

How to integrate the Elasticsearch Logstash Kibana (ELK) log analytics stack into IBM Bluemix

Nick Cawood

IBM Cloud Client Adoption and Technical Enablement
Client and Technical Engagement
August 2016

CONTENTS

Introduction.....	3
What is Elasticsearch Logstash Kibana?	3
LOGSTASH	3
ELASTICSEARCH.....	4
KIBANA	4
ELK SYSTEM ARCHITECTURE.....	4
Service Management architecture	6
Log Monitoring System architecture	6
Bluemix environment system architecture.....	6
Creating a virtual server in Bluemix.....	7
CREATE AN SSH KEY	8
PROVISIONING A VIRTUAL SERVER.....	9
SET UP THE VIRTUAL SERVER.....	12
 Install the ELK stack	13
INSTALL ELASTICSEARCH	14
INSTALL KIBANA.....	15
BROADCAST BLUEMIX APP LOG RECORDS	19
INSTALL LOGSTASH.....	19
 Use ELK.....	22
KIBANA DISCOVER PAGE	22
BUILD SIMPLE KIBANA HISTOGRAMS.....	27
Alerting with ELK	36
CREATE ALERTS IN LOGSTASH	36
ENRICH ALERTS WITH MESSAGE DETAIL	38
ADD CUSTOM PLUGINS TO LOGSTASH.....	39
INTEGRATE LOGSTASH WITH SLACK.....	40
INTEGRATE LOGSTASH WITH PAGERDUTY.....	45
INTEGRATE LOGSTASH WITH IBM NETCOOL OMNIBUS.....	48
Appendix: Logstash configuration	51

Introduction

IBM® Bluemix®, IBM's Platform as a Service (PaaS), allows you to rapidly build and deploy an application. Applications can run in a native or a hybrid Bluemix environment.

This document describes the integration of the Elasticsearch Logstash Kibana (ELK) log analytics open source stack into the Bluemix environment, how to collect logs from Bluemix applications (apps), and send alerts from those log records to selected alert/event notification targets (PagerDuty, Slack, and IBM Netcool Omnibus).

This document describes how to provision Bluemix virtual servers in the Bluemix environment, install the ELK stack on a Bluemix virtual server, and configure the ELK stack to integrate with other components deployed on the Bluemix environment.

The steps to integrate the ELK stack into the Cloud Architecture and Solution Engineering (CASE) environment are used as the basis for this document. For reference, both the ELK architecture within the CASE environment and the whole CASE environment are defined using system architecture diagrams, as well as diagrams defining the ELK structure and ELK integration into Bluemix architecture.

What is Elasticsearch Logstash Kibana?


Elasticsearch Logstash Kibana is an open source stack of three applications that work together to provide an end-to-end search and visualisation platform for the investigation of log file sources in real time. Log file records can be searched from a variety of log sources, charts, and dashboards built to visualize the log records. The three applications are Elasticsearch, Logstash, and Kibana.

Logstash

Logstash is an application that allows the ELK stack to manage the gathering, transformation, and transport of log file records. Logstash is able to pull records from remote sources or listen for records on specified ports.

Logstash has three areas of function:

- Process input of the log file records from the remote source. You can configure Logstash to receive records in a variety of methods, for example, [reading from a flat file](#) or from [a TCP port](#) or direct from a remote or local [syslog](#). These methods are defined in the [Logstash online support and input](#) section of the [Logstash .conf file](#). Once all input processing is completed, Logstash moves onto the next function.
- Transform and enrich the log records that take place in the [Logstash filter section](#). Any number of changes to the format (and content, if required) of the log file record can be made using the various methods provided with Logstash, for example, methods to add and enrich log record fields and the flagging. I describe excluding and limiting records for alerting later in this document. After the log record is transformed Logstash processes the record.
- Send the log records to the Elasticsearch application from the [output section](#) of the Logstash configuration. A specific [elasticsearch method](#) is included with the Logstash instance that sends the log records to the Elasticsearch instance. However, as with the input and filter



sections, a variety of methods are available to process the data sent from the filter section. For example, the records can be sent to external alerting tools such as [PagerDuty](#), social media tools such as Slack, or event management hubs like IBM Netcool Omnibus via output methods [syslog](#) or [http](#).

As an open source tool, non-Logstash employees develop other methods and share them with the user community for addition to the library of methods Logstash can execute (examples of this process are included later in this document).

Elasticsearch

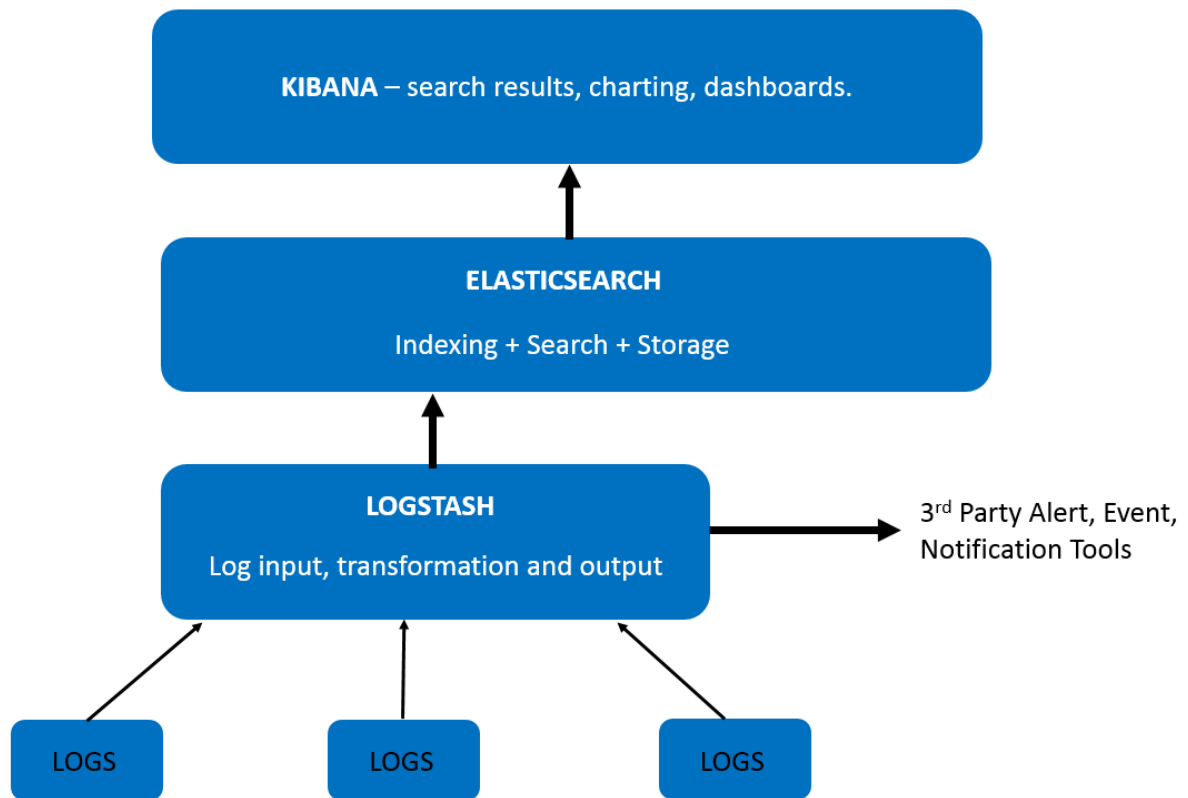
Elasticsearch is the index, store, and query application on the ELK stack. It provides a standard full text search facility of the log file records processed by Logstash. The ELK online help documentation describes Elasticsearch as follows: “It is generally used as the underlying engine/technology that powers applications that have complex search features and requirements.”

Kibana

Kibana is the user interface for the ELK stack. Kibana integrates with Elasticsearch to visualise the results of searches and allow the creation of charts and dashboards based on the searches. The searches, charts, and dashboard can visualise real-time or historical log records. Users access Kibana through an http URL from a client web browser.

ELK System Architecture

The ELK stack can be described as follows:



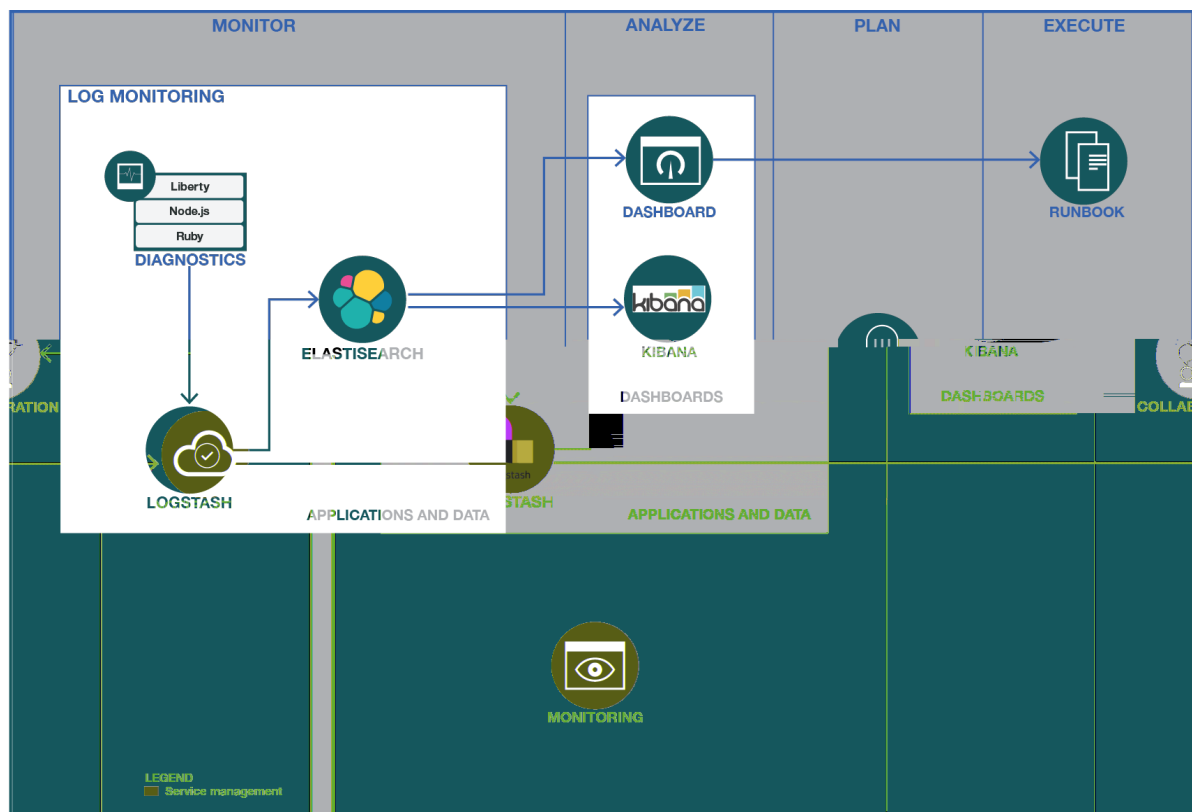
Service Management architecture

The IBM Cloud Architecture Center created a standard reference architecture for how to set up your cloud environment to handle incident management.

