Using Liberty with Elastic Stack

Subtitle of presentation in this location as long as needed

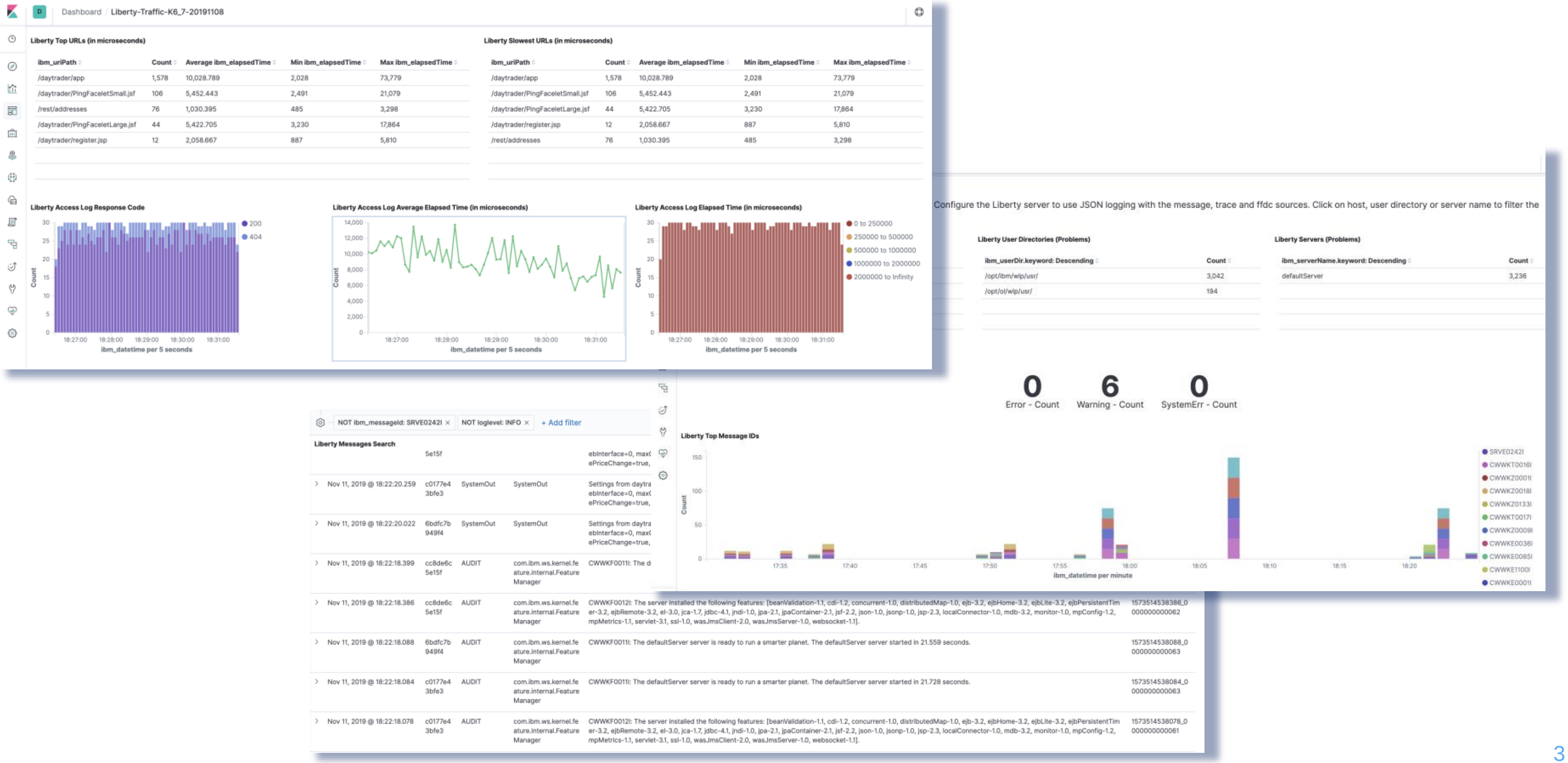
Don Bourne, WebSphere Observability Architect

Agenda

- Logging
- Elastic Stack
- Liberty Events
- JSON Logging
- LogstashCollector
- Logging in Red Hat OpenShift Container Platform

Logging

Liberty Dashboards



Liberty Top URLs (in microseconds)

ibm_uriPath	Count	Average ibm_elapsedTime	Min ibm_elapsedTime	Max ibm_elapsedTime
/daytrader/app	1,578	10,028.789	2,028	73,779
/daytrader/PingFaceletSmall.jsf	106	5,452.443	2,491	21,079
/rest/addresses	76	1,030.395	485	3,298
/daytrader/PingFaceletLarge.jsf	44	5,422.705	3,230	17,864
/daytrader/register.jsp	12	2,058.667	887	5,810

Liberty Slowest URLs (in microseconds)

ibm_uriPath	Count	Average ibm_elapsedTime	Min ibm_elapsedTime	Max ibm_elapsedTime
/daytrader/app	1,578	10,028.789	2,028	73,779
/daytrader/PingFaceletSmall.jsf	106	5,452.443	2,491	21,079
/daytrader/PingFaceletLarge.jsf	44	5,422.705	3,230	17,864
/daytrader/register.jsp	12	2,058.667	887	5,810
/rest/addresses	76	1,030.395	485	3,298

Configure the Liberty server to use JSON logging with the message, trace and ffdc sources. Click on host, user directory or server name to filter the

Liberty User Directories (Problems)

ibm_userDir.keyword: Descending	Count
/opt/ibm/wlp/usr/	3,042
/opt/ol/wlp/usr/	194

Liberty Servers (Problems)

ibm_serverName.keyword: Descending	Count
defaultServer	3,236

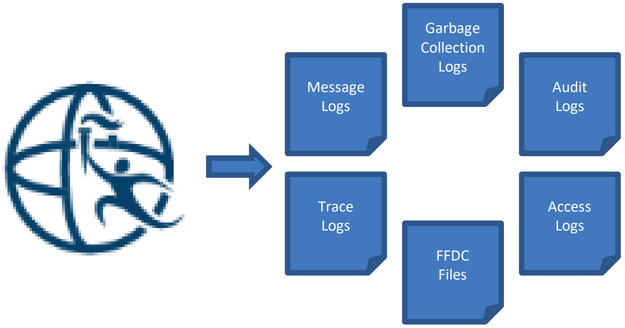
0	6	0
Error - Count	Warning - Count	SystemErr - Count

