data source is accepting messages from the three different web server access log files.

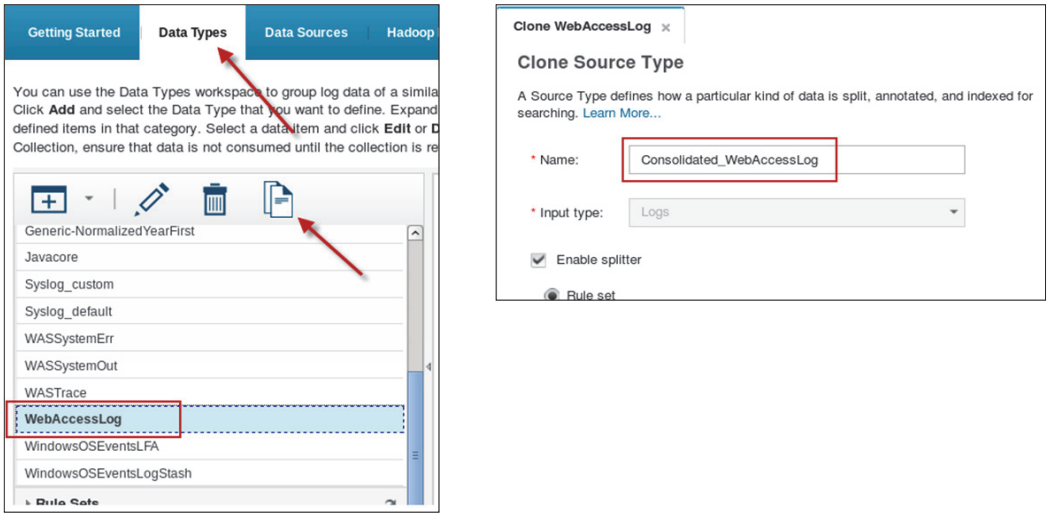
This diagram also illustrates the difference between physical data sources and logical data sources:

- Physical data sources are the actual target log files, such as `Dallas-IHS-access.log`, `Pune-IHS-access.log`, or `Cork-IHS-access.log`.
- Logical data sources are administrative objects that you create with the Log Analysis administrator interface, such as Web_Servers.

IBM Training

## Adding a custom source type

Use the Log Analysis administrator interface to clone an existing source type.




Log consolidation



*Adding a custom source type*

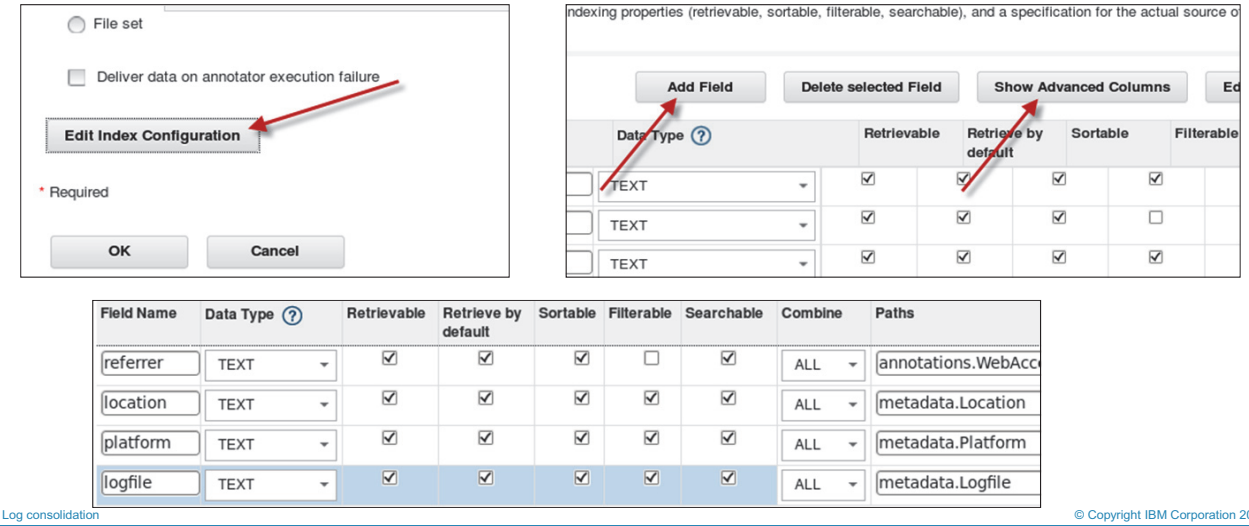
To consolidate logs, you must create a custom index configuration to accept extra metadata about each log file. An index configuration is an administrative object within a Log Analysis source type. To create a custom index configuration, you must create a custom source type. Use the Log Analysis administrator user interface to add a custom source type.

To create a custom source type, create a copy of the source type for the log format you want to consolidate. In this example, you want to consolidate web access log files, so you create a copy of the existing WebAccessLog source type.

IBM Training

## Editing the index configuration

Within the new source type, edit the index configuration and add fields that identify the physical data source. In this example, the new fields are **location**, **platform**, and **logfile**.




Log consolidation



*Editing the index configuration*

An index configuration determines which fields in each log message is indexed, and how that indexed data can be used in search queries. The Log Analysis administrator user interface includes a tool that you can use to create a custom index configuration.

In this example, you want to index three new fields in your consolidated log messages:

- **location**
- **platform**
- **logfile**

The following table explains the properties that are required for each field.

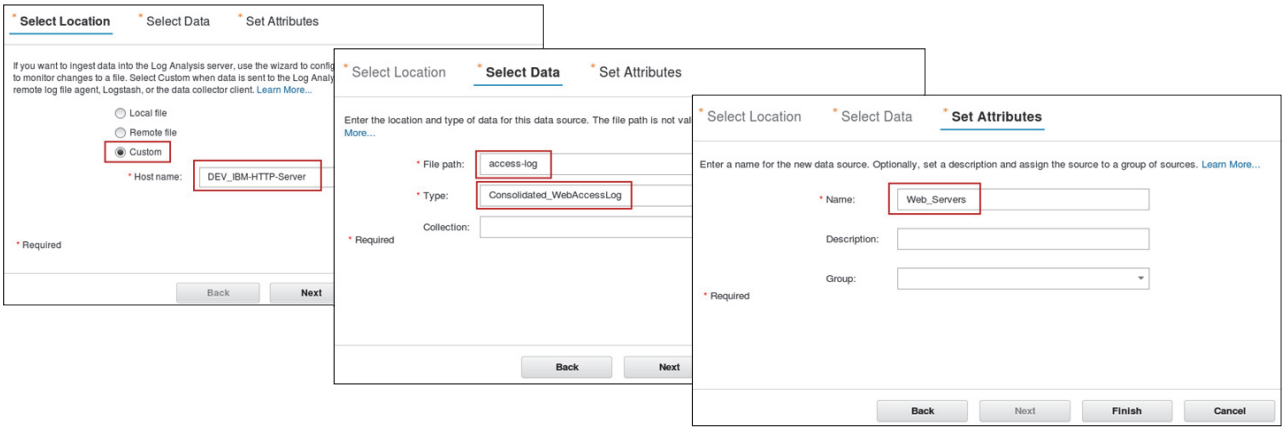
| Property | Description |
|---|---|
| Field name | The name of the field that users see in their search results. |
| Data type | The type of data in the field. Possible values are TEXT, LONG, DOUBLE, and DATE. |
| Retrievable | Determines whether the contents of this field are stored for retrieval. When set to false, the content is not stored in the index. When set to True, the content is stored and available for retrieval. |

| Property | Description |
|---|---|
| RetrieveByDefault | When set to True, the contents of the field is always returned as part of any search response. When set to False, the field is not part of the default response. However, when required, the user can explicitly query the content of the field. The retrievable option must be set to True for this attribute to work. |
| Sortable | Enable or disable the field for sorting and range queries. |
| Filterable | Enable or disable facet counting and filtering on this field. |
| Searchable | Controls whether the field is enabled for searching and matching. |
| Combine | Determines how the values that are returned by the paths and dateFormats attributes are used. Possible values are ALL and FIRST. |
| Paths | JSON path expression. For indexing metadata in the field, use the string `metadata.` followed by a name that describes the metadata. |

IBM Training

## Creating a logical data source

- Use the Log Analysis administrator user interface to add a data source.
- Use the host and path that were set by the Logstash sender, not the actual path of the target log files.
- Use the new source type as the **Type**.