[View the standard Service Management architecture.](#)

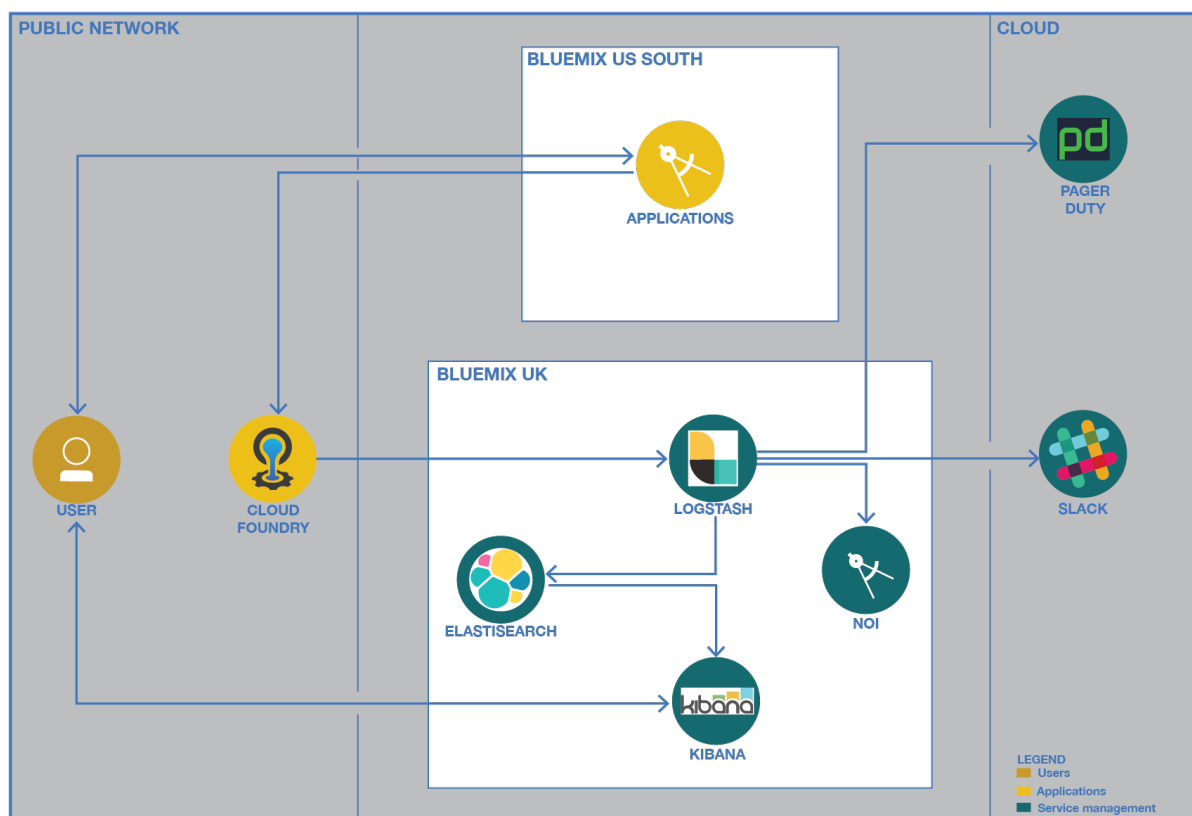
Log Monitoring System architecture

The following diagram shows a standard architecture for the log-monitoring system described in this tutorial.



Bluemix environment system architecture

The ELK stack can be installed into a variety of environments. When using Bluemix, the following figure shows what the environment might look like:



[The Bluemix cloud platform](#) offers a Public network and access to US South and UK environments. From all these environments, third-party, cloud-based solutions are also accessible.

The ELK stack is installed in the Bluemix United Kingdom environment because this is the environment where Bluemix has [virtual servers](#) available for provisioning. Other applications in the CASE project were also installed on virtual servers in the Bluemix United Kingdom region.

Bluemix apps were used to create a system to be monitored by ELK. These apps were hosted in the Bluemix US South environment using Bluemix microservices, which produce logs during normal operation and during crashes and outages.

Through the [Bluemix Cloud Foundry](#) process the log records can be broadcast from the Bluemix apps so Logstash can receive them and send them to Elasticsearch within the ELK stack or other virtual servers (hosting other IBM or third-party products) in the Bluemix United Kingdom environment or to third-party applications in the public cloud (Pager Duty, Slack etc.).

Creating a virtual server in Bluemix

With Bluemix, you can provision a virtual server in the cloud to install standalone or integrated system components. Not all Bluemix regions offer this service all the time due to beta-project constraints. In this article's examples, virtual servers were allowed to be provisioned in the United Kingdom region. For more information, read [how to provision a Bluemix virtual server](#) including some simple setup steps for ease of use.



Create an SSH key

To access your virtual server once it is provisioned, allocate an SSH key to the virtual server.

To generate a public/private RSA key pair, follow these steps:

1. In a UNIX or Linux® window, run the following command, specifying a <uniquekeyname> of your your choice and a password or passphrase (memorize the password):
`$ ssh-keygen -t rsa -f <uniquekeyname>`
2. Enter the passphrase (empty for no passphrase).
3. Enter the same passphrase again.

Your identification has been saved in <uniquekeyname>.

Your public key has been saved in <uniquekeyname>.pub.

The key fingerprint is:

d6:26:84:64:50:65:25:93:0c:e4:63:2c:69:3c:20:3c <user>@<server>

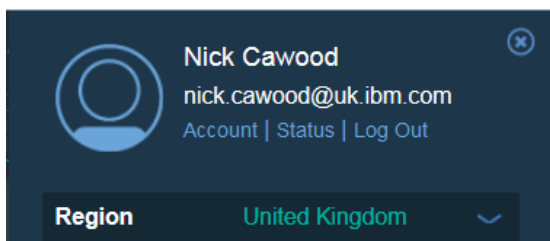
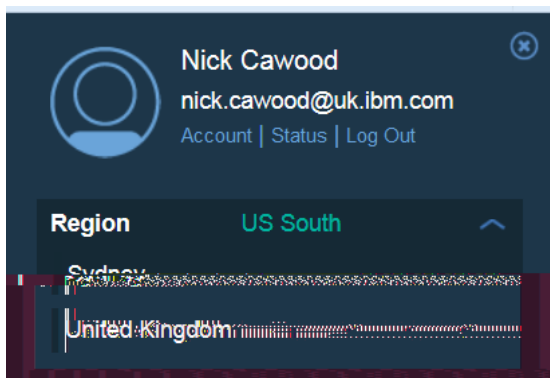
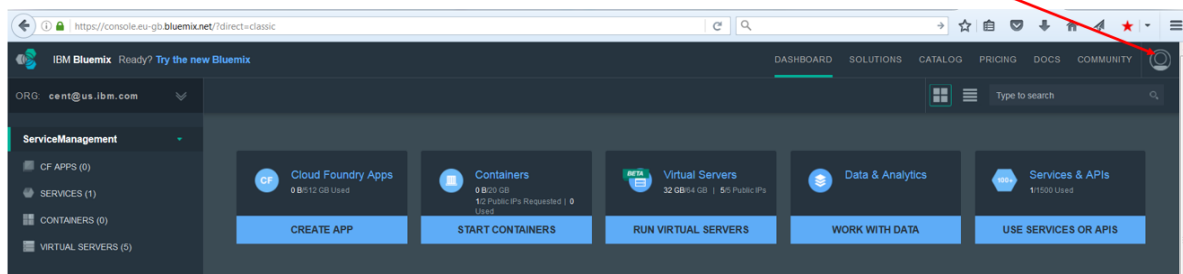
The key's randomart image is:

```
+--[ RSA 2048 ]-----+
| o . . +*+=o .      |
| E o * ooo          |
| . * * .            |
| . + o .            |
|          S o        |
|          . o        |
|                    |
|                    |
|                    |
+-----+
```

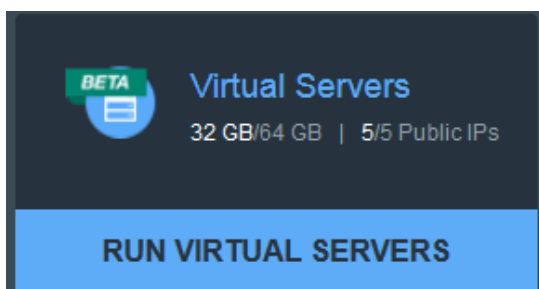
Two keys are created: public and private. The public key contents are used to create a virtual server with a public IP address. The private key is used to access the virtual server from a client machine.

Provisioning a virtual server

1. A Bluemix account is required for this process. Sign up [here](#) and log in. Make sure your environment is correct. For example, if you are in the United Kingdom, select the United Kingdom region in the Bluemix console:

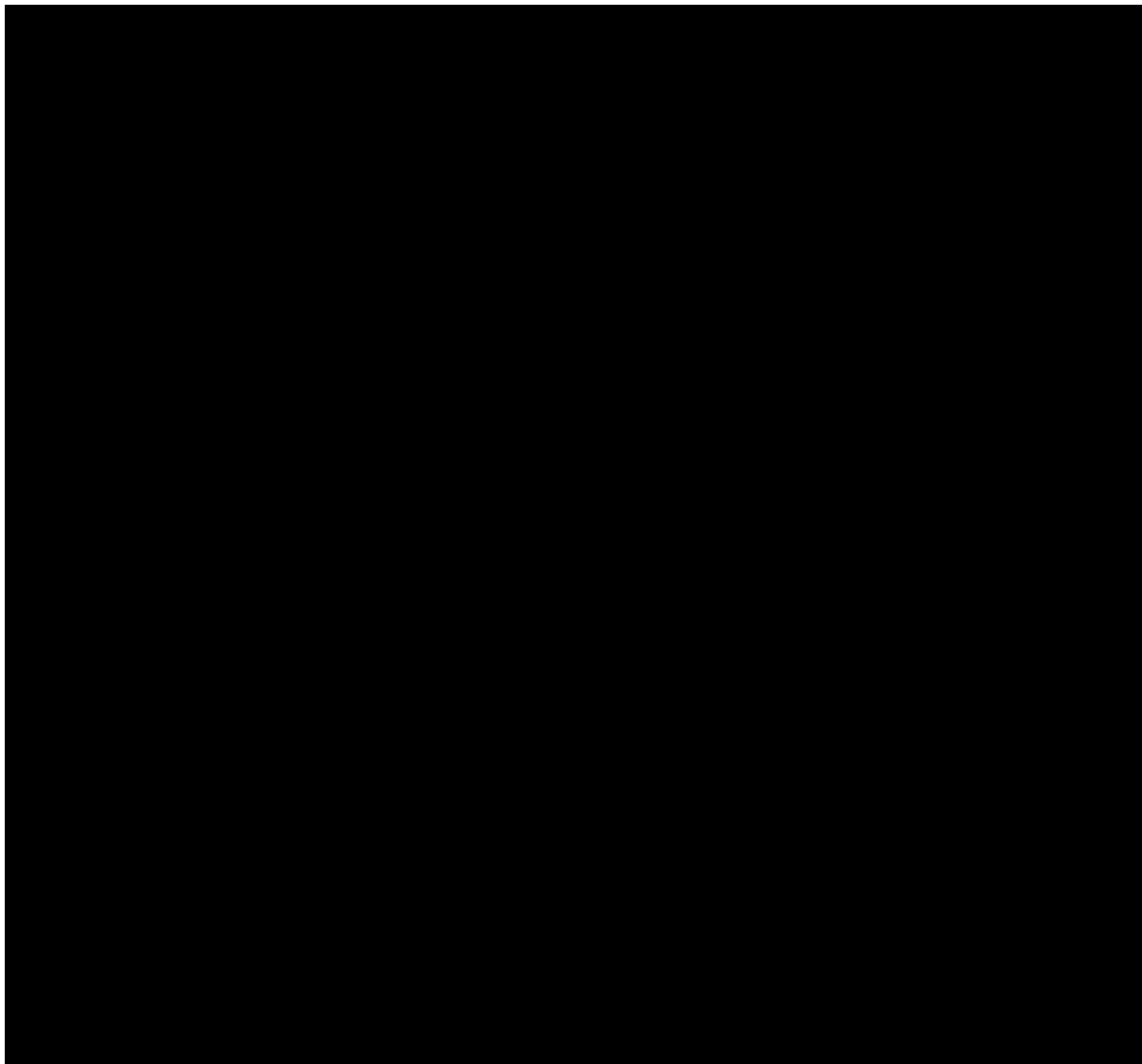


2. Add a virtual server to your Bluemix environment. Click **Run Virtual Servers**:



3. Select an OS image from the EXISTING list. For ELK, select **Centos OS** (use latest available version).

4. Name the virtual server.
5. Make sure **Assign public IP address** is checked.



6. Click **Add Key** to add the SSH key you created previously. Copy the contents of the public key into the Public Key to import field:

Add Key

Import your own public key, or create a new security key pair.

[Learn more](#)

IMPORT

CREATE

Key pair name:

uniquekeyname

Public Key to import:

ssh-rsa

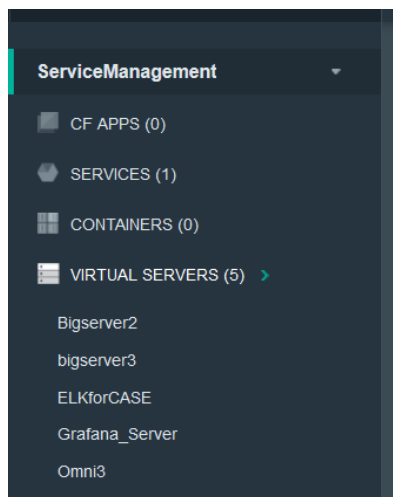
AAAAB3NzaC1yc2EAAAABIwAAAQEA1U24b8MskCcWN0p7iOlka5G30KI9oA2uh2ooZes16qXnaq
xasP9ZiLwvCehbRxyFfGtGmLOxWw9kJFqY6kuu5C3JJseD929sO6mvzIZyZGk2yZs5VIHYUiQEn1f
2IRWS0g7uBR7ib7EFgcCbgRE9RTIsVQesmgRKBTRhUJnKHPK2SZ712sY6EtFg1+0qBHmt2OQSI2h9
N+XsyJVPP7ZJV8vsnhP2WXuPJ6FwcQI6Anme1itRcEdIA9e/ZtlqTRqwbdm2ktonD9hcndWYzjrGGilx
3YrYrYvI1tuYhbPKnkT0JqEogrBHK99aqUG1unM07uwz7+aMD80PAaZG2955Q== user@server

CANCEL

OK

- Click **OK**. Then click **CREATE**.

The virtual server is provisioned and added to the Bluemix dashboard under VIRTUAL SERVERS:



Set up the virtual server

Log on to the virtual server

Now you can log on to the virtual server and add new users and programs to make it easier to access the virtual server.

To access the server, go to the UNIX or Linux window that you used to create the SSH key. In the directory where the <uniquekeyname> files exist, run the following command referencing the SSH keyname, the public IP address Bluemix allocated for the virtual server, and the password for the SSH key:

```
[user@server ~]$ ssh -i <uniquekeyname> ibmcloud@<publicIPAddress>
```

Enter the passphrase for the key <uniquekeyname>:

```
Last login: Tue Jul  5 12:55:46 2016 from  
108-210-96-231.lightspeed.rlghnc.sbcglobal.net
```

```
[ibmcloud@elkforcase ~]$
```

ibmcloud is the default user for the Bluemix virtual servers and allows SSH to access the server.

Access the root user

To access the root user, log on as ibmcloud user and run the following command:

```
[ibmcloud@elkforcase ~]$ sudo bash
```

```
[root@elkforcase ibmcloud]#
```

The root user can be used to add programs to the server and create and edit users as required to access applications installed on the virtual server.

Copy files to the virtual server

To move installation files to the server, go to the UNIX or Linux window you used to create the SSH key and the directory where the <uniquekeyname> files exist. Move the installation files to the same location.

Run the following command referencing the SSH keyname, the public IP address Bluemix allocated for the virtual server, and the password for the SSH key:

```
[user@server ~]$ scp -i <uniquekeyname> <INSTALLATIONFILENAME>
ibmcloud@<publicIPaddress>:/home/ibmcloud
```

Enter the passphrase for key cloudkey3.

This copies the file using the ibmcloud user to the /home/ibmcloud directory on the virtual server.

Install and access the virtual server via VNC server

Using the ssh command to access the virtual server is not the most stable method and does not offer [X-Windows support](#). Therefore, we suggest that you use a VNC client to access the virtual server.

Installing a VNC client should be done as the root user. Use the yum command as follows:

```
[ibmcloud@elkforcase ~]$ yum groupinstall "GNOME Desktop"
[ibmcloud@elkforcase ~]$ yum install tigervnc-server
```

These commands first install a desktop style OS user interface and then a VNC client to remotely access the desktop style OS.

Then, as either the root user or another user (example below uses elk), run the following command to create the VNC client connection:

```
[elk@elkforcase ~]$ vncserver
New 'elkforcase:2 (elk)' desktop is elkforcase:2
Starting applications specified in /home/elk/.vnc/xstartup
Log file is /home/elk/.vnc/elkforcase:2.log
[elk@elkforcase ~]$
```

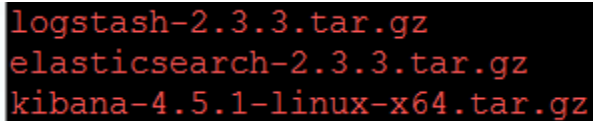
Install a VNC client on the client machine used to access the Internet and (in the example above) use the command <publicIPaddress>:2 and the password for ELK to access the virtual server.

Using a VNC client means that you can leave long duration installations running without fear of the terminal losing connection and the installations failing.

Install the ELK stack

Download the source software code for the ELK stack from the [ELK website](#).

At the time of writing the downloaded software files were:



```
logstash-2.3.3.tar.gz
elasticsearch-2.3.3.tar.gz
kibana-4.5.1-linux-x64.tar.gz
```

Transfer the files to the virtual server (using the scp method detailed above).

Install Elasticsearch

As a root user, create a non-root user with the [adduser](#) command.

Create a directory to install Elasticsearch into and untar the contents of the Elasticsearch install file into that directory:

```
[elk@elkforcase]$ cd /home
[elk@elkforcase]$ mkdir elasticsearch
[elk@elkforcase]$ cp /home/ibmcloud/elasticsearch-2.3.3.tar.gz
/home/elasticsearch
[elk@elkforcase]$ cd elasticsearch
[elk@elkforcase]$ gunzip elasticsearch-2.3.3.tar.gz
[elk@elkforcase]$ untar elasticsearch-2.3.3.tar
```

Elasticsearch is now installed.

Start Elasticsearch

To start Elasticsearch, go to the bin directory as the non-root user when you installed it and run the elasticsearch executable:

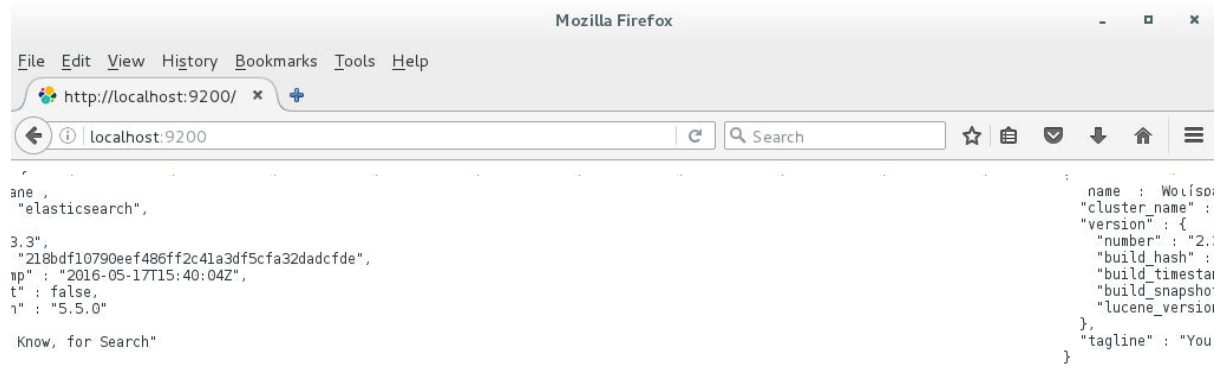
```
[elk@elkforcase]$ cd /home/elasticsearch/elasticsearch-2.3.3/bin
[elk@elkforcase]$ ./elasticsearch &
[2] 22211
[elk@elkforcase]$ [2016-07-05 13:38:57,730][INFO ][node] [Arsenal]
version[2.3.3], pid[22211], build[218bdf1/2016-05-17T15:40:04Z]
[2016-07-05 13:38:57,740][INFO ][node] [Arsenal] initializing ...
[2016-07-05 13:38:58,984][INFO ][plugins] [Arsenal] modules [reindex,
lang-expression, lang-groovy], plugins [], sites []
[2016-07-05 13:38:59,146][INFO ][env] [Arsenal] using [1] data paths,
mounts [[/ (rootfs)]], net usable_space [15.5gb], net total_space
[19.9gb], spins? [unknown], types [rootfs]
[2016-07-05 13:38:59,146][INFO ][env] [Arsenal] heap size [1015.6mb],
compressed ordinary object pointers [true]
```

```
[2016-07-05 13:38:59,147][WARN ][env] [Arsenal] max file descriptors
[4096] for elasticsearch process likely too low, consider increasing
to at least [65536]

[2016-07-05 13:39:03,526][INFO ][node] [Arsenal] initialized

[2016-07-05 13:39:03,526][INFO ][node] [Arsenal] starting ...
```

To test that Elasticsearch is running correctly, open the browser in the VNC client and run localhost:9200.