Liberty Messages Search

	5e15f			ebInterface=0, max(ePriceChange=true,	
Nov 11, 2019 @ 18:22:20.259	c0177e4 3bfe3	SystemOut	SystemOut	Settings from daytra ebInterface=0, max(ePriceChange=true,	
Nov 11, 2019 @ 18:22:20.022	6bdfc7b 949f4	SystemOut	SystemOut	Settings from daytra ebInterface=0, max(ePriceChange=true,	
Nov 11, 2019 @ 18:22:18.399	cc8de6c 5e15f	AUDIT	com.ibm.ws.kernel.feature.internal.FeatureManager	CWWKF0011I: The d	
Nov 11, 2019 @ 18:22:18.386	cc8de6c 5e15f	AUDIT	com.ibm.ws.kernel.feature.internal.FeatureManager	CWWKF0012I: The server installed the following features: [beanValidation-1.1, cdi-1.2, concurrent-1.0, distributedMap-1.0, ejb-3.2, ejbHome-3.2, ejbLite-3.2, ejbPersistentTimer-3.2, ejbRemote-3.2, el-3.0, jca-1.7, jdbc-4.1, jndi-1.0, jpa-2.1, jpaContainer-2.1, jsf-2.2, json-1.0, jsonp-1.0, jsp-2.3, localConnector-1.0, mdb-3.2, monitor-1.0, mpConfig-1.2, mpMetrics-1.1, servlet-3.1, ssl-1.0, wasJmsClient-2.0, wasJmsServer-1.0, websocket-1.1].	1573514538386_0 000000000062
Nov 11, 2019 @ 18:22:18.088	6bdfc7b 949f4	AUDIT	com.ibm.ws.kernel.feature.internal.FeatureManager	CWWKF0011I: The defaultServer server is ready to run a smarter planet. The defaultServer server started in 21.559 seconds.	1573514538088_0 000000000063
Nov 11, 2019 @ 18:22:18.084	c0177e4 3bfe3	AUDIT	com.ibm.ws.kernel.feature.internal.FeatureManager	CWWKF0011I: The defaultServer server is ready to run a smarter planet. The defaultServer server started in 21.728 seconds.	1573514538084_0 000000000063
Nov 11, 2019 @ 18:22:18.078	c0177e4 3bfe3	AUDIT	com.ibm.ws.kernel.feature.internal.FeatureManager	CWWKF0012I: The server installed the following features: [beanValidation-1.1, cdi-1.2, concurrent-1.0, distributedMap-1.0, ejb-3.2, ejbHome-3.2, ejbLite-3.2, ejbPersistentTimer-3.2, ejbRemote-3.2, el-3.0, jca-1.7, jdbc-4.1, jndi-1.0, jpa-2.1, jpaContainer-2.1, jsf-2.2, json-1.0, jsonp-1.0, jsp-2.3, localConnector-1.0, mdb-3.2, monitor-1.0, mpConfig-1.2, mpMetrics-1.1, servlet-3.1, ssl-1.0, wasJmsClient-2.0, wasJmsServer-1.0, websocket-1.1].	1573514538078_0 000000000061

Logging

Liberty produces a variety of logs as it runs. Each log file may store hundreds or thousands of entries each day.



Message Logs

Garbage Collection Logs

Audit Logs

Trace Logs

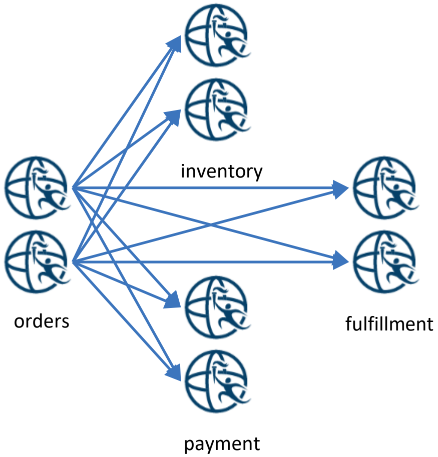
FFDC Files

Access Logs

There are multiple kinds of events worth gathering

Logging

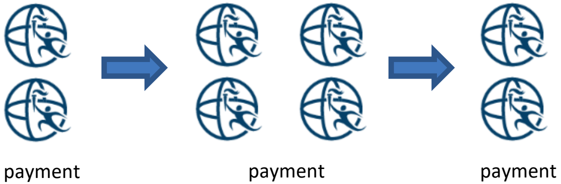
Lots of Liberty servers are typically used together to provide the various parts of an application – particularly apps that use microservice architectures. Each server has its own set of logs.



It is useful to be able to see logs from multiple servers together

Logging

Throughout the day servers can be added or removed, to meet demands of traffic. Ops teams may have hundreds of logs to inspect to identify the scope and duration of problems.



Servers and their logs are ephemeral – logs need to be centralized

Elastic Stack

Elastic Stack?

There is a good chance you already have an enterprise logging solution.

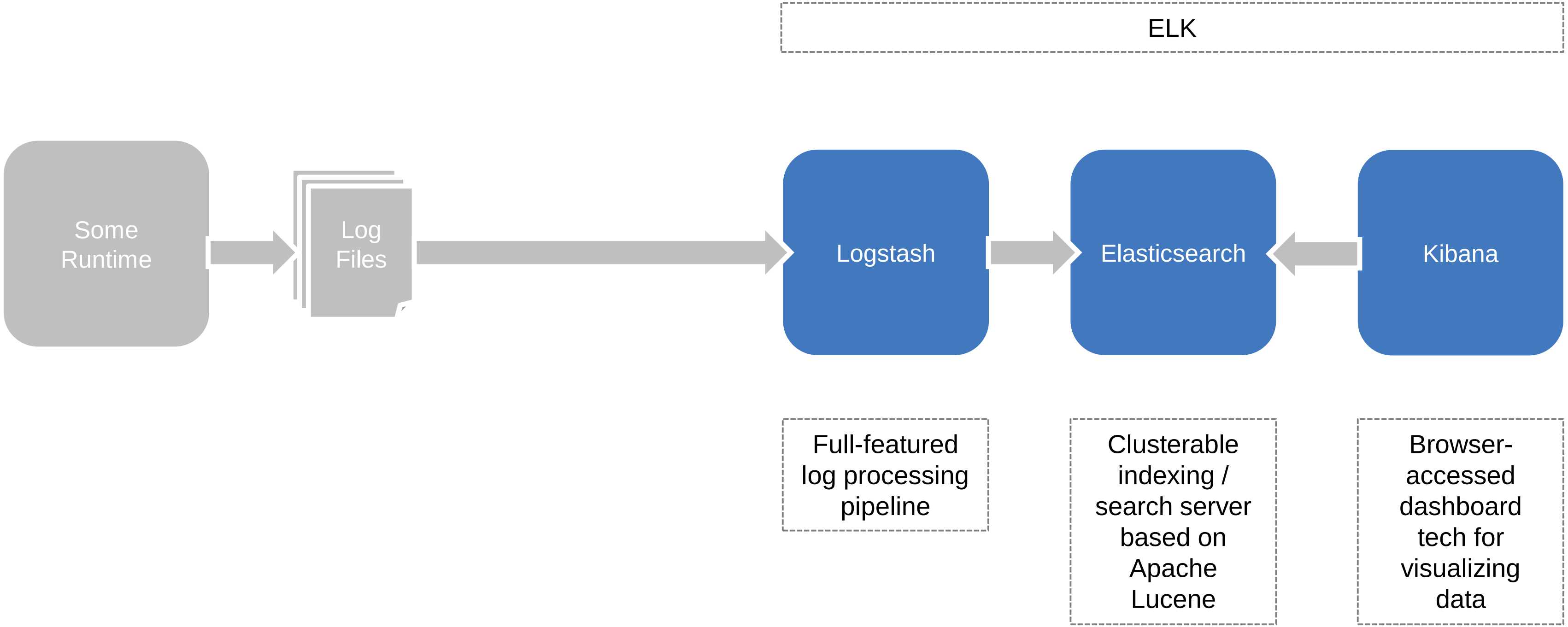
That logging solution may or may not be based on the Elastic Stack.