*Creating a logical data source*

After you create a custom source type, create a new data source to accept messages from the consolidate log files. Use the name of your custom source type in the **Type** field of the data source wizard.

IBM Training

## Configuring the collection agent: Filebeat

Add metadata to each log that the collection agent is monitoring.

In this example, metadata for site and platform are added in the `filebeat.yml` file.

```
paths:
   - /opt/IBM/HTTPServer/logs/Dallas-IHS-access.log
  input_type: log
  fields:
   collector: filebeats-collection.csite.ibm.edu
   env: DEV
   module: IBM-HTTP-Server
   type: access-log
   site: DALLAS
   platform: RHEL
-
  paths:
   - /opt/IBM/HTTPServer/logs/Pune-IHS-access.log
  input_type: log
  fields:
   collector: filebeats-collection.csite.ibm.edu
   env: DEV
   module: IBM-HTTP-Server
   type: access-log
   site: PUNE
   platform: SLES
-
  paths:
   - /opt/IBM/HTTPServer/logs/Cork-IHS-access.log
  input_type: log
  fields:
   collector: filebeats-collection.csite.ibm.edu
   env: DEV
   module: IBM-HTTP-Server
   type: access-log
   site: CORK
   platform: RHEL
```

filebeat.yml




*Configuring the collection agent: Filebeat*

When users search a consolidated data source, the search results show messages from multiple log files. Metadata that you add to each log message can show users exactly which messages came from which log file.

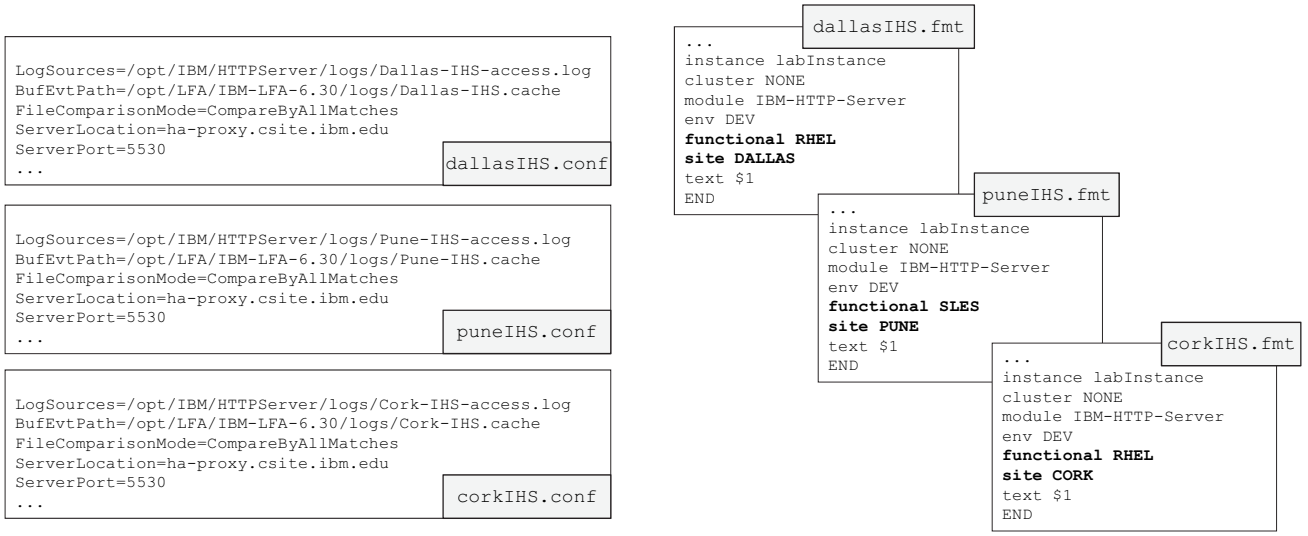
When you consolidate logs, you must include metadata so that users can determine which log file a message in the search interface came from. Use the log collection agent to add metadata to log messages.

In this example configuration, Filebeat adds two metadata fields that you can include in user search results: site and platform. You configure Filebeat by editing the `filebeat.yml` file.

IBM Training

## Configuring the collection agent: Log File Agent

In this example, metadata for site and platform are added in the .conf and .fmt file for each target log file.




*Configuring the collection agent: Log File Agent*

In this example configuration, the Log File Agent adds to metadata fields that you can include in user search results: **site** and **platform**. You configure the Log File Agent to add metadata by editing the .fmt file.

**Hint:** Notice that the values for **platform** (`RHEL` and `SLES`) are in a field named **functional**. Remember that the grok filter in a Logstash receiver configuration uses a pattern named LFAMESSAGE to match meaningful strings of text from the Log File Agent. This pattern matches the value of the **functional** field. That is why this example uses **functional** to store a value for operating system platform. If you to create a different metadata field in the .fmt file, for example, **platform**, you must change the LFAMESSAGE grok pattern to match the new field.

IBM Training

## Logstash sender filter section

```
filter {

    if "mutate_filebeat" in [tags] {
        mutate {
          add_field => [ "path", "%{[fields][type]}" ]
          replace => { "host" => "%{[fields][env]}_%{[fields][module]}" }
        } # end mutate
    } # end filebeat mutate condition

    if [fields][type] == "access-log" {
        mutate {
          add_field => [ "Location", "%{[fields][site]}" ]
          add_field => [ "Platform", "%{[fields][platform]}" ]
          add_field => [ "Logfile", "%{resourceID}"]
        } # end web server mutate
    } # end web server mutate condition

}# end filter section
```




*Logstash sender filter section*

To send metadata that was added by the log collection agents, use the Logstash sender. In this example, the Logstash sender filter section is adding three new fields to store metadata from a Filebeat collection agent:

- **Location**, to store values from the Filebeat **site** field such as DALLAS, PUNE, or CORK.
- **Platform**, to store values from the Filebeat **platform** field such as RHEL or SLES.
- **Logfile**, to store a unique identifier for each target log file. This example uses the value of **resourceID**.

IBM Training

## Logstash sender output section

```
output {
    scala {
        scala_url => "https://log-analysis.csite.ibm.edu:9987/Unity/DataCollector"
        scala_user => "unityadmin"
        scala_password => "object00"
        scala_keystore_path => ""
        batch_size => 500000
        idle_flush_time => 5
        sequential_flush => true
        num_concurrent_writers => 20
        use_structured_api => false
        disk_cache_path => "/opt/logstashW/training/cache/basecache"
        date_format_string => "yyyy-MM-dd'T'HH:mm:ssX"
        log_file => "/opt/logstashW/logstash-2.2.1/log/scala_logstashW.log"
        log_level => "info"
        metadata_fields => {
          "DEV_IBM-HTTP-Server@access-log" => {
            "field_names" => "Location,Platform,Logfile"
            "field_paths" => "Location,Platform,Logfile"
          }
        }
    }#end scala output
}# end output section
```

| Field Name | Data Type | Retrievable | Retrieve by default | Sortable | Filterable | Searchable | Combine | Paths |
|---|---|---|---|---|---|---|---|---|
| referrer | TEXT | ☑ | ☑ | ☑ | ☐ | ☑ | ALL | annotations.WebAcc |
| location | TEXT | ☑ | ☑ | ☑ | ☑ | ☑ | ALL | metadata.Location |
| platform | TEXT | ☑ | ☑ | ☑ | ☑ | ☑ | ALL | metadata.Platform |
| logfile | TEXT | ☑ | ☑ | ☑ | ☑ | ☑ | ALL | metadata.Logfile |

Log consolidation



*Logstash sender output section*

To send metadata that was added by the log collection agents, use the metadata_fields option in the scala output plug-in.