Install Kibana

ELK recommends installing and running Kibana as root.

1. As root user, create an installation directory and untar the installation file:

```
[root@elkforcase]$ cd /home
[root@elkforcase]$ mkdir kibana
[root@elkforcase]$ cp /home/ibmcloud/kibana-4.5.1-linux-x64.tar.gz /
home/kibana
[root@elkforcase]$ cd kibana
[root@elkforcase]$ gunzip kibana-4.5.1-linux-x64.tar.gz
[root@elkforcase]$ untar kibana-4.5.1-linux-x64.tar
```

2. Modify the manifest.yml configuration file. Go to the kibana-4.5.1-linux-x64 directory:

```
[root@elkforcase]$ cd /home/kibana/kibana-4.5.1-linux-x64
```

3. Edit the manifest.yml file so the contents are:

applications:

- name: kibana-in-bluemix

memory: 128M

host: kibana-in-bluemix

buildpack: <https://github.com/cloudfoundry-community/nginx-buildpack.git>

4. Edit the config/kibana.yml file so the contents are:

```
# Kibana is served by a back end server. This controls which
port to use.

port: 5601

# The host to bind the server to.

host: "0.0.0.0"

# If you are running kibana behind a proxy, and want to mount
it at a path,

# specify that path here. The basePath can't end in a slash.

# server.basePath: ""

# The maximum payload size in bytes on incoming server
requests.

server.maxPayloadBytes: 1048576

# The Elasticsearch instance to use for all your queries.

elasticsearch.url: "http://localhost:9200"

# preserve_elasticsearch_host true will send the hostname
specified in `elasticsearch`. If you set it to false,

# then the host you use to connect to *this* Kibana instance
will be sent.

elasticsearch.preserveHost: true

# Kibana uses an index in Elasticsearch to store saved
searches, visualizations

# and dashboards. It will create a new index if it doesn't
already exist.

kibana.index: ".kibana"

# The default application to load.


kibana.defaultAppId: "discover"

# If your Elasticsearch is protected with basic auth, these
are the user credentials

# used by the Kibana server to perform maintenance on the
kibana_index at startup. Your

# Kibana users will still need to authenticate with
Elasticsearch (which is proxied through

# the Kibana server)
```

```
# elasticsearch.username: "user"
# elasticsearch.password: "pass"
# SSL for outgoing requests from the Kibana Server to the
browser (PEM formatted)
# server.ssl.cert: /path/to/your/server.crt
# server.ssl.key: /path/to/your/server.key
# Optional setting to validate that your Elasticsearch backend
uses the same key files (PEM formatted)
# elasticsearch.ssl.cert: /path/to/your/client.crt
# elasticsearch.ssl.key: /path/to/your/client.key
# If you need to provide a CA certificate for your
Elasticsearch instance, put
# the path of the pem file here.
# elasticsearch.ssl.ca: /path/to/your/CA.pem
# Set to false to have a complete disregard for the validity
of the SSL
# certificate.
# elasticsearch.ssl.verify: true
# Time in milliseconds to wait for elasticsearch to respond to
pings, defaults to
# request_timeout setting
elasticsearch.pingTimeout: 1500
# Time in milliseconds to wait for responses from the back end
or elasticsearch.
# This must be > 0
elasticsearch.requestTimeout: 30000
# Time in milliseconds for Elasticsearch to wait for responses
from shards.
# Set to 0 to disable.
elasticsearch.shardTimeout: 0
# Time in milliseconds to wait for Elasticsearch at Kibana
startup before retrying
# elasticsearch.startupTimeout: 5000
elasticsearch.startupTimeout: 30000
```

Set the path to where you would like the process id file to be created.

```
pid.file: /var/run/kibana.pid
```

If you would like to send the log output to a file you can set the path below.

```
logging.dest: stdout
```

Set this to true to suppress all logging output.

```
logging.silent: false
```

Set this to true to suppress all logging output except for error messages.

```
logging.quiet: false
```

Set this to true to log all events, including system usage information and all requests.


```
logging.verbose: false
```

Kibana is now installed.

Start Kibana

To start Kibana, go to the bin directory and run the kibana executable:

```
[root@elkforcase]$  
[root@elkforcase]$ cd /home/kibana/kibana-4.5.1-linux-x64/bin  
[root@elkforcase]$ ./kibana &  
[1] 22283  
[root@elkforcase]# log [09:59:00.542] [info][status]  
[plugin:kibana] Status changed from uninitialized to green - Ready  
log [09:59:00.600] [info][status][plugin:elasticsearch] Status  
changed from uninitialized to yellow - Waiting for Elasticsearch  
log [09:59:00.689] [info][status][plugin:kbn_vislib_vis_types]  
Status changed from uninitialized to green - Ready  
log [09:59:00.717] [info][status][plugin:markdown_vis] Status  
changed from uninitialized to green - Ready  
log [09:59:00.721] [info][status][plugin:metric_vis] Status changed  
from uninitialized to green - Ready  
log [09:59:00.814] [info][status][plugin:spyModes] Status changed  
from uninitialized to green - Ready  
log [09:59:00.934] [info][status][plugin:statusPage] Status  
changed from uninitialized to green - Ready
```



```
log    [09:59:00.941] [info][status][plugin:table_vis] Status changed
from uninitialized to green - Ready

log    [09:59:01.067] [info][listening] Server running at http://
0.0.0.0:5601

log    [09:59:06.083] [info][status][plugin:elasticsearch] Status
changed from yellow to yellow - No existing Kibana index found

log    [09:59:08.942] [info][status][plugin:elasticsearch] Status
changed from yellow to green - Kibana index ready
```

Kibana is now started.

Broadcast Bluemix app log records

Bluemix apps (which are shown in the US South Bluemix region of the System Architecture diagram) will produce logs in a syslog rfc5424 format using a Bluemix tool called [Loggregator](#).

For Logstash to receive these logs, each Bluemix app needs to broadcast these logs. To do that, run the following commands on each Bluemix app using the public IP address of the ELK virtual server, the ELK tag to trace the process, and insert the name of each app in turn:

```
cf cups elk -l syslog://<publicIPAddress:5000
cf bind-service <myapp> elk
cf restage <myapp>
```

Install Logstash

To install Logstash, as root user create a directory for the installation and untar the installation file there:

```
[root@elkforcase]$ cd /home
[root@elkforcase]$ mkdir logstash
[root@elkforcase]$ cp /home/ibmcloud/logstash-2.3.3.tar.gz /home/
logstash
[root@elkforcase]$ cd logstash
[root@elkforcase]$ gunzip logstash-2.3.3.tar.gz
[root@elkforcase]$ untar logstash-2.3.3.tar
```

Write Logstash configuration

Logstash requires a configuration file to be written to receive, transform, and output the log records to Elasticsearch. This section explains the basic requirements of processing Bluemix app log records into Elasticsearch (more complex configurations are detailed later in this document for the integrations with third-party tools).

Create a .conf file as root user:

```
[root@elkforcase]$ cd /home/logstash/logstash-2.3.3
[root@elkforcase]$ mkdir config
```



```
[root@elkforcase]$ cd config
[root@elkforcase]$ touch loggregator.conf
```

Edit the loggregator file with the following contents:

Input section defines which log records to receive.

port and type reflect method selected for Cloud Foundry to broadcast the records.

```
input {
  tcp {
    port => 5000
    type => syslog
  }
}
```

Filter section – transforms the data.

```
filter {
# Add a field to identify the records as Cloud Foundry based.
  mutate {
    add_field => { "format" => "cf" }
  }
}
```

Output section – sends the data to its destination(s).

```
output {
# Use the elasticsearch plugin to send the data to the elasticsearch instance.
  elasticsearch {
    hosts => localhost
  }
  stdout { codec => rubydebug }
}
```

End of Config.

Save the file. Verify the format of the file is acceptable by running the following commands:

```
[root@elkforcase]$ cd /home/logstash/logstash-2.3.3
[root@elkforcase]$ bin/logstash agent --configtest -f config/
loggregator.conf
```

Configuration is OK.

Start Logstash

After you have written and verified the Logstash configuration, you can start Logstash to process records from the Bluemix apps (via Cloud Foundry). Run the following as root user:

```
[root@elkforcase]$ cd /home/logstash/logstash-2.3.3
[root@elkforcase]$ bin/logstash agent -f config/loggregator.conf
```

Logstash is now started.

Monitor Logstash processing

Once Logstash is running and log records are being produced by the Bluemix apps, you can monitor progress in the same window that was used to start Logstash. Output is similar to the following:

```
{
  "message" => "270 <14>1 2016-07-08T10:47:45.188962+00:00 loggregator 0c677e82-5bf
c-4e38-9155-e416249e3ee6 [APP/0] - - {\"v\":0,\"level\":20,\"name\": \"newrelic\", \"hostname\": \"19n32r6pk9v\", \"pid\":62,\"time\": \"2016-07-08T10:47:45.188Z\", \"msg\": \"No new
metric mappings from server\", \"component\": \"mapper\" }",
  "@version" => "1",
  "@timestamp" => "2016-07-08T10:47:45.352Z",
  "host" => "75.126.167.71",
  "port" => 55008,
  "type" => "syslog",
  "format" => "cf",
  "num1" => "270",
  "num2" => "14",
  "num3" => "1",
  "tstamp" => "2016-07-08T10:47:45.188962+00:00",
  "CFsource" => "loggregator",
  "data1" => "ms-catalog-postpyretic-pomiculture",
  "data2" => "APP/0",
  "dash1" => "-",
  "dash2" => "-",
  "msgtosend" => "{\"v\":0,\"level\":20,\"name\": \"newrelic\", \"hostname\": \"19n32r6pk9v\", \"pid\":62,\"time\": \"2016-07-08T10:47:45.188Z\", \"msg\": \"No new metric mappings
from server\", \"component\": \"mapper\" }",
  "tags" => [
    "grokked",
    "translated"
  ]
}
```

The log records being broadcast from the Bluemix apps are now sent to Elasticsearch from the Logstash output configuration. Records are sent in real time. If Logstash is not running while log records are being produced, then these records will not be processed by Logstash and will not be stored in Elasticsearch.