Elastic Stack is one of many popular solutions for log aggregation and analysis.

If you are excited about collecting and analyzing your Liberty logs (and other events) with the Elastic Stack, this presentation is for you.

In no particular order you might also be interested in Splunk, Graylog, IBM Operations Analytics, Loggly, Datadog, LogDNA, Papertrail, … let me know if you'd like to know more about using Liberty with other logging solutions!

Elasticsearch Logstash Kibana (ELK)



ELK

Some Runtime

Log Files

Logstash

Elasticsearch

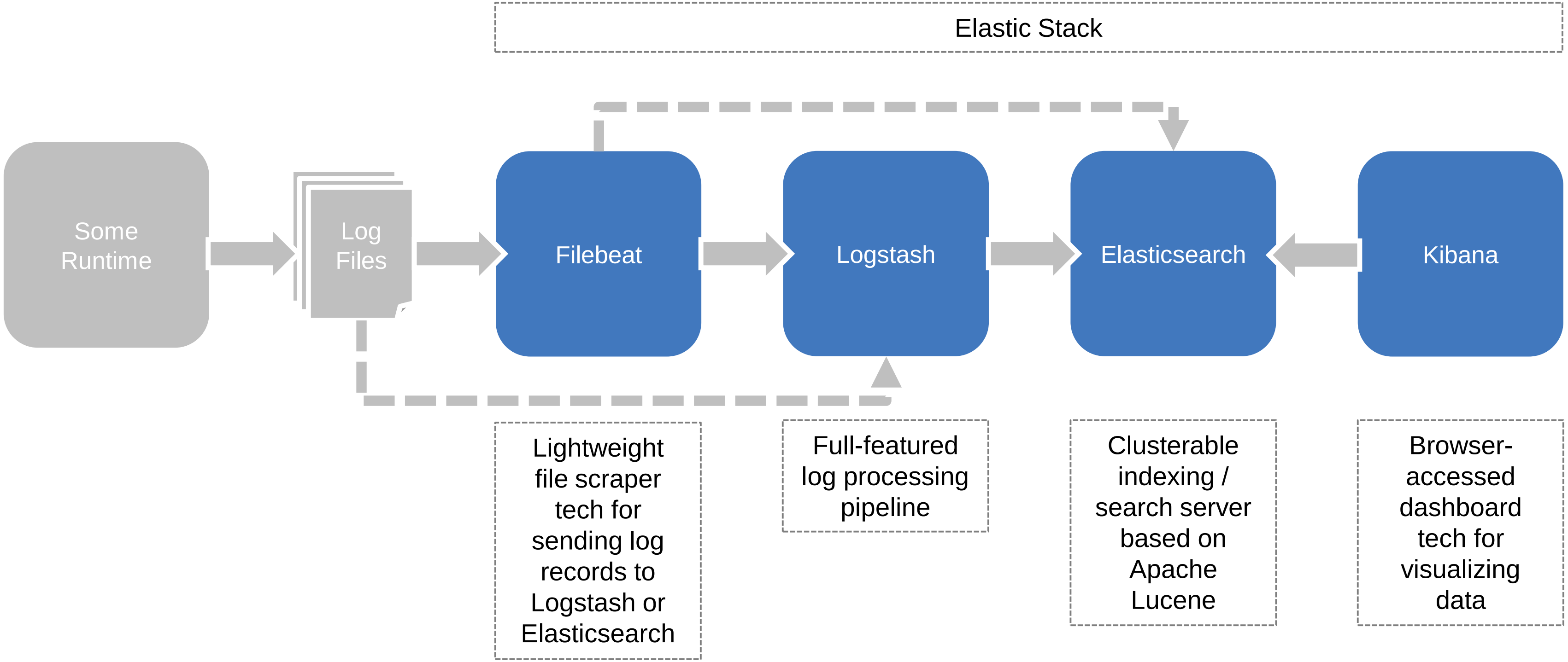
Kibana

Full-featured log processing pipeline

Clusterable indexing / search server based on Apache Lucene

Browser-accessed dashboard tech for visualizing data

Elastic Stack



Elastic Stack

Some Runtime

Log Files

Filebeat

Logstash

Elasticsearch

Kibana

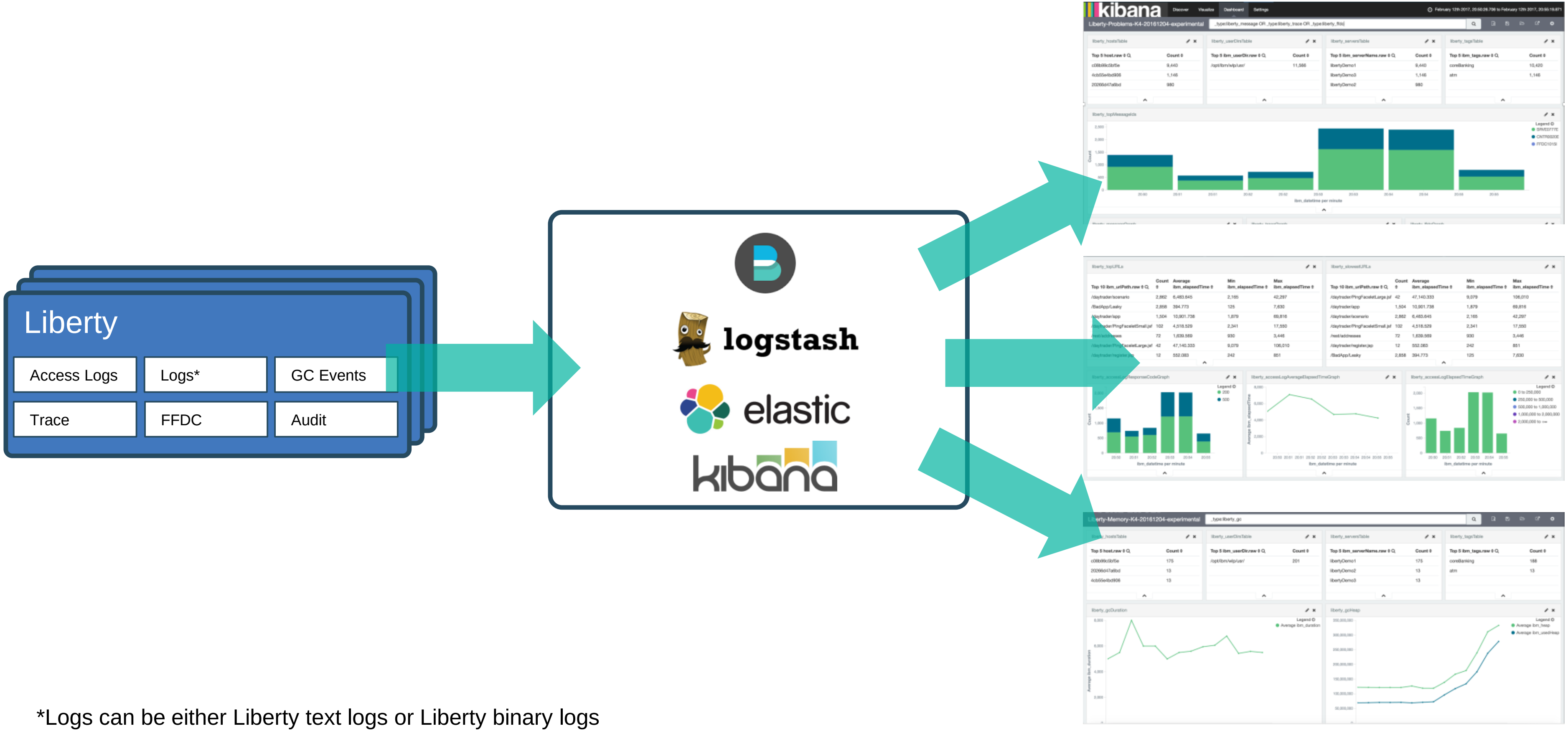
Lightweight file scraper tech for sending log records to Logstash or Elasticsearch

Full-featured log processing pipeline

Clusterable indexing / search server based on Apache Lucene

Browser-accessed dashboard tech for visualizing data

Log Analysis with Elastic Stack



Liberty

Access Logs

Logs*

GC Events

Trace

FFDC

Audit

*Logs can be either Liberty text logs or Liberty binary logs

Liberty Events

Liberty Events

Liberty has 2 main ways of emitting log/event data for use in log aggregation solutions:

- JSON Logging
- logstashCollector-1.0 feature

Liberty (currently) emits 6 different event types.

Event Type	JSON Logging	logstashCollector
Logs	Yes	Yes
Trace	Yes	Yes
FFDC	Yes	Yes
Access Logs	Yes	Yes
Audit Logs	Yes	Yes
Garbage Collection	No	Yes (IBM JDKs only)

Liberty Events