## *Scala output plug-in configuration example*

In the following example, line numbers have been added to the scala output plug-in configuration for clarity.

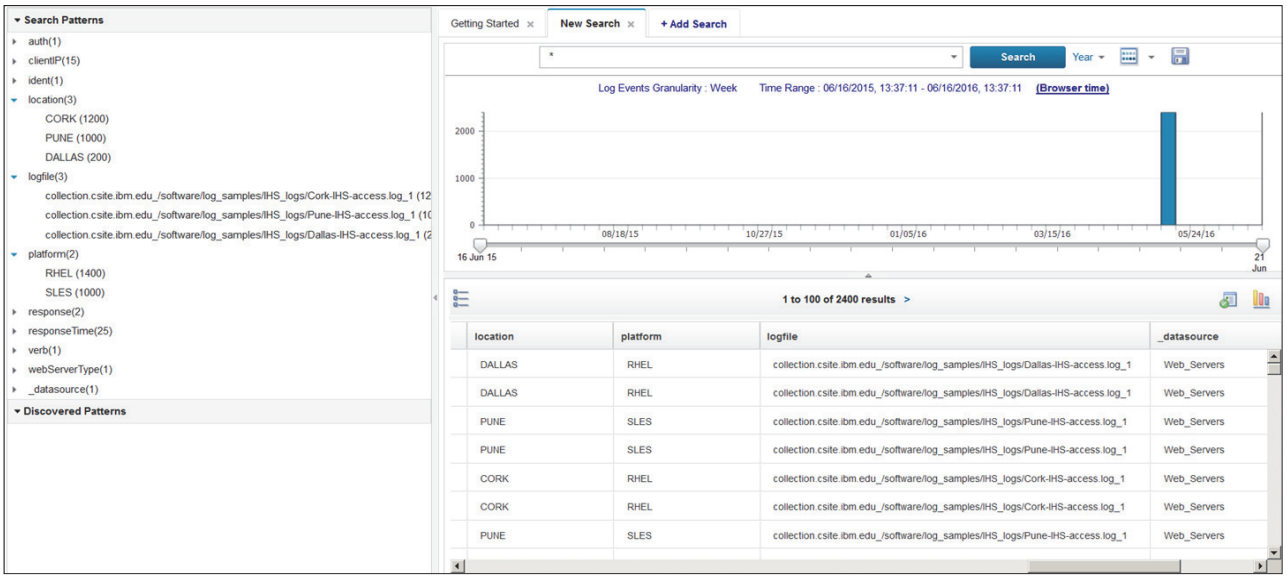
```
1 metadata_fields => {
2     "DEV_IBM-HTTP-Server@access-log" => {
3         "field_names" => "Location,Platform,Logfile"
4         "field_paths" => "Location,Platform,Logfile"
5     } # end web server meta data
6 } # end meta data fields
```

- Line 1 sends extra metadata fields to the Log Analysis server.
- Line 2 specifies the Log Analysis data source that the metadata is sent to. The data source is specified by its host and path. The format is host@path.
- Line 3 is a comma-delimited list of field names from the message metadata that you want to send to Log Analysis. You add these fields in the filter section of the Logstash sender.
- Line 4 is a comma-delimited list of the paths that you created in your custom index configuration. In this example, three fields were added in the index configuration: **metadata.Location**, **metadata.Platform**, and **metadata.Logfile**.

- Line 5 closes the configuration for metadata fields that are sent to the `DEV_IBM-HTTP-Server@access-log` data source.
- Line 6 closes the configuration for the metadata_fields option.

IBM Training

## Searching consolidated logs







*Searching consolidated logs*

When users search a consolidated data source, the search results show messages from multiple log files. Metadata that you add to each log message can show users exactly which messages came from which log file.

In this example, users can filter the search results for the consolidated data source by location, operating system platform, or by the actual log file the message results came from.

IBM Training

## Unit summary

- Diagram the data flow for consolidated logs
- Configure a custom source type and index configuration
- Configure log collection agents to add metadata
- Configure Logstash senders to add metadata




*Unit summary*

IBM Training

## Log consolidation exercises

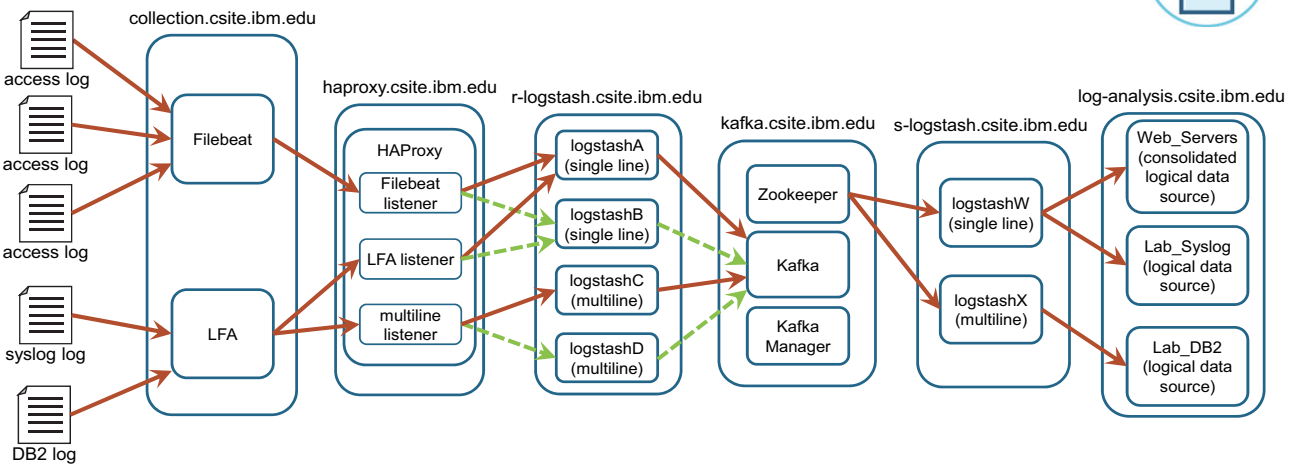
**Exercise 1 Deleting the current Log Analysis data source**
**Exercise 2 Adding a source type and a new data source**
**Exercise 3 Configuring the Logstash sender**
**Exercise 4 Configuring Filebeat**
**Exercise 5 Verifying log consolidation**