Use ELK

The log records from the Bluemix apps are now being stored and indexed by the Elasticsearch component. This data can be accessed via the Kibana application.

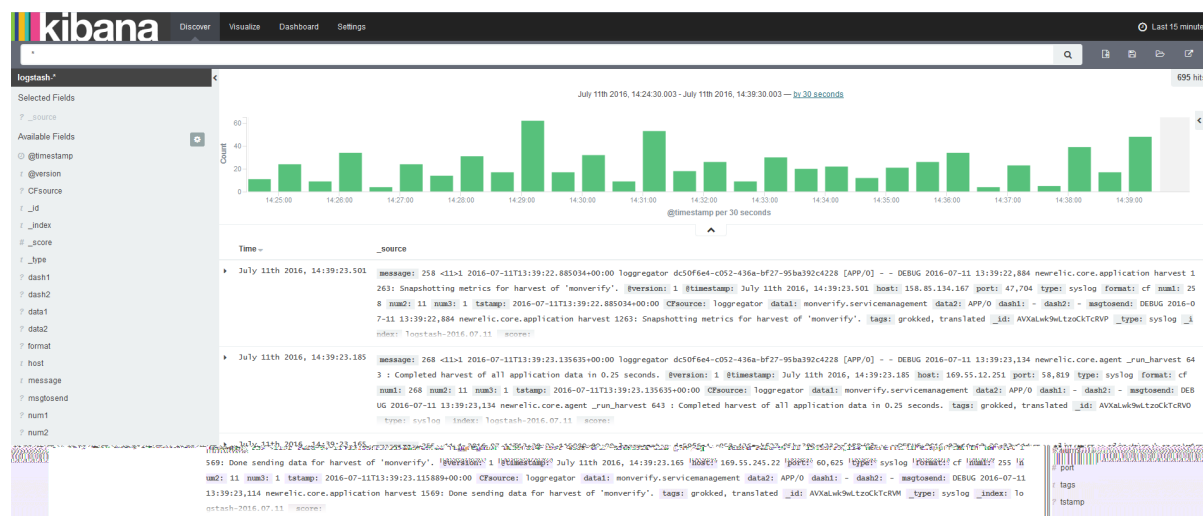
Kibana Discover page

Using a web browser, either directly from a client or using the VNC client, access Kibana using the publicIPAddress of the ELK virtual server:

<http://<publicIPAddress>:5601>

No user/password credentials are required.

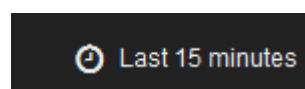
Kibana defaults to a search of all records (*) in the last 15 minutes and returns the results:



[More complex searches are possible](#) by changing the time filter and supplying search queries.

Change the time filter

To change the time ranges, click the clock symbol in the top-right of the Kibana Discover page:



You will see the following options:

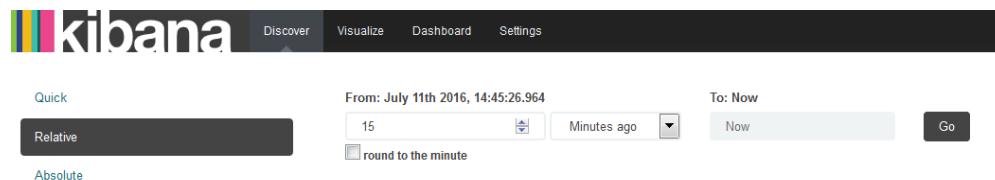
kibana				
Discover				
Quick				
Relative				
Absolute				
Today	Yesterday	Last 15 minutes	Last 30 days	
This week	Day before yesterday	Last 30 minutes	Last 60 days	
This month	This day last week	Last 1 hour	Last 90 days	
This year	Previous week	Last 4 hours	Last 6 months	
The day so far	Previous month	Last 12 hours	Last 1 year	
Week to date	Previous year	Last 24 hours	Last 2 years	
Month to date		Last 7 days	Last 5 years	
Year to date				



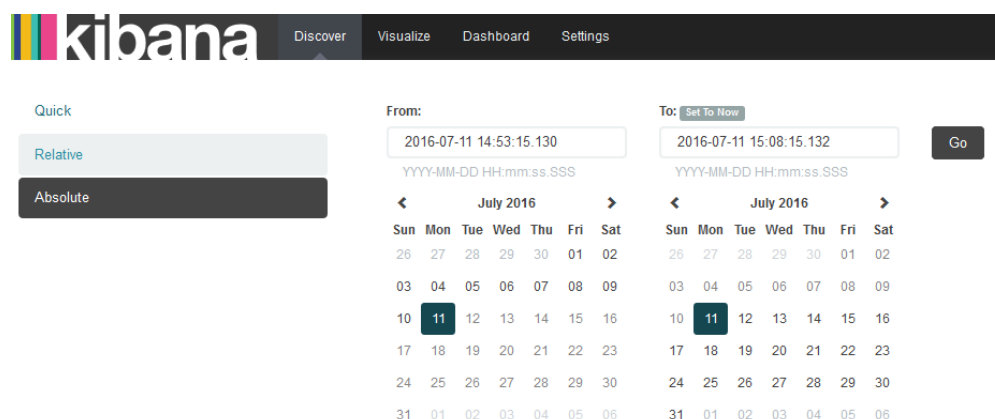
A variety of quick time ranges can be selected or you can switch to relative or absolute time ranges.

Quick time ranges allow switching between last X minutes to last Y hours or Z days etc.

Relative time ranges allow more specific versions of the Quick options, from specific minutes/hours/days to now:

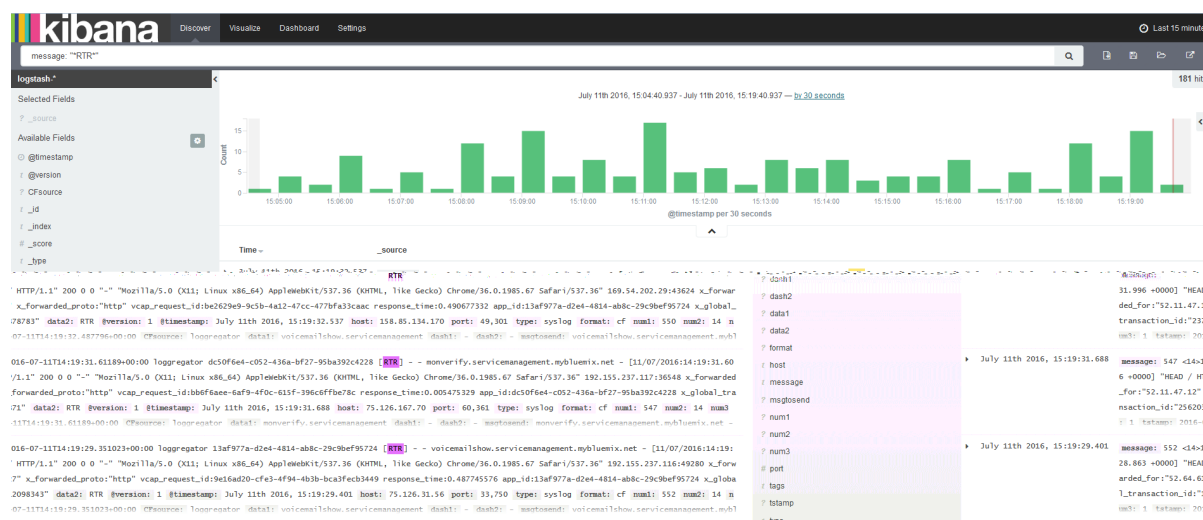


Absolute time ranges specify the start and end times for the searches:



Custom search queries

You can use the text window of the Discover page to search for matches within the log records. You can use [Apache Lucene](#) or [Elasticsearch query DSL](#) syntax.

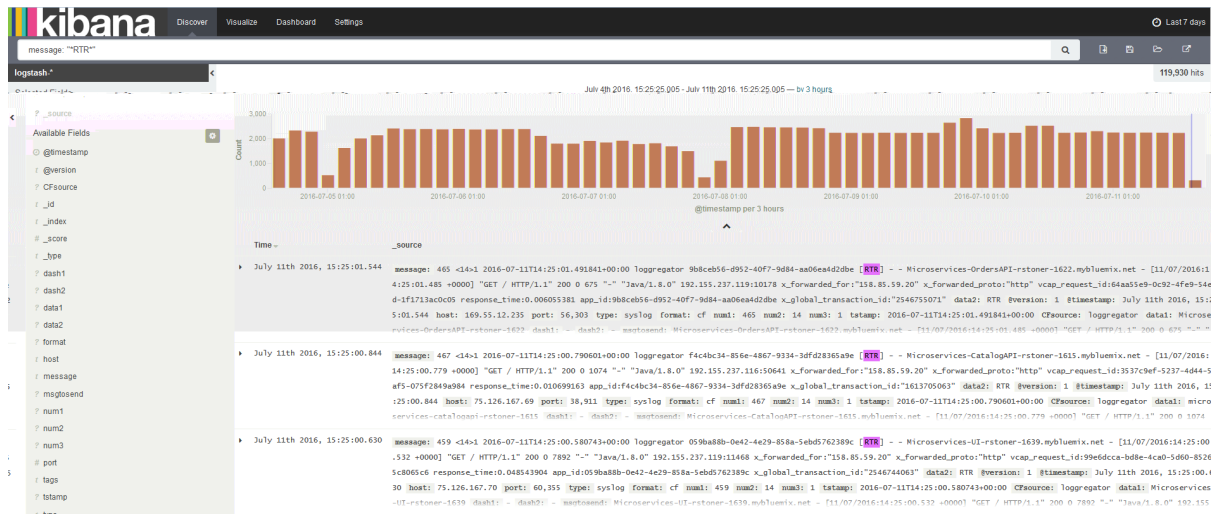


Saved searches

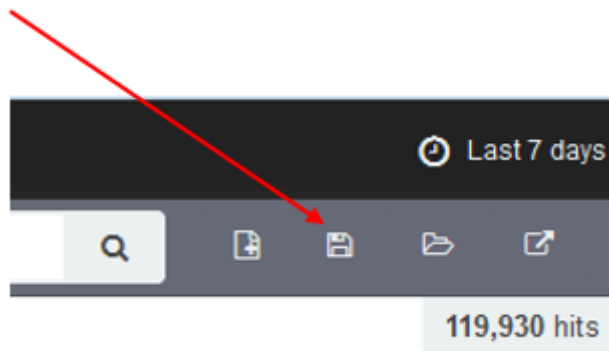
You can save any custom search syntax and choice of time range for future ease of use. Any relative or absolute time frame is saved as part of the search.