Each event type has a set of fields. Fields below in bold are common to all event types.

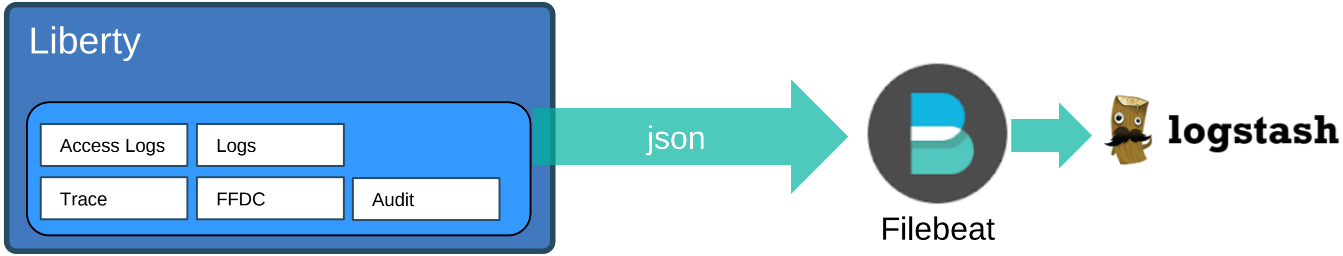
Log Event Fields	Sample Value
type	liberty_message
ibm_datetime	2019-11-10T19:16:32.531+0000
ibm_sequence	1573413392531_0000000000022
host	c2011270034a
ibm_userDir	/opt/ol/wlp/usr/
ibm_serverName	defaultServer
ibm_threadId	00000030
module	com.ibm.ws.session.WASSessionCore
ibm_className	SessionContextRegistryImpl
ibm_methodName	getSessionContext
loglevel	INFO
ibm_messageId	SESN0176I
message	SESN0176I: A new session context will be created for application key default_host/tradelite

JSON Logging

JSON Logging

Use Liberty's **JSON Logging** when…

- ☐ You are running Liberty in an environment that manages your logs for you (eg. Kubernetes)
- ☐ You don't want to hard code details of where your logs should go inside your image [1]
- ☐ You want flexibility to have any log analysis solution that understands JSON be able to consume your logs



Filebeat, Logstash configuration and Kibana dashboards for Elastic Stack 5/6/7 provided at [2]

[1] https://12factor.net/logs
[2] https://github.com/WASdev/sample.dashboards

JSON Logging

What do logs look like in JSON format?

```
{
  "type": "liberty_message",
  "host": "97d853b79f8e",
  "ibm_userDir": "/opt/ibm/wlp/usr/",
  "ibm_serverName": "defaultServer",
  "ibm_datetime": "2018-07-19T12:20:23.892+0000",
  "ibm_messageId": "CWWKE0001I",
  "ibm_threadId": "00000001",
  "module": "com.ibm.ws.kernel.launch.internal.FrameworkManager",
  "loglevel": "AUDIT",
  "ibm_sequence": "1532002823892_0000000000001",
  "message": "CWWKE0001I: The server defaultServer has been launched."
}
```

(new lines and spaces added for clarity – each JSON record is output on one line)