*Student exercises*

IBM Training

## Exercise introduction: Finished lab environment

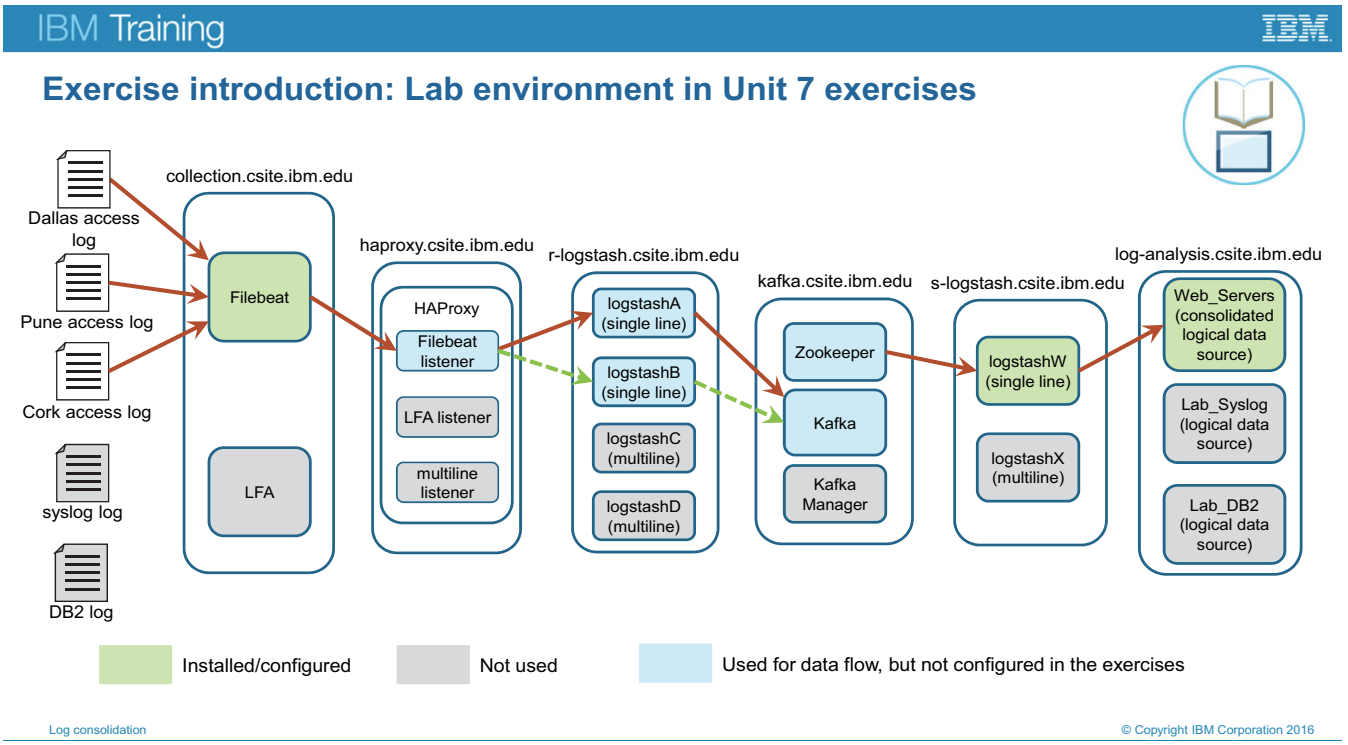



Log consolidation



*Exercise introduction: Finished lab environment*

At the end of this course, your lab environment will look like this diagram. Refer to this diagram as you complete the steps in the student exercises to help you understand what you are installing and configuring.




*Exercise introduction: Lab environment in Unit 7 exercises*

In the exercises for this unit, you configure the following components:

- Your Filebeat collection agent, to monitor two more web server log files (configuration only)
- Your logstashW sender, to send metadata to the Log Analysis server (configuration only)
- A custom source type and index configuration in Log Analysis, to index the extra metadata from Filebeat (configuration only)
- A data source in Log Analysis, to accept data from the target log files (configuration only)

As you configure these components, you also verify the data flow between them.

# *Unit 8* Troubleshooting


In this unit, you learn how to troubleshoot each of the components in the scalable collection architecture.

IBM Training

## Unit objectives

- Troubleshoot Filebeat
- Troubleshoot the Log File Agent
- Troubleshoot HAProxy
- Troubleshoot Logstash
- Troubleshoot Kafka
- Troubleshoot Log Analysis




*Unit objectives*

IBM Training

## Filebeat logging

- Filebeat logging is configured in the `filebeat.yml` file.
- Filebeat log levels are: critical, error, warning, info, debug

filebeat.yml

```
...
logging:
 to_files: true
 files:
  path: /opt/filebeat-1.1.1-x86_64/
  name: mybeat
  rotateeverybytes: 10485760
 level: debug
```




*Filebeat logging*

The `filebeat.yml` file controls logging for Filebeat. Use the `level` property to set the log level. You must restart Filebeat for any changes to take effect.

IBM Training

## Filebeat file processing

Messages like these indicate that Filebeat is reading the target log file correctly:

```
2016-06-16T17:55:54Z DBG  full line read

2016-06-16T17:55:59Z DBG  Publish: {
...
    "env": "DEV",
    "module": "IBM-HTTP-Server",
    "platform": "RHEL",
    "site": "CORK",
    "type": "access-log"
  },
  "input_type": "log",
  "message": "Apache/IHS,10.10.255.90,-,-,[10/May/2016:14:04:42 -0400],GET,\"GET
//app?action=sell\u0026holdingID=5015 HTTP/1.1\",200,3409,1090,\"-\",\"curl/7.21.3 (x86_64-
linux-gnu) libcurl/7.21.3 OpenSSL/1.0.0 zlib/1.2.3\"",
  "offset": 280044,
  "source": "/software/log_samples/IHS_logs/Cork-IHS-access.log",
  "type": "log"
```

Troubleshooting



*Filebeat file processing*

These messages confirm that Filebeat is reading the target log file and trying to output log messages.

IBM Training

## Filebeat event publishing

Messages like these indicate that Filebeat is successfully sending messages:

```
2016-06-16T18:27:41Z DBG  710 events out of 952 events sent to logstash. Continue
sending ...
2016-06-16T18:27:41Z DBG  Try to publish 242 events to logstash with window size 952
2016-06-16T18:27:42Z DBG  242 events out of 242 events sent to logstash. Continue
sending ...
2016-06-16T18:27:42Z DBG  send completed
2016-06-16T18:27:42Z INFO Events sent: 952
2016-06-16T18:27:42Z DBG  Processing 952 events
```




*Filebeat event publishing*

These messages confirm that Filebeat is successfully sending messages to a Logstash receiver. Notice that the number of events in the window decreases as Filebeat sends messages to the Logstash receiver.

IBM Training

## Filebeat connectivity

Messages like these indicate that HAProxy is not responding:

```
2016-06-16T17:55:59Z DBG  output worker: publish 200 events
2016-06-16T17:55:59Z DBG  Try to publish 200 events to logstash with window size 120
2016-06-16T17:55:59Z DBG  close connection
2016-06-16T17:55:59Z DBG  0 events out of 200 events sent to logstash. Continue sending ...
2016-06-16T17:55:59Z INFO Error publishing events (retrying): EOF
2016-06-16T17:55:59Z INFO send fail
2016-06-16T17:56:14Z INFO Connecting error publishing events (retrying): dial tcp
192.168.100.176:20737: getsockopt: connection refused
```

Messages like these indicate that the Logstash receiver cluster is not responding:

```
2016-06-16T18:09:12Z DBG  Try to publish 200 events to logstash with window size 30
2016-06-16T18:09:12Z DBG  close connection
2016-06-16T18:09:12Z DBG  0 events out of 200 events sent to logstash. Continue sending ...
2016-06-16T18:09:12Z INFO Error publishing events (retrying): write tcp 192.168.100.175:42323-
>192.168.100.176:20737: write: broken pipe
2016-06-16T18:09:12Z INFO send fail
```

Troubleshooting 

*Filebeat connectivity*

The first set of messages show that Filebeat cannot connect to HAProxy. Notice the `tcp connection refused` message.

The second set of messages show that Filebeat cannot connect to any Logstash receivers on the other side of HAProxy. Notice the `broken pipe` message.