JSON Logging

Typical Liberty Configuration

- direct messages, trace, FFDC, access log and audit events to stdout in JSON format
- write nothing to messages.log
- write nothing to trace.log

```
com.ibm.ws.logging.console.source=message,trace,ffdc,accessLog,audit
com.ibm.ws.logging.console.format=json
com.ibm.ws.logging.console.log.level=info

com.ibm.ws.logging.message.source=
com.ibm.ws.logging.message.format=json

com.ibm.ws.logging.trace.file.name=stdout
```

wlp/usr/servers/*serverName*/bootstrap.properties

JSON Logging

Alternately, if you want to set configuration with environment variables

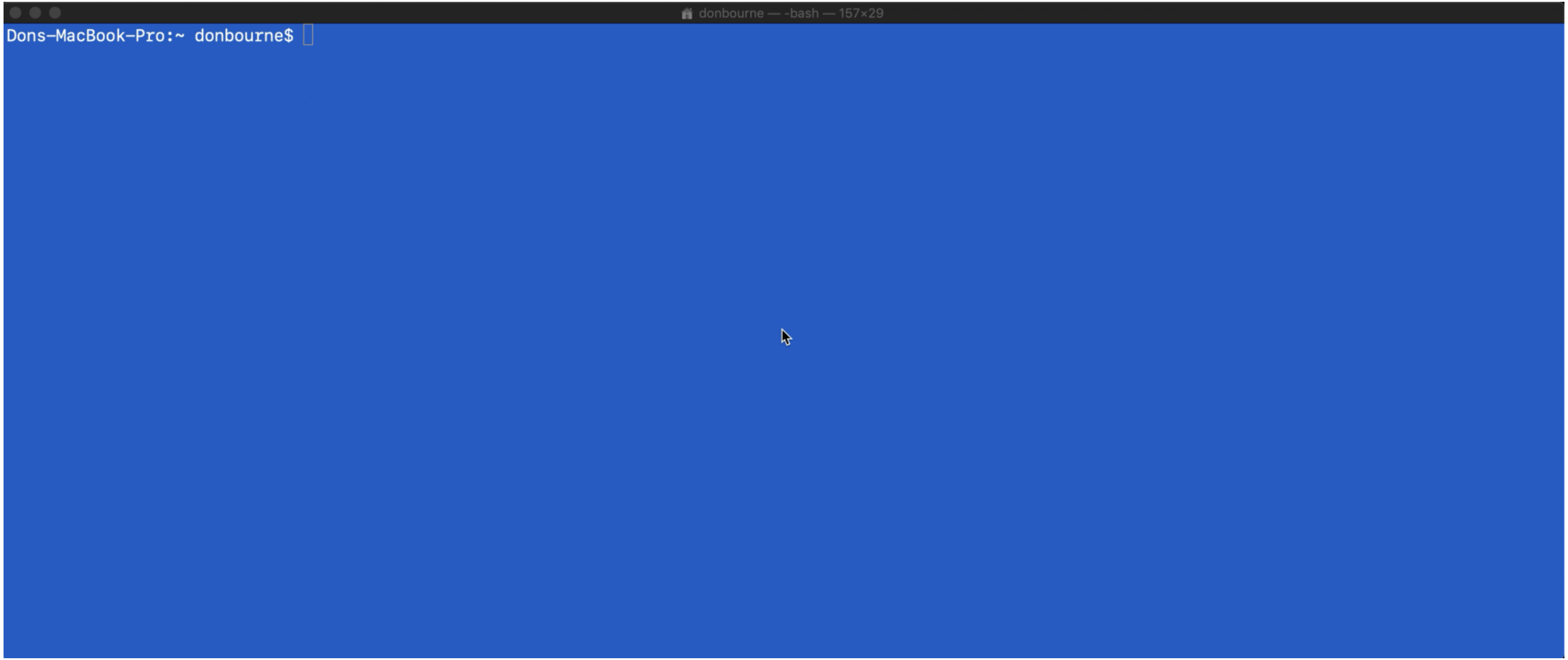
- Using environment variables set at deployment time avoids need to put logging configuration in your Docker image

```
docker run \
 -e "WLP_LOGGING_CONSOLE_SOURCE=message,trace,accessLog,ffdc,audit" \
 -e "WLP_LOGGING_CONSOLE_FORMAT=json" \
 -e "WLP_LOGGING_CONSOLE_LOGLEVEL=info" \
 -e "WLP_LOGGING_MESSAGE_FORMAT=json" \
 -e "WLP_LOGGING_MESSAGE_SOURCE=" \
 open-liberty
```

- If desired, you can even change the JSON field names to suit your needs

```
WLP_LOGGING_JSON_FIELD_MAPPINGS=loglevel:level,message:log
```

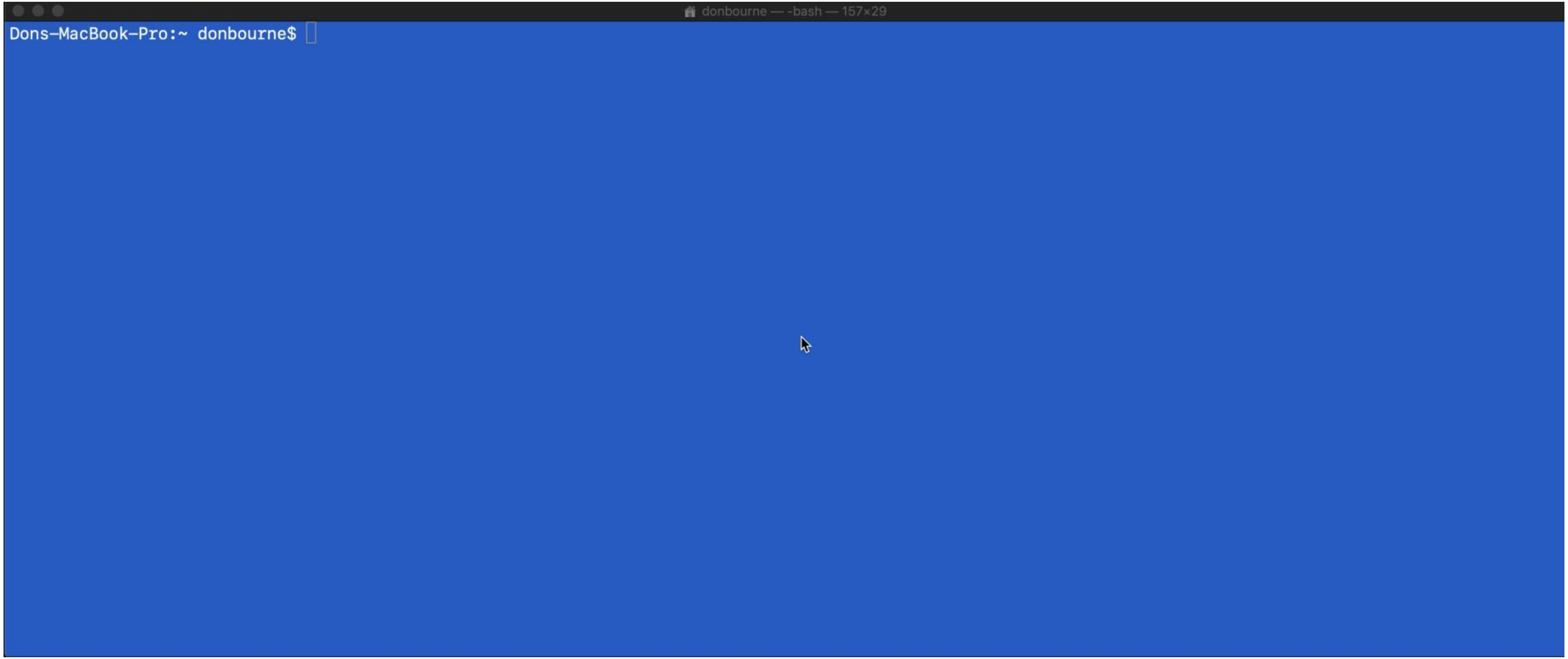
JSON Logging



```
Dons-MacBook-Pro:~ donbourne$ 
```

Running open-liberty docker image with default log settings

JSON Logging



```
Dons-MacBook-Pro:~ donbourne$ 
```

Running open-liberty docker image with environment variables to enable JSON logging

JSON Logging

What if a human needs to read it?

Use jq [1], or similar tools, to format output

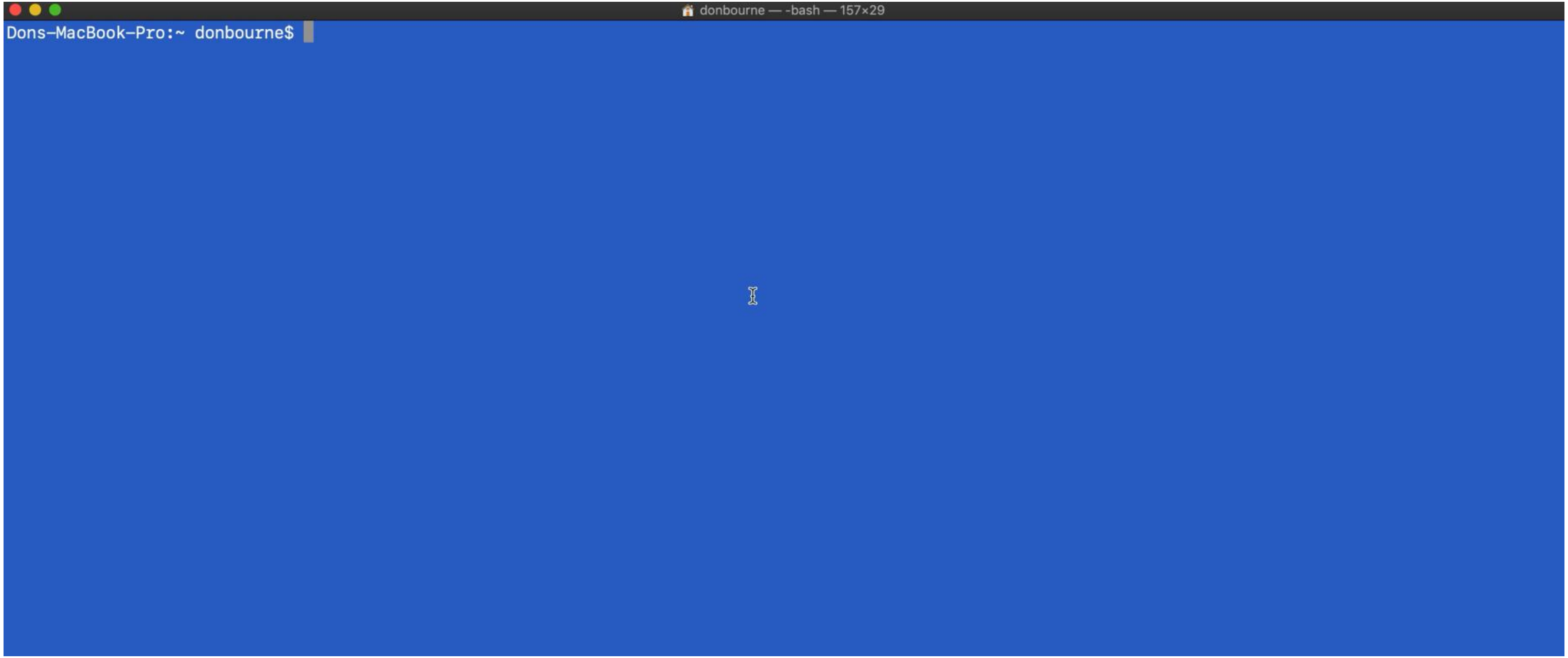
Create an alias for your jq command when you find a format you like!

```
# example 1 - just include the message
alias prettylog="jq '.message' -r"

# example 2 - include the datetime, log level, message
alias prettylog="jq '.ibm_datetime + \"  \" + .loglevel + \"\\t\" + \" \" + .message' -r"

docker logs <liberty containerId> | prettylog
```

---

[1] https://stedolan.github.io/jq/download/ (this is not an IBM tool or website)

JSON Logging



```
Dons-MacBook-Pro:~ donbourne$
```

Using jq command to make JSON logs more human friendly

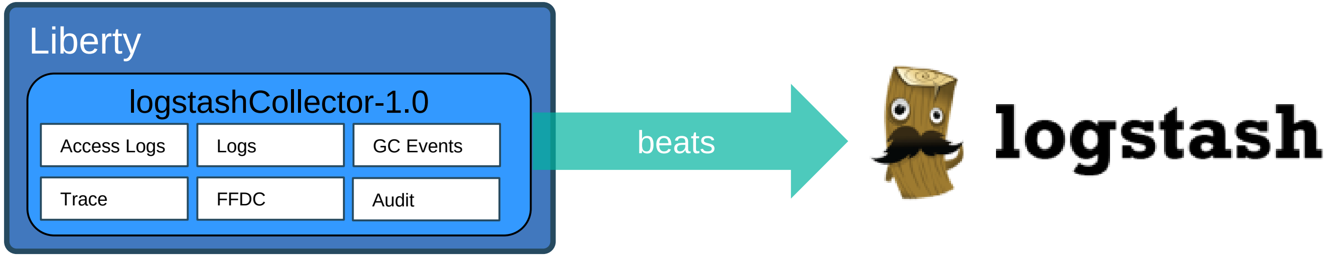
Logstash Collector

Logstash Collector

The logstashCollector-1.0 feature sends events directly from Liberty to your remote or local Logstash endpoint.