IBM Training IBM

## Log File Agent logging

- Log File Agent logging is configured in the `<LFA_HOME>/IBM-LFA-6.30/config/lo_default_workload_instance.config` file.
- Log levels are explained in the configuration file.

```
# ================================================================
DEFAULTCFG='NO'
KBB_SIG1='-asyncoff -syncoff -dumpoff'
KBB_RAS1='ERROR (UNIT:logmonitor ALL)'
KBB_VARPREFIX='%'
KBB_RAS1_LOG='%(CTIRA_LOG_PATH)/collection_lo_default_workload_instance_%(systask)_%(sysutcstart)-.log
INVENTORY=%(CTIRA_LOG_PATH)/collection_lo_default_workload_instance_%(systask).inv COUNT=03 LIMIT=5 PRESERVE=1
MAXFILES=9'
KDC_FAMILIES='ip.pipe  HTTPS_CONSOLE:N HTTPS_SERVER:N HTTPS:0'
LOGSHOME='/opt/LFA/IBM-LFA-6.30/logs'
```

Troubleshooting 

*Log File Agent logging*

The lo_default_workload_instance.config file controls logging for the Log File Agent. Use the `KBB_RAS1` property to set the log level. You must restart the Log File Agent for any changes to take effect.

IBM Training

## Log File Agent log file

Logs are in the `<LFA_HOME>/IBM-LFA-6.30/logs` directory.

Look for the most recent log file name with the format:
collection_lo_default_workload_instance_kloagent_NNNNNNNN-NN.log.

```
-rw-r--r-- 1 netcool ncoadmin       0 Jun 16 19:09 collection_lo_1466104169.pid6382
-rwxr-xr-x 1 netcool ncoadmin   41866 Jun  8 19:02 collection_lo_default_workload_instance_kloagent_57584219-01.log
-rw-r--r-- 1 netcool ncoadmin   38815 Jun 16 18:58 collection_lo_default_workload_instance_kloagent_5762a250-01.log
-rw-r--r-- 1 netcool ncoadmin 5206756 Jun 16 19:00 collection_lo_default_workload_instance_kloagent_5762f6eb-01.log
-rw-r--r-- 1 netcool ncoadmin 2764151 Jun 16 19:04 collection_lo_default_workload_instance_kloagent_5762f6eb-02.log
-rw-r--r-- 1 netcool ncoadmin 5167405 Jun 16 19:05 collection_lo_default_workload_instance_kloagent_5762f837-01.log
-rw-r--r-- 1 netcool ncoadmin 4198253 Jun 16 19:09 collection_lo_default_workload_instance_kloagent_5762f837-02.log
-rw-r--r-- 1 netcool ncoadmin 5162248 Jun 16 19:07 collection_lo_default_workload_instance_kloagent_5762f837-03.log
-rw-r--r-- 1 netcool ncoadmin   36663 Jun 16 19:10 collection_lo_default_workload_instance_kloagent_5762f969-01.log
-rwxr-xr-x 1 netcool ncoadmin     736 Jun 16 19:09 collection_lo_default_workload_instance_kloagent.inv
-rw-r--r-- 1 netcool ncoadmin    2826 Jun 16 19:09 default_workload_instance:collec.LG0
-rw-r--r-- 1 netcool ncoadmin    2826 Jun 16 19:04 default_workload_instance:collec.LG1
```

Troubleshooting



*Log File Agent log file*

Log File Agent logs are in the `<LFA_HOME>/IBM-LFA-6.30/logs` directory. The name of the log files in this directory is `collection_lo_default_workload_instance_kloagent`, followed by a time stamp. To find the file that the Log File Agent is currently logging to, look for the file with the most recent date.

IBM Training

## Log File Agent event publishing

Messages like these indicate that the Log File Agent is successfully sending messages to HAProxy:

```
(5762A82A.0001-C:sockeif.c,1209,"_imp_eipc_recv_data") KDE1_ReceiveOn returned 0x1DE0000B
(5762AD1F.0000-C:sockeif.c,1209,"_imp_eipc_recv_data") KDE1_ReceiveOn returned 0x1DE0000B
(5762AD1F.0001-C:sockeif.c,1209,"_imp_eipc_recv_data") KDE1_ReceiveOn returned 0x1DE0000B
(5762AD1F.0002-C:sockeif.c,1209,"_imp_eipc_recv_data") KDE1_ReceiveOn returned 0x1DE0000B
(5762AD1F.0003-C:sockeif.c,1209,"_imp_eipc_recv_data") KDE1_ReceiveOn returned 0x1DE0000B
(5762AD1F.0004-C:sockeif.c,1209,"_imp_eipc_recv_data") KDE1_ReceiveOn returned 0x1DE0000B
(5762AE00.0000-C:sockeif.c,1209,"_imp_eipc_recv_data") KDE1_ReceiveOn returned 0x1DE0000B
```




*Log File Agent event publishing*

This set of messages show that the Log File Agent is successfully reading messages from the target log file and sending them to HAProxy.

IBM Training

## Log File Agent connectivity

Messages like these indicate that HAProxy is not responding:

```
(5762FCEE.0000-B:sockeif.c,390,"_imp_connect") KDE1 connection returned 0x1DE00045
errno 107 for ha-proxy.csite.ibm.edu port 5530

(5762FCEE.0001-B:sockeif.c,537,"_imp_eipc_create_remote_client") Cannot connect to ha-
proxy.csite.ibm.edu<192.168.100.176> port 5530, rc -1
```




*Log File Agent connectivity*

This set of messages show that the Log File Agent cannot connect to HAProxy.

IBM Training

## HAProxy logging

- HAProxy logging is configured in the `/etc/haproxy/haproxy.cfg` file.
- HAProxy writes to syslog or rsyslog.
- HAProxy log levels are: emerg, alert, crit, err, warning, notice, info, debug.
- The tcplog option enables advanced logging of TCP connections with session state.

haproxy.cfg

```
global
    log         127.0.0.1 local0 debug
    #log        127.0.0.1 local1
    #log        127.0.0.1 local2
    #log        127.0.0.1 local3
...
#---------------------------------------------------------------------
-
# common defaults that all the 'listen' and 'backend' sections will
# use if not designated in their block
#---------------------------------------------------------------------
-
defaults
    mode                    tcp
    log                     global
    option                  tcplog
...
```

Troubleshooting



*HAProxy logging*

The `haproxy.cfg` file controls logging for HAProxy. Use the global `log` property to set the log level. You must restart HAProxy for any changes to take effect.

In the global defaults section of haproxy.cfg, the `tcplog` option configures HAProxy to record a detailed message about each TCP connection that it handles.

IBM Training

## HAProxy TCP session logging

Messages like these are details about each TCP connection.

```
Jun 16 19:48:04 localhost haproxy[4271]: 192.168.100.175:53237 [16/Jun/2016:19:48:02.247]
LFA_Cluster LFA_Cluster/receiver-logstashA 1/0/120003 0 sD 0/0/0/0/0 0/0

Jun 16 19:48:07 localhost haproxy[4271]: 192.168.100.175:42343 [16/Jun/2016:19:48:03.777]
Beats_Cluster Beats_Cluster/receiver-logstashB 1/0/63448 30 cD 0/0/0/0/0 0/0

Jun 16 19:48:59 localhost haproxy[4271]: 192.168.100.175:42344 [16/Jun/2016:19:48:12.287]
Beats_Cluster Beats_Cluster/receiver-logstashA 1/1/61013 72 cD 0/0/0/0/0 0/0
```




*HAProxy TCP session logging*

HAProxy can record detailed messages about each TCP session. Each of the messages in this example describes one TCP session. The log message is not written to the log file until the TCP session has finished.

The following table explains the format of a TCP log message.

| Field | Example | Description |
|---|---|---|
| 1 | Jun 16 19:48:59 | Message time stamp |
| 2 | localhost | HAProxy host name |
| 3 | haproxy[4271] | Process name and process ID |
| 4 | 192.168.100.175:42344 | IP address and port number of the client that initiated the TCP connection |
| 5 | [16/Jun/2016:19:48:12.287] | Time stamp when HAProxy received the connection |
| 6 | Beats_Cluster | The listener that is used for the connection |
| 7 | Beats_Cluster/receiver-logstashA | The Logstash receiver that the data was forwarded to |
| 8 | 1/1/61013 | Total wait time / wait time to connect to the Logstash receiver / total transaction time |
| 9 | 72 | Bytes sent from the server to the client |
| 10 | cD | A two-letter code indicating the session termination state |

| Field | Example | Description |
|---|---|---|
| 11 | 0/0/0/0/0 | Number of concurrent HAProxy connections / number of concurrent listener connections / number of concurrent connections to the Logstash receiver / Number of active HAProxy connections at the time of the log message / session retries |
| 12 | 0/0 | Number of session requests before this transaction / Number of session requests before this transaction for the Logstash receiver cluster |

For more information about the format of a TCP log message, refer to the HAProxy documentation at http://www.haproxy.org/download/1.5/doc/configuration.txt.

IBM Training

## HAProxy connectivity

Messages like these indicate that a Logstash server is down.

```
Jun 16 20:27:25 localhost haproxy[4557]: Server Beats_Cluster/receiver-logstashB is DOWN,
reason: Layer4 connection problem, info: "Connection refused", check duration: 0ms. 1 active
and 0 backup servers left. 0 sessions active, 0 requeued, 0 remaining in queue.
Jun 16 20:27:25 localhost haproxy[4557]: Server LFA_Cluster/receiver-logstashB is DOWN,
reason: Layer4 connection problem, info: "Connection refused", check duration: 0ms. 1 active
and 0 backup servers left. 0 sessions active, 0 requeued, 0 remaining in queue.
```

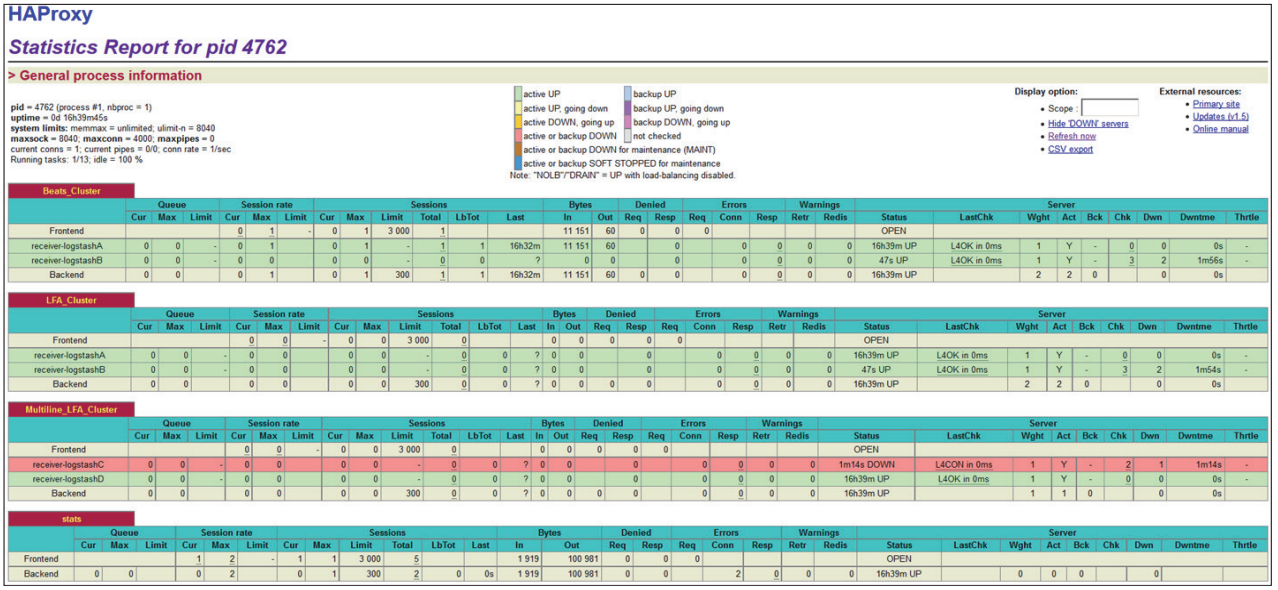



*HAProxy connectivity*

HAProxy performs periodic health checks for the Logstash receivers in the back end cluster. If a Logstash receiver stops responding, HAProxy records that event in its log.

IBM Training

## HAProxy statistics page







*HAProxy statistics page*

You can configure HAProxy to display a web page that shows the status of each server that it is forwarding data to. The page also shows statistics about each of the endpoints in the cluster, such as the volume of data sent. This page is useful for troubleshooting. You configure this web page in the `haproxy.cfg` file.

The following example shows how to enable the HAProxy statistics page in the `haproxy.cfg` file.

```
...
stats socket /var/lib/haproxy/stats
...
listen stats :9000 #Listen on localhost port 9000
    mode http
    stats enable
    stats hide-version
    stats realm Haproxy\ Statistics
    stats uri /haproxy_stats
    stats auth netcool:object00
```

The following table describes each line in the configuration example.

| Configuration option | Description |
|---|---|
| listen stats :9000 | This line configures the statistics page to use port 9000. |
| mode http | This line enables http mode. |
| stats enable | This line enables statistics. |

| Configuration option | Description |
|---|---|
| stats hide-version | This line hides the version of HAProxy so it is not visible on the web page. |
| stats realm Haproxy\ Statistics | This text is what users see when they are prompted to log in to the page. |
| stats uri /haproxy_stats | The URI of the web page, for example, `http://localhost:9000/haproxy_stats` |
| stats auth netcool:object00 | This line enables authentication. It also sets a user name and password. In this example, the user name is `netcool` and the password is `object00`. |

IBM Training IBM

## Logstash log level

Start Logstash with the `--verbose` option to increase the log level.

```
/opt/logstashA/logstash-2.2.1/bin/logstash --verbose -f /opt/logstashA/logstash-
2.2.1/conf/logstashA.conf -l /opt/logstashA/logstash-2.2.1/log/logstashA-debug.log &
```




*Logstash log level*

To increase the log level of a Logstash instance, start it with the `--verbose` or the `--debug` option.

With the `--debug` option, you can see Logstash accepting input, adding tags, using filters, and sending output. In this example, a Logstash sender uses the mutate filter to add and replace fields in a message.

```
{:timestamp=>"2016-07-08T19:24:29.341000+0000", :message=>"config
LogStash::Filters::Mutate/@replace =
{\"host\"=>\"%{[fields][env]}_%{[fields][module]}\"}", :level=>:debug,
:file=>"logstash/config/mixin.rb", :line=>"122", :method=>"config_init"}
{:timestamp=>"2016-07-08T19:25:13.351000+0000",
:message=>"filters/LogStash::Filters::Mutate: adding value to field",
:field=>"path", :value=>["%{[fields][type]}"], :level=>:debug,
:file=>"logstash/util/decorators.rb", :line=>"31", :method=>"add_fields"}
{:timestamp=>"2016-07-08T19:25:13.366000+0000",
:message=>"filters/LogStash::Filters::Mutate: adding value to field",
:field=>"Location", :value=>["%{[fields][site]}"], :level=>:debug,
:file=>"logstash/util/decorators.rb", :line=>"31", :method=>"add_fields"}
{:timestamp=>"2016-07-08T19:25:13.380000+0000",
:message=>"filters/LogStash::Filters::Mutate: adding value to field",
:field=>"Platform", :value=>["%{[fields][platform]}"], :level=>:debug,
:file=>"logstash/util/decorators.rb", :line=>"31", :method=>"add_fields"}
```

IBM Training

## Use the file output plug-in for troubleshooting

```
output {

    file {
        path => "/opt/logstashA/logstash-2.2.1/log/logstashA-debug.log"
        codec => rubydebug
    } #end file

    if ("mutate_filebeat" or in [tags]) and ! ("_grokparsefailure" in [tags]) {
        kafka {
            bootstrap_servers =>"kafka.csite.ibm.edu:17991"
            topic_id => "%{datasource}"
            message_key => "%{resourceID}"
        } #end Kafka output
    } #end Kafka condition

} # end output section
```

Troubleshooting



*Use the file output plug-in for troubleshooting*

Use the file output plug-in to configure Logstash to send log messages it has processed to its log file. You can look at the messages to see the result of Logstash processing, for example:

- Whether filters were successful
- The the value for each field in the log message
- Tag values
- The message after Logstash is finished processing it

In this example, a Logstash sender is sending messages to a Kafka broker and to a file named `/opt/logstashA/logstash-2.2.1/log/logstashA-debug.log`.