Use Liberty's **logstashCollector** when…

- ☐ You want to send logs to your Elastic Stack but can't install an agent to forward your logs (for example, running Liberty in a public Cloud Foundry)



Logstash configuration and Kibana dashboards for Elastic Stack 5/6/7 provided [1]

[1] https://github.com/WASdev/sample.logstash.collector/tree/master/logstashCollector-1.0

Logstash Collector

Steps to get set up (detailed instructions at [1])

1. Add Logstash certificate to Liberty truststore
2. Add configuration to server.xml (next page)
3. Use provided Logstash configuration
4. Start the Liberty server and generate some events
5. Create Kibana index pattern
6. Import provided Kibana dashboards

[1] https://www.ibm.com/support/knowledgecenter/SSEQTP_liberty/com.ibm.websphere.wlp.doc/ae/twlp_analytics_logstash.html

Logstash Collector Configuration

Note: Once created, configuration can be shared with other servers or copied

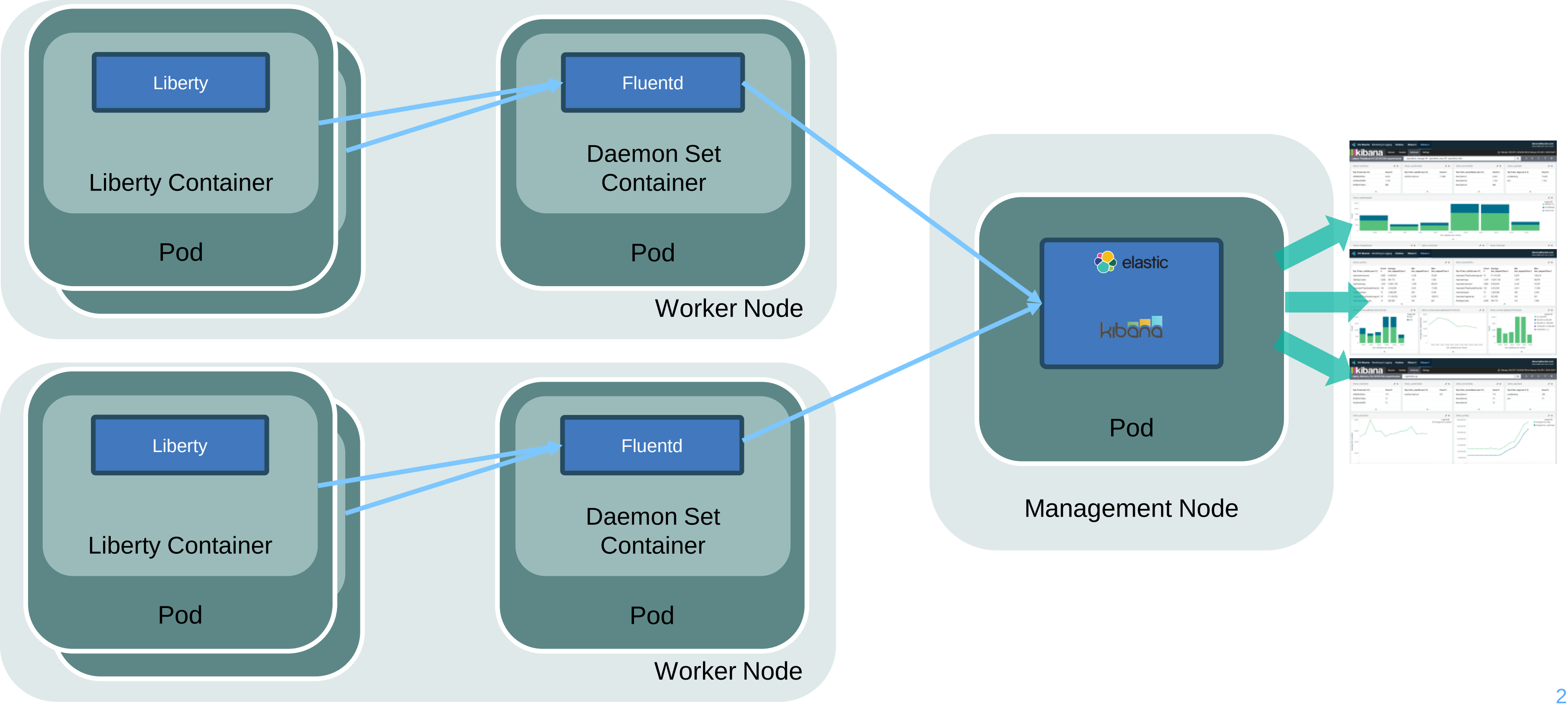
```
<featureManager>
    <feature>logstashCollector-1.0</feature>
</featureManager>

<keyStore id="defaultKeyStore" password="Liberty" />
<ssl id="mySSLConfig" trustStoreRef="defaultKeyStore" keyStoreRef="defaultKeyStore" />

<logstashCollector
    source="message, trace, ffdc, garbageCollection, accessLogs, audit"
    host="myhost.acme.com"
    port="9091"
    sslRef="mySSLConfig"
    maxFieldLength="5000”             <<< Adjust, if needed, to avoid message truncation
    <tag>toronto</tag>                <<< tags are included in all events from this server
    <tag>coreBanking</tag>
/>
```

*Configure logs, trace, access logs, audit separately as usual

Red Hat OpenShift Container Platform

RHOCP Elastic Stack

Liberty running in Red Hat OpenShift Container Platform



Liberty

Liberty Container

Pod

Fluentd

Daemon Set Container

Pod

Worker Node

Liberty

Liberty Container

Pod

Fluentd

Daemon Set Container

Pod

Worker Node

elastic

kibana

Pod

Management Node

Red Hat OpenShift Container Platform – Logging

RHOCP / OKD provides ability to deploy two logging stacks, each consisting of Elasticsearch, Fluentd and Kibana (EFK)

- “ops” stack, for logs from Kubernetes and OpenShift components (not intended for app logs)
- another stack for app logs

Red Hat OpenShift Container Platform – Logging

One-time set up of EFK stack for application data (1/2)

Set configuration parameters in the inventory file[1] to indicate you want to install the second logging stack for application logs:

```
openshift_logging_use_ops=True

openshift_logging_es_ops_nodeselector={"node-role.kubernetes.io/infra":"true"}
openshift_logging_es_nodeselector={"node-role.kubernetes.io/infra":"true"}

openshift_logging_es_ops_memory_limit=5G
openshift_logging_es_memory_limit=3G
```

Run the ansible-playback command to install the logging stack:

```
ansible-playbook -i <inventory_file> openshift-ansible/playbooks/openshift-logging/config.yml -e
openshift_logging_install_logging=true
```

[1] https://docs.okd.io/3.11/install/configuring_inventory_file.html

Red Hat OpenShift Container Platform – Logging

One-time set up of EFK stack for application data (2/2)

Once deployed you can see all pods related to logging within the openshift-logging namespace:

```
[root@rhel7-okd ~]# oc get pods -n openshift-logging

NAME                                        READY   STATUS      RESTARTS   AGE
logging-curator-1565163000-9fvpf            0/1     Completed   0          20h
logging-curator-ops-1565163000-5l5tx        0/1     Completed   0          20h
logging-es-data-master-iay9qoim-4-cbtjg     2/2     Running     0          3d
logging-es-ops-data-master-hsmsi5l8-3-vlrgs 2/2     Running     0          3d
logging-fluentd-vssj2                       1/1     Running     1          3d
logging-kibana-2-tplkv                      2/2     Running     6          4d
logging-kibana-ops-1-bgl8k                  2/2     Running     2          3d
```