**Important:** If you send all messages to a file, every log message in the target log file is also saved on the Logstash server. You might want to use the file output plug-in for testing only and disable file output after you are finished troubleshooting to conserve disk space.

IBM Training

## Use tags to verify filter operations

```
filter {
    if [fields][collector] == "filebeats-collection.csite.ibm.edu" {
        mutate {
          add_field => [ "datasource", "%{[fields][env]}_%{[fields][module]}_%{[fields][type]}" ]
          add_field => [ "resourceID", "%{[beat][hostname]}_%{source}_1" ]
          add_tag => "mutate_filebeat"
        }# end mutate
    } #end filebeat condition

    if [type] == "lfa" {
        grok {
            patterns_dir => "/opt/logstashA/logstash-2.2.1/patterns"
            match => [ "message", "%{LFAMESSAGE}" ]
            add_tag => ["grok_lfa"]
        } #end initial LFA grok
    } #end initial LFA condition

    if "grok_lfa" in [tags] {
        mutate {
            replace => ["message","%{LFA_ORIG_MSG}"]
            add_field => [ "datasource", "%{LFA_INSTANCE}_%{LFA_MODULE}"]
            add_field => [ "resourceID", "%{LFA_HOSTNAME}_%{LFA_LOGNAME}_1"]
        }# end mutate
    }# end grok_lfa condition

} #end filter section
```




*Use tags to verify filter operations*

A good practice is to add a tag to each message after you alter it with a filter. These tags can confirm if a filter was successful.

In this example, you can confirm if the mutate and grok filters were successful by the presence or absence of the `mutate_filebeat` and `grok_lfa` tags.

IBM Training

## Scala output plug-in logging

```
output {

    scala {
        scala_url => "https://log-analysis.csite.ibm.edu:9987/Unity/DataCollector"
        scala_user => "unityadmin"
        scala_password => "object00"
        scala_keystore_path => ""
        batch_size => 500000
        idle_flush_time => 5
        sequential_flush => true
        num_concurrent_writers => 20
        use_structured_api => false
        disk_cache_path => "/opt/logstashW/training/cache/basecache"
        date_format_string => "yyyy-MM-dd'T'HH:mm:ssX"
        log_file => "/opt/logstashW/logstash-2.2.1/log/scala_logstashW.log"
        log_level => "info"
    }#end scala output

}# end output section
```

Troubleshooting 

*Scala output plug-in logging*

The scala output plug-in can write messages to its own log file. This log file shows communication between the Logstash sender and the Log Analysis server.

Use the following options in the scala output plug-in configuration to control the log file:

- **log_file**: The file that is used for logging information from the scala output plug-in.
- **log_level**: The level of logging information for the scala output plug-in. The supported levels are: fatal, error, warn, info, and debug.

IBM Training

## Kafka troubleshooting

- Kafka broker logging is configured in the `<KAFKA_HOME>/config/log4j.properties` file.
- Kafka broker log levels are: TRACE, DEBUG, INFO, WARN, ERROR, and FATAL.
- Use the following log files in the `<KAFKA_HOME>/log/server.log` directory for troubleshooting:
  - `controller.log`
  - `kafka-request.log`
  - `kafkaServer-gc.log`
  - `kafkaServer.out`
  - `log-cleaner.log`
  - **`server.log`**
  - `state-change.log`

```
# Turn on all our debugging info
#log4j.logger.kafka.producer.async.DefaultEventHandler=DEBUG, kafkaAppender
#log4j.logger.kafka.client.ClientUtils=DEBUG, kafkaAppender
#log4j.logger.kafka.perf=DEBUG, kafkaAppender
#log4j.logger.kafka.perf.ProducerPerformance$ProducerThread=DEBUG, kafkaAppender
#log4j.logger.org.I0Itec.zkclient.ZkClient=DEBUG
log4j.logger.kafka=DEBUG, kafkaAppender
```

log4j.properties

Troubleshooting



*Kafka troubleshooting*

The `log4j.properties` file controls logging for Kafka brokers. Use the `log4j.logger.kafka` property to set the log level. You must restart the Kafka broker for any changes to take effect.

IBM Training

## Zookeeper logging

- Zookeeper logging is configured in the `<KAFKA_HOME>/config/log4j.properties` file.
- Zookeeper log levels are: TRACE, DEBUG, INFO, WARN, ERROR, and FATAL.
- Use the `<KAFKA_HOME>/log/server.log`, `zookeeper-gc.log`, and `zookeeper.out` files for troubleshooting.

```
# Turn on all our debugging info
#log4j.logger.kafka.producer.async.DefaultEventHandler=DEBUG, kafkaAppender
#log4j.logger.kafka.client.ClientUtils=DEBUG, kafkaAppender
#log4j.logger.kafka.perf=DEBUG, kafkaAppender
#log4j.logger.kafka.perf.ProducerPerformance$ProducerThread=DEBUG, kafkaAppender
log4j.logger.org.I0Itec.zkclient.ZkClient=DEBUG, kafkaAppender
log4j.logger.kafka=INFO, kafkaAppender
```

log4j.properties




*Zookeeper logging*

The `log4j.properties` file controls logging for the Zookeeper server. Use the `log4j.logger.org.I0Itec.zkclient.ZkClient` property to set the log level. You must restart the Zookeeper server for any changes to take effect.

IBM Training

## Viewing data in Kafka

Use the following command to list all topics:

```
kafka-topics.sh --list --zookeeper kafka.csite.ibm.edu:17981

DEV_IBM-HTTP-Server_access-log
LabDB2Instance_db2diag
labInstance_syslog
```

Use the following command to show all data in a topic:

```
kafka-console-consumer.sh --zookeeper kafka.csite.ibm.edu:17981 --topic
labInstance_syslog --from-beginning
```




*Viewing data in Kafka*

You can use command line tools to confirm that a Kafka cluster is receiving data. These tools are in the `<KAFKA_HOME>/bin` directory.

Use the `kafka-topics.sh` tool to list all topics in the Kafka cluster.

Use the `kafka-console-consumer.sh` tool to view the data in a topic. If you use the `--from-beginning` option, the tool shows all data in a topic. If you do not use the `--from-beginning` option, the tool shows only data that is currently being written to the topic.

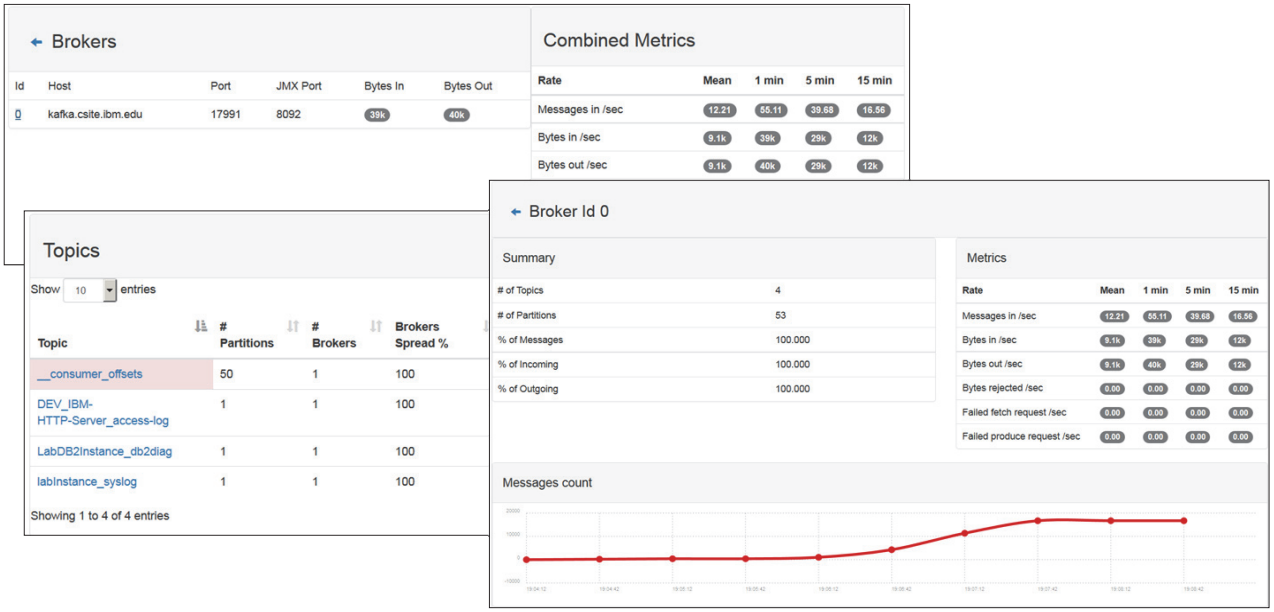
You can also use the `kafka-topics.sh` tool to show details about a topic. In the following example, the tool shows information about the topic named **DEV_IBM-HTTP-Server_access-log**:

```
./kafka-topics.sh --describe --zookeeper kafka.csite.ibm.edu:17981 --topic
DEV_IBM-HTTP-Server_access-log

Topic:DEV_IBM-HTTP-Server_access-logPartitionCount:3ReplicationFactor:1Configs:
   Topic: DEV_IBM-HTTP-Server_access-logPartition: 0Leader: 0Replicas: 0Isr: 0
```

IBM Training

## Kafka Manager




Troubleshooting

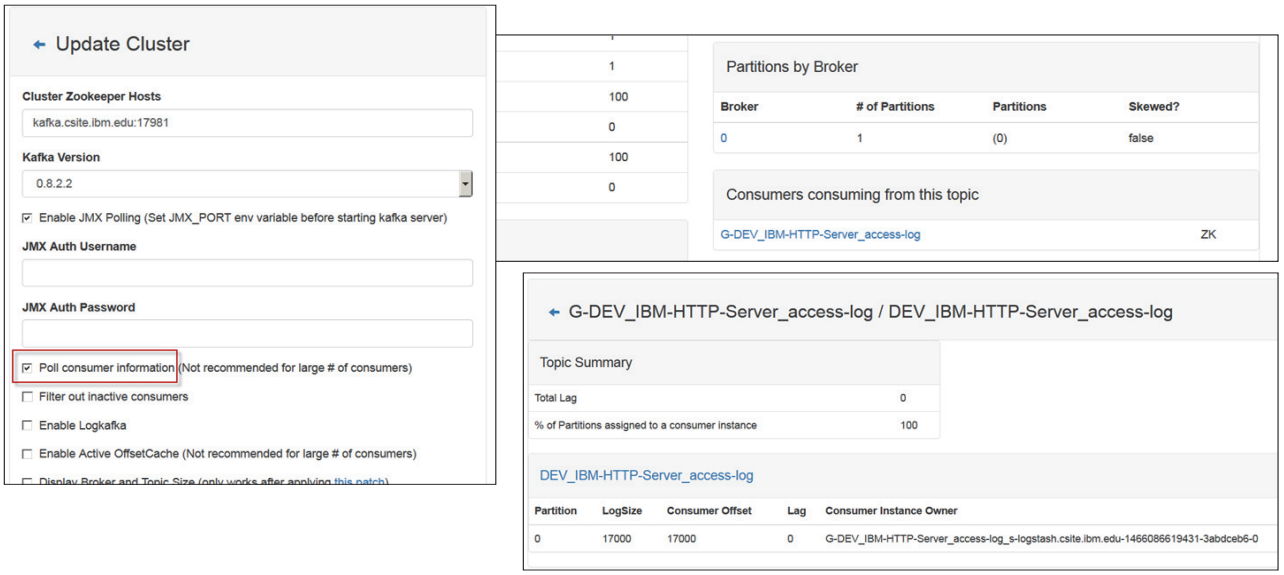


*Kafka Manager*

Kafka Manager is a web-based user interface that shows the configuration of your cluster. You can also use Kafka Manager to view data and modify your cluster. Although Kafka Manager is not required, it can be useful for troubleshooting and maintenance.

IBM Training

## Using Kafka Manager to verify sender Logstash data flow




Troubleshooting © Copyright IBM Corporation 2016

*Using Kafka Manager to verify sender Logstash data flow*

If you enable the **Poll consumer information** option in Kafka Manager, you can see which Logstash senders are pulling data from Kafka. This information is useful for troubleshooting the data flow between Kafka and the Logstash senders.

IBM Training

## Log Analysis troubleshooting

- The `GenericReceiver.log` file shows data coming in to the generic receiver interface.
- To increase the log level logs, edit the `<LA_HOME>/wlp/usr/servers/Unity/apps/Unity.war/WEB-INF/classes/log4j.properties` file.
- Log levels are: TRACE, DEBUG, INFO, WARN, ERROR, and FATAL

log4j.properties

```
# Unity Generic Receiver Log configurations
log4j.logger.UnityGenericReceiver=DEBUG,UNITY_GR_FILE
#define the appender named FILE
log4j.appender.UNITY_GR_FILE=org.apache.log4j.RollingFileAppender
log4j.appender.UNITY_GR_FILE.MaxFileSize=50000KB
```




*Log Analysis troubleshooting*

The generic receiver is a REST interface for loading data into IBM Operations Analytics Log Analysis.

The `log4j.properties` file controls logging for the Log Analysis generic receiver. Use the `log4j.logger.UnityGenericReceiver` property to set the log level. You must restart Log Analysis for any changes to take effect.

IBM Training

## Generic receiver accepting data

Messages like these indicate that the generic receiver is successfully processing data.

```
06/17/16 19:07:42:146 UTC [Default Executor-thread-17915] INFO  - UnityFlowController :
Batch Status for -> Lab_DB2 , Size: 273 , Num successful: 273 , Num failures: 0 ,
Indexed Source volume: 136484

06/17/16 19:07:42:149 UTC [Default Executor-thread-17915] INFO  -
DataCollectorRestServlet : Batch of Size 273 processed and encountered 0 failures

06/17/16 19:07:42:223 UTC [Thread-93] INFO  - IndexStatusChecker : Updating statistics
for data source [Lab_DB2], stream [_unity_default_stream], ingested bytes [137097],
write date [Fri Jun 17 19:07:42 UTC 2016].
```

Troubleshooting 

*Generic receiver accepting data*

The `GenericReceiver.log` file shows any attempt to load data into Log Analysis. This log file is useful for troubleshooting communication between a Logstash sender and the Log Analysis server.

This set of messages confirms that a Logstash sender is successfully sending messages to the Log Analysis server and that the data in the messages is successfully indexed.

IBM Training

## Unit summary

- Troubleshoot Filebeat
- Troubleshoot the Log File Agent
- Troubleshoot HAProxy
- Troubleshoot Logstash
- Troubleshoot Kafka
- Troubleshoot Log Analysis




*Unit summary*

IBM Training