Notice that you will have curator, elasticsearch, and kibana pods for each ”stack” (one with “ops” suffix, one without)

Fluentd automatically routes logs to appropriate logging stack based on which project logs come from

- Logs from default, openshift, openshift-infra projects go to the ops stack
- Logs from other projects go to the other stack

Red Hat OpenShift Container Platform – Logging

One-time set up of Kibana for use with Liberty

1. Find the URL for external access to Kibana and ops Kibana web consoles. Access Kibana at the host/port indicated for your system.

```
[root@rhel7-okd ~]# oc get routes -n openshift-logging
NAME                HOST/PORT                           PATH  SERVICES            PORT    TERMINATION         WILDCARD
logging-kibana      kibana.apps.9.37.135.153.nip.io           logging-kibana      <all>   reencrypt/Redirect  None
logging-kibana-ops  kibana-ops.apps.9.37.135.153.nip.io       logging-kibana-ops  <all>   reencrypt/Redirect  None
```

2. Log in using your OKD / RHOCP username and password.
3. Click Management > Index Pattern. Find the `project.\*` index. Click the refresh fields button.
4. Download the Liberty Kibana dashboards from [1]
5. Click Management > Saved Objects > Import. Drag / Drop the Liberty dashboard files you want to import.

[1] https://github.com/OpenLiberty/open-liberty-operator/tree/master/deploy/dashboards/logging

Red Hat OpenShift Container Platform – Logging

Tips for Liberty logging when running on RHOCP:

- Configure Liberty to use JSON logging with output going to console (for example by setting properties in bootstrap.properties or by setting environment variables)

IBM Cloud Pak for Applications:

- Application Navigator provides action menus – one click to get to your deployment's Problems dashboard directly from your deployment in the Application Navigator UI

Bonus Points

Adding Access Logs

Add access logs by configuring your HTTP endpoint in your server.xml file as follows:

```
<server>

    <!-- access log -->
    <httpEndpoint id="defaultHttpEndpoint" host="*" httpPort="9080" httpsPort="9443">
        <accessLogging
            filepath="${server.output.dir}/logs/http_defaultEndpoint_access.log"
            logFormat='%h %u %t "%r" %s %b %D %{User-agent}i'>
        </accessLogging>
    </httpEndpoint>

</server>
```

JSON logging and logstashCollector-1.0 feature do not themselves enable access logs. Enable access logs (as shown above) and add the “accessLog” source to the list of sources for JSON logging or logstashCollector-1.0.

Adding Audit Logs

Add audit logs by adding the audit-1.0 feature to your server.xml file as follows. In this example, only authentication and authorization events are enabled.

```
<featureManager>
    <feature>audit-1.0</feature>
</featureManager>

<auditFileHandler compact="false" eventsRef="authn,authz"/>

<auditEvent id="authn" eventName="SECURITY_AUTHN" />
<auditEvent id="authz" eventName="SECURITY_AUTHZ" />
```

JSON logging and logstashCollector-1.0 feature do not themselves enable audit events. Enable audit logs (as shown above) and add the “audit” source to the list of sources for JSON logging or logstashCollector-1.0.

Adding Custom Fields to Logs and Trace

LogRecordContext API (think MDC)

Applications that use the LogRecordContext API will have the name/value pairs they have added to the JSON mapping for logs and trace emitted on the same thread.

```
STRING/STRING pairs
// included in json at root level as "ext_someName":"someValue”
LogRecordContext.addExtension("someName","someValue");

STRING/INTEGER pairs
// included in json at root level as "ext_someName_int":someValue (or entirely omitted if someValue isn't parseable as an int)
LogRecordContext.addExtension("someName_int","someValue");

STRING/FLOAT pairs
// included in json at root level as "ext_someName_float":someValue (or entirely omitted if someValue isn't parseable as a float)
LogRecordContext.addExtension("someName_float","someValue");

STRING/BOOLEAN pairs
// included in json at root level as "ext_someName_bool":someValue (or entirely omitted if someValue isn't parseable as a bool)
LogRecordContext.addExtension("someName_bool","someValue");
```

Example

```
LogRecordContext.addExtension("userName","don");
LogRecordContext.addExtension("isCool_bool":"true”);
Logger.info("some message");
```

```
{"ibm_datetime":"2018-02-04T18:56:30.318-
0500","type":"liberty_message","host":"192.168.2.15","ibm_userDir":"
\/wlp\/usr\/","ibm_serverName":"server1","ibm_sequence":"15177885903
18_000000003A4A7","loglevel":"INFO","module":"com.ibm.somepackage.So
meClass","ibm_threadId":"00002db5","message":"some message",
"ext_userName":"don", "ext_isCool_bool":true}
```

Add RequestTiming-1.0 or EventLogging-1.0

Add another dimension to your logs by using requestTiming-1.0 or eventLogging-1.0 Liberty features [1].

requestTiming-1.0

- Prints a report to your logs whenever a request is detected to be slow
- Recommended for use in all production deployments of Liberty
- ext_requestID
  - Unique request identifier field added to JSON for **all** logs/trace for duration of the request

eventLogging-1.0

- Prints a message to your logs at the start and/or end of each event
- ext_requestID
  - Unique request identifier field added to JSON for **all** logs/trace for duration of the request
- ext_contextInfo
  - Provides details of servlet and JDBC calls
  - Included in log entries logged from eventLogging feature itself
- ext_eventType
  - Indicates the kind of event
  - Included in log entries logged from eventLogging feature itself

[1] https://developer.ibm.com/wasdev/docs/request-timing-diagnosing-slow-requests-liberty/

WebSphere Customer Advisory Board – open invitation

http://ibm.biz/WebSphereAdvisoryBoard
email: claudiab@us.ibm.com

Sign up now

Join 100 other companies

Be part of customer round tables and deep dive meetings

Influence deliverables

Choose your engagement level:

1. **Stay ahead of the curve**: more time commitment
2. **Close the gap**: quarterly involvement
3. **At your own pace**: impact longer term goals

Get involved.
Be successful.

Summary

- Use a log aggregation and analysis solution to get good use out of the mountain of logs your servers generate
- The Elastic Stack is a popular third party logging solution
- Liberty has an event framework that includes logs, trace, FFDC, access logs, audit logs, and garbage collection events
- Liberty’s JSON logging uses the event framework to output events in a convenient JSON format
- Liberty’s logstashCollector-1.0 feature uses the event framework to send events to Logstash
- Red Hat OpenShift Container Platform (RHOCP) integrates with EFK
- Liberty’s JSON log stream works great with RHOCP
- Developers can add extra fields to logs and trace using the LogRecordContext API
- Liberty’s requestTiming-1.0 and eventLogging-1.0 features add context to logs / trace (and are handy features)
- The Liberty team provides sample dashboards and config files to make it easy to get started using Elastic Stack

Thank You