To save a search:

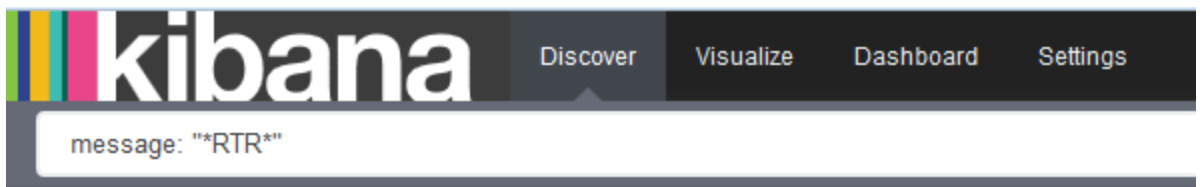
1. Set up and run the required syntax and time frame:



2. Click the **Save Search** icon:

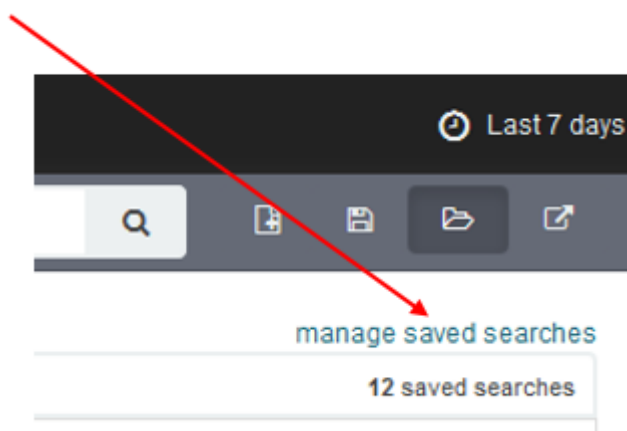


3. Type in a name for the saved search:

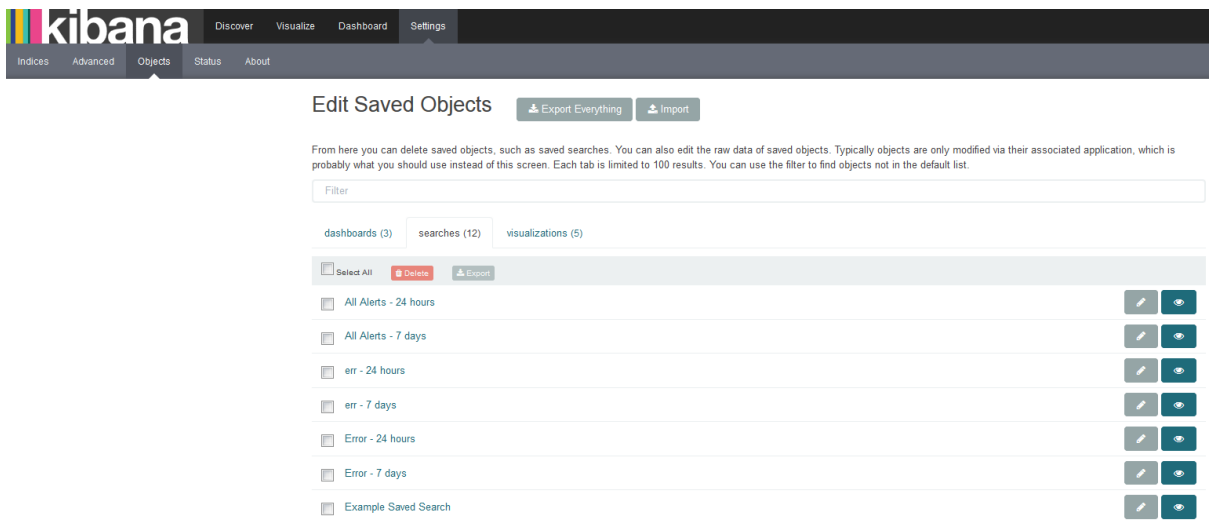


Saved Search Filter
All Alerts - 24 hours
All Alerts - 7 days
Error - 24 hours
Error - 7 days
Example Saved Search

7. To delete or update saved searches, click **manage saved searches**:



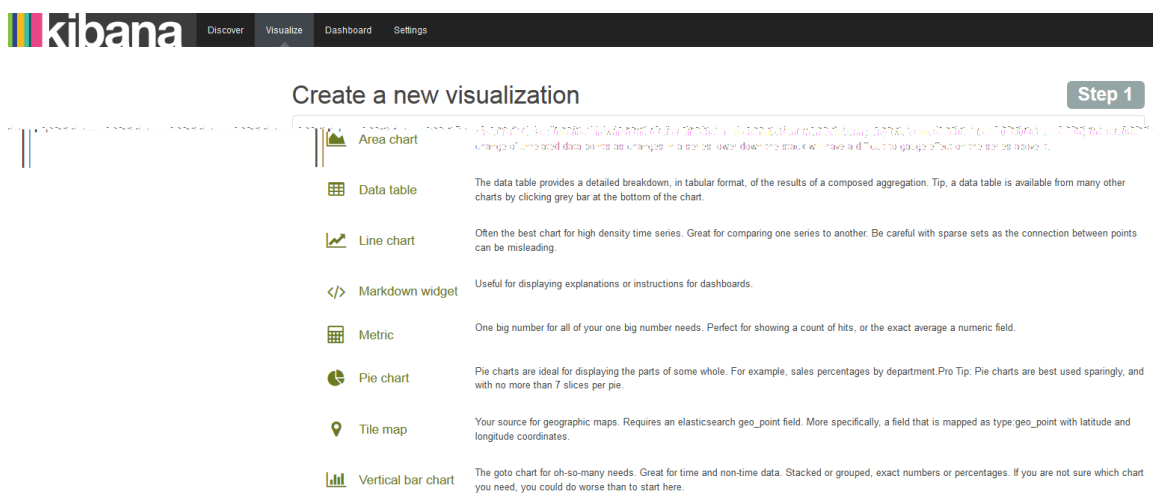
Use the edit icons to make any changes.



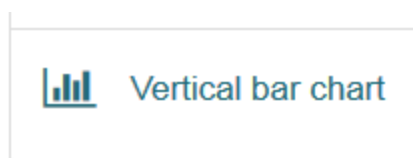
Build simple Kibana histograms

Kibana allows you to build charts and dashboards based on the search capability offered by the Discover page. Saved searches can be used as the basis for charts and dashboards built using those charts. To build a simple histogram:

1. Select the **Visualize** page:



2. Select **Vertical Bar Chart**:



3. To use a saved search, click **From a saved search**:

Select a search source

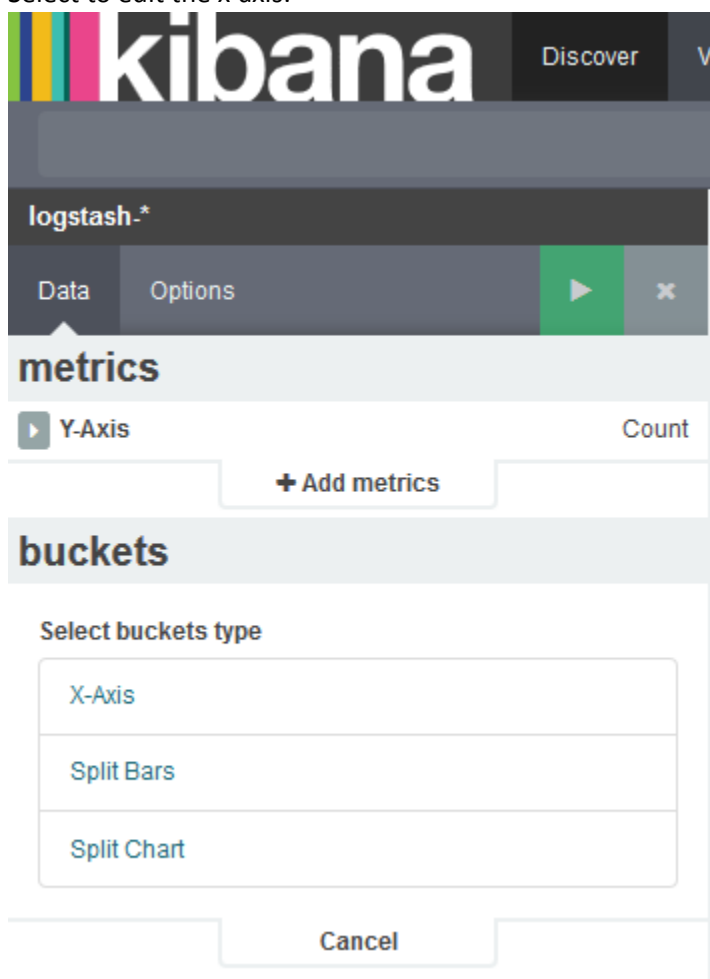
[From a new search](#)

[From a saved search](#)

- Select a previously saved search:

[Example Saved Search](#)

- Select to edit the x-axis:



- Edit the fields with information about the aggregation, field, interval and custom label:

kibana

Discover

Visualize

logstash-*

Data

Options

metrics

Y-Axis

Count

+ Add metrics

buckets

X-Axis

Aggregation

Date Histogram

Field

@timestamp

Interval

Daily

CustomLabel

Example Custom Label

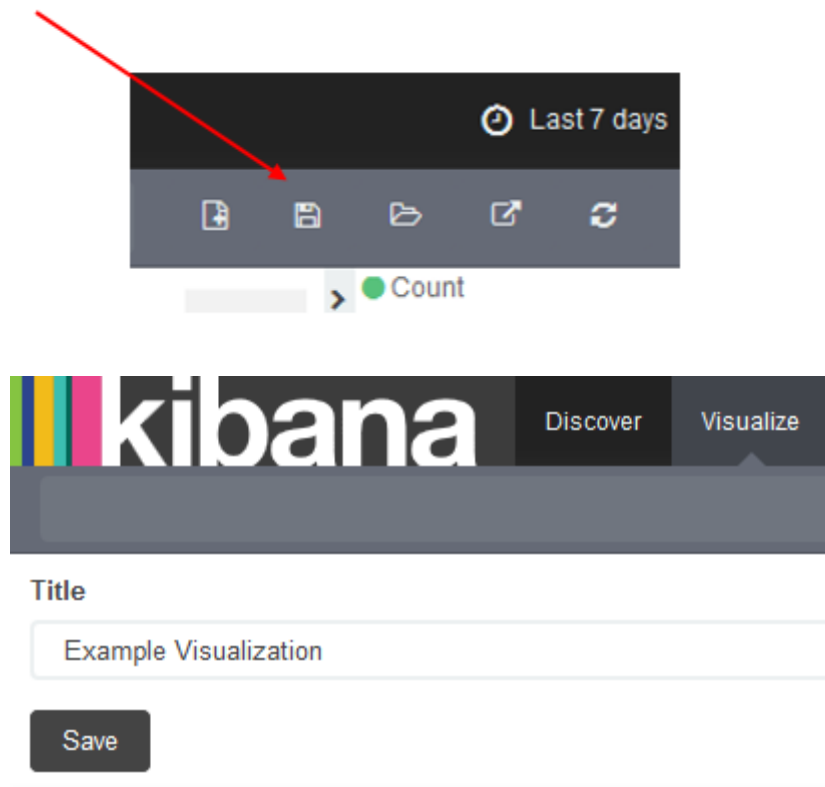
Advanced

+ Add sub-buckets

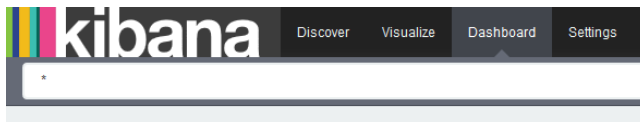
7. Click the green arrow. The histogram is created and run based on the saved search:



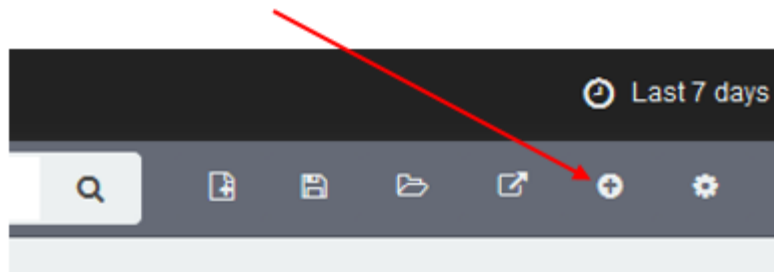
8. To save the histogram, click the **Save Visualization** icon and add a title:



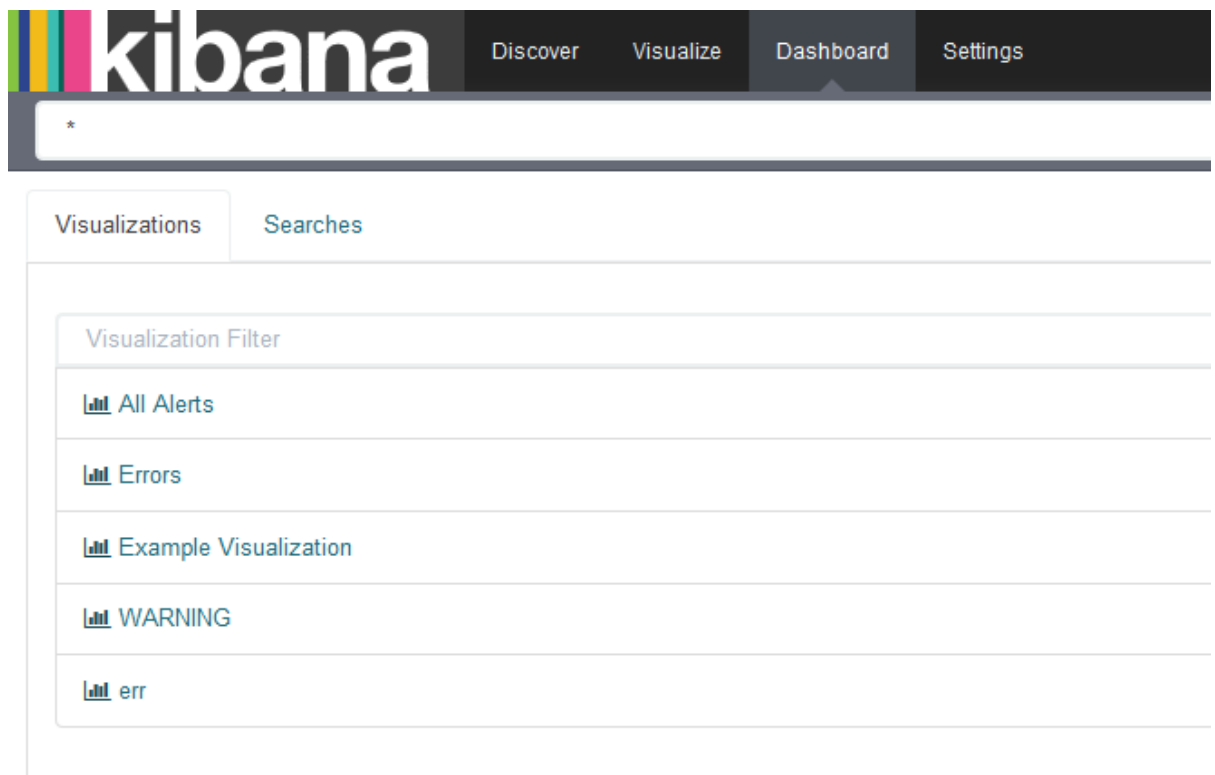
9. Click **Save**.
10. To view the visualisation, click the **Dashboard** page:

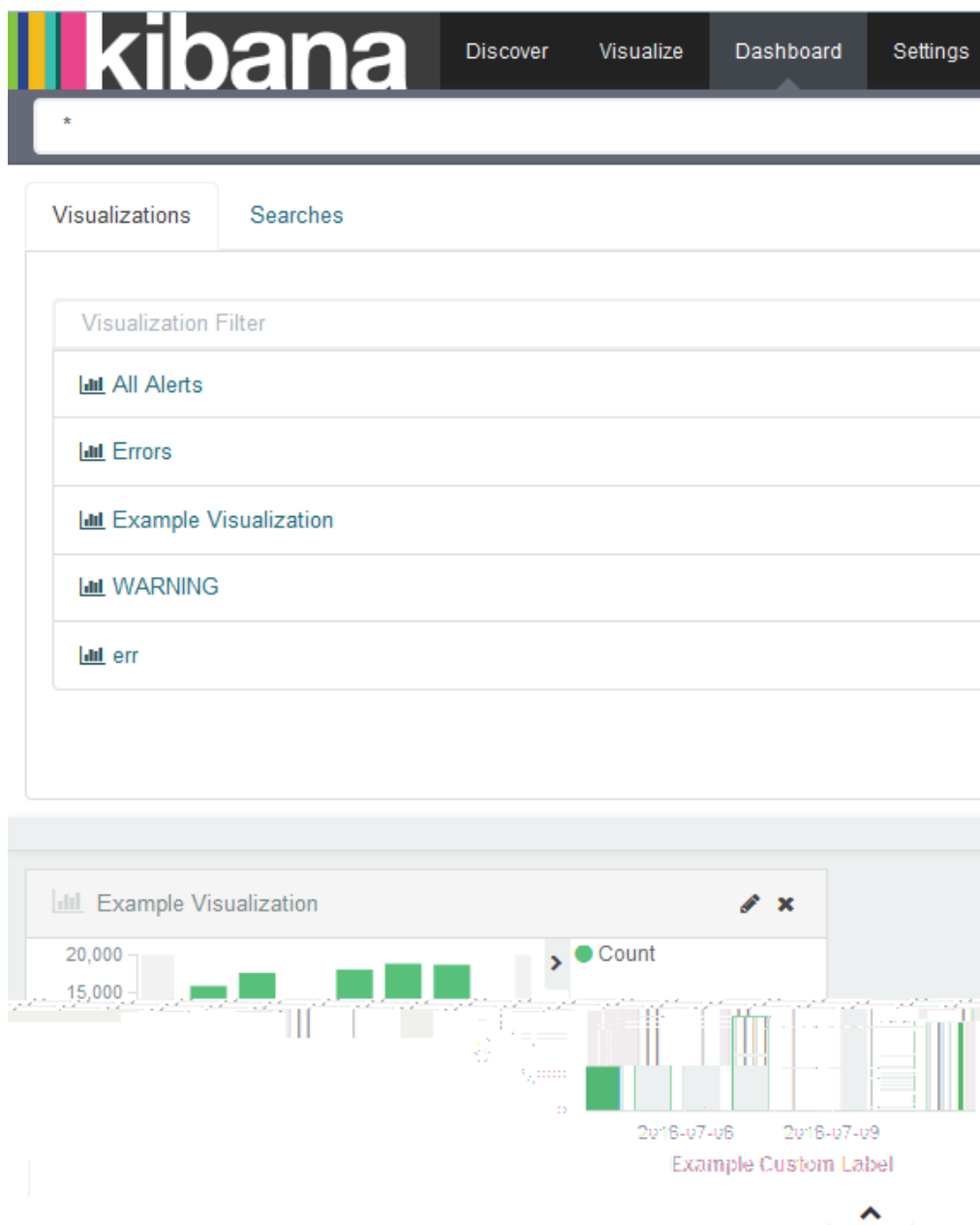


11. Click the **Add Visualization** icon:

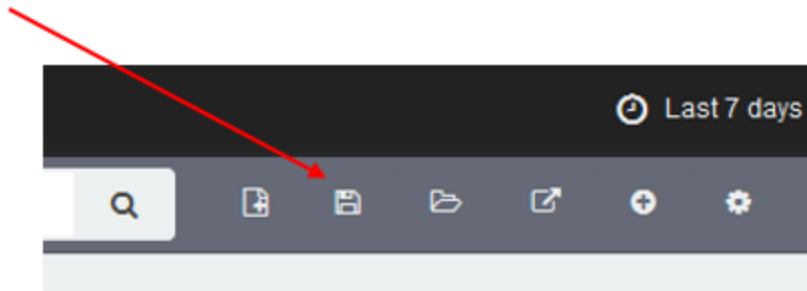


12. Select the visualization that you just created:





13. To save the dashboard, click the **Save Dashboard** icon:

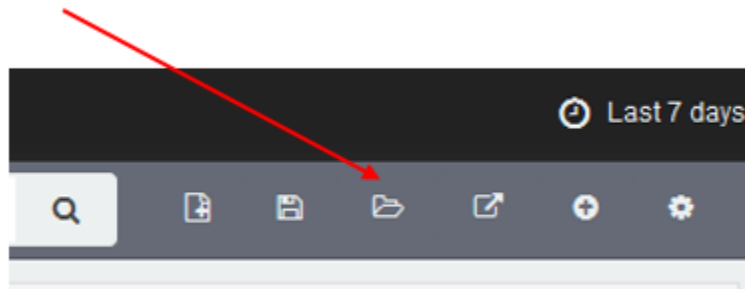


14. Add a title and select whether to store the selected time frame with the dashboard:

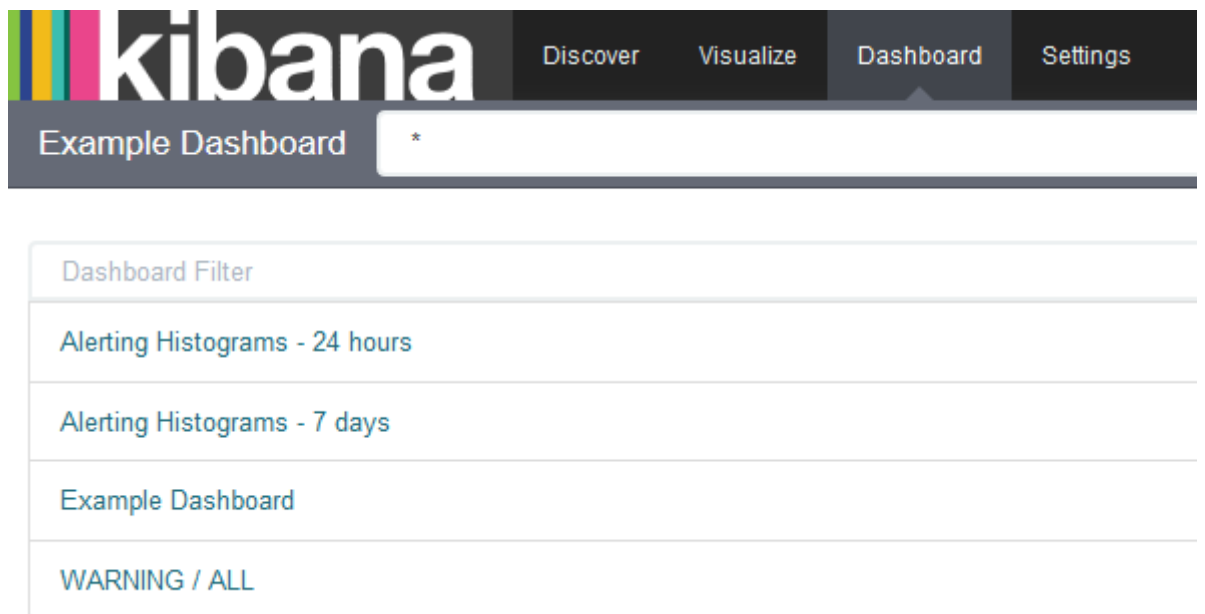
15. Click **Save**.

16. You can edit the shape and number of charts and save the dashboard.

17. To re-run the dashboard, select the Dashboard page and click the **Load Saved Dashboard** icon:



18. Select the saved dashboard from the list:



19. The dashboard runs:



Alerting with ELK

ELK examines the log records in process to send alerts to third-party alerting, event management, and social media applications. As shown previously in the ELK architecture, the ELK strategy for alerting is to send alerts from the Logstash component directly. Therefore, log records and alerts are sent to both Elasticsearch and third-party tools at the same instant using the output section of the Logstash configuration. Any alerts are generated during the filter section processing of the Logstash configuration.

Create alerts in Logstash

The Logstash filter section is used to [trigger alerts](#) based on the log records being processed. For example, log records can be checked for errors, warnings, crashed or stopped messages, and generate an alert.

Here is an example Logstash filter code:

```
if [message] =~ /.*(err).*/ {
    mutate { add_tag => "servicename_alarm_err" }
}

if [message] =~ /.*(Error).*/ {
    mutate { add_tag => "servicename_alarm_error" }
}

if [message] =~ /.*(fatal).*/ {
    mutate { add_tag => "servicename_alarm_fatal" }
}

if [message] =~ /.*(WARNING).*/ {
    mutate { add_tag => "servicename_alarm_warning" }
}

if [message] =~ /.*(STOPPED|Stopped|CRASHED).*/ {
    mutate { add_tag => "servicename_alarm_stopped" }
}
```

The specific tags are set against the log records to trigger the alert later in the configuration in the output section.

Control the number of generated alerts using the [Logstash throttle plugin](#) so that either a lot of instances in a fixed time period are required before an alert is sent or a specific number of instances triggers the alert.

Edit the plugin parameters to meet system requirements:

```
if "servicename_alarm_err" in [tags] {
    throttle {
        after_count => 5
        period => 300
        key => "servicename_alarm_err"
        add_tag => "servicename_throttled"
    }
}

if "servicename_alarm_error" in [tags] {
    throttle {
        after_count => 10
        period => 600
        key => "servicename_alarm_error"
        add_tag => "servicename_throttled"
    }
}

if "servicename_alarm_fatal" in [tags] {
    throttle {
        after_count => 20
        period => 900
        key => "servicename_alarm_fatal"
        add_tag => "servicename_throttled"
    }
}

if "servicename_alarm_warning" in [tags] {
    throttle {
        after_count => 30
        period => 1200
        key => "servicename_alarm_warning"
```

```

        add_tag => "servicename_throttled"
    }
}

if "servicename_alarm_stopped" in [tags] {
    throttle {
        after_count => 40
        period => 1500
        key => "servicename_alarm_stopped"
        add_tag => "servicename_throttled"
    }
}

```

Enrich alerts with message detail

Some of the third-party integrations benefit from having components of the raw log record sent in addition to the raw record or fields in the record edited, transformed, or translated.

Grok

Logstash allows the log record to be broken up into logical sections and each section stored in a new parameter using [the grok parameter](#) in the filter section. Here is an example grok:

```

grok {
    match => { "message" => "%{NUMBER:num1} \<%{NUMBER:num2}\>%
{NUMBER:num3} %{TIMESTAMP_ISO8601:tstamp} %{WORD:CFsource} %{DATA:data1} \
[%{DATA:data2}\] %{NOTSPACE:dash1} %{NOTSPACE:dash2} %
{GREEDYDATA:msgtosend}" }

    add_tag => ["grokked"]
} #end grok

```

To build the grok statement, Logstash offers an online tool at <https://grokdebug.herokuapp.com/>.

Translate

If specific fields in the log record are coded (for example, the Bluemix app name is often not sent in the syslog messages, instead the GUID is sent which is not useful) then the [filter section translate plugin](#) can be used to add new fields by holding the decoded versions of those values:

```

translate {
    field => "data1"
    destination => "data1"
    override => "true"
}

```

```

        dictionary => [ "13af977a-d2e4-4814-
ab8c-29c9bef95724", "voicemailshow.servicemanagement",
                        "dc50f6e4-c052-436a-
bf27-95ba392c4228", "monverify.servicemanagement",
                        "0c677e82-5bfc-4e38-9155-
e416249e3ee6", "ms-catalog-postpyretic-pomiculture",

                        "f4c4bc34-856e-4867-9334-3dfd28365a9e", "microservices-catalogapi-
rstoner-1615",

                        "059ba88b-0e42-4e29-858a-5ebd5762389c", "Microservices-UI-rstoner-1639",
                        "9b8ceb56-d952-40f7-9d84-
aa06ea4d2dbe", "Microservices-OrdersAPI-rstoner-1622" ]

        add_tag => "translated"
    }

```

However, the translate plugin is not included with the Logstash installation. In the next section, I will describe how to add new plugins to the Logstash instance.

Add custom plugins to Logstash

As an open source tool, Logstash encourages users to develop content for all users to share. Some of the plugins that were created by our community include a translation plugin, an output Slack plugin, and an output syslog plugin. Let's look at how to add these plugins to your environment.

Filter translate plugin

To add the translate plugin:

1. [Download the plugin](#).
2. Copy the plugin file, `logstash-filter-translate-2.1.4.gem`, to the ELK virtual server (see previous section in this document).
3. As a root user, move the file to `/home/logstash/logstash-2.3.3/plugins`, then edit the file `/home/logstash/logstash-2.3.3/Gemfile` to add the line:

```
gem "logstash-filter-translate", "2.1.4"
```

4. Run the command:

```
[root@elkforcase logstash-2.3.3]# /home/logstash/logstash-2.3.3/bin/
logstash-plugin install --no-verify
```

The plugin installs and is ready to use in a Logstash filter.

Output Slack plugin

To add the Slack plugin:



1. [Download the plugin.](#)
2. Copy the plugin file, logstash-output-slack-0.1.4.gem, to the ELK virtual server (see previous section in this document).
3. As a root user, move the file to /home/logstash/logstash-2.3.3/plugins, then edit the file /home/logstash/logstash-2.3.3/Gemfile to add the line:

```
gem "logstash-output-slack", "0.1.4"
```

4. Run the command:

```
[root@elkforcase logstash-2.3.3]# /home/logstash/  
logstash-2.3.3/bin/logstash-plugin install --no-verify
```

The plugin installs and is ready to use in a Logstash filter.

Output syslog plugin

To add the syslog plugin:

1. [Download the plugin.](#)
2. Copy the plugin file, logstash-output-syslog-2.1.4.gem, to the ELK Virtual Server (see previous section in this document).
3. As a root user, move the file to /home/logstash/logstash-2.3.3/plugins, then edit the file /home/logstash/logstash-2.3.3/Gemfile to add the line:

```
gem "logstash-output-syslog", "2.1.4"
```

4. Run the command:

```
[root@elkforcase logstash-2.3.3]# /home/logstash/  
logstash-2.3.3/bin/logstash-plugin install --no-verify
```

The plugin installs and is ready to use in a Logstash filter.

Integrate Logstash with Slack

To send generated alerts to Slack, complete these steps:

1. Create a Slack [integration](#).
2. Use the Slack plugin in the Logstash output.

To create the Slack integration:

1. Login to Slack and go to the [Apps page](#).

Incoming WebHooks

Incoming WebHooks are a simple way to post messages from external sources into Slack. They make use of normal HTTP requests with a JSON payload, which includes the message and a few other optional details described later.

[Message Attachments](#) can also be used in Incoming WebHooks to display richly-formatted messages that stand out from regular chat messages.

Add Configuration

Help and support >

Privacy policy >

Configurations

- Posts to #cloud-operations-mon as incoming-webhook
Robert Barron on May 25, 2016
- Posts to #cloud-operations-mon as incoming-webhook
Robert Barron on Jun 22, 2016
- Posts to #cloud-operations-mon as incoming-webhook
Nick Cawood on Jun 24, 2016
- Posts to #cloud-operations-mon as ANSBMX
Ray Stoner on Jun 28, 2016
- Posts to #cloud-operations-mon as incoming-webhook
James Williams on Jul 8, 2016

3. To add a new WebHook, click **Add Configuration** and select a Slack channel to integrate with:

Incoming WebHooks
Send data into Slack in real-time.

Incoming WebHooks are a simple way to post messages from external sources into Slack. They use normal HTTP requests with a JSON payload, which includes the message and a few other optional details described later.

[Message Attachments](#) can also be used in Incoming WebHooks to display richly-formatted messages.

New to Slack integrations?
Check out our [Getting Started](#) guide to familiarize yourself with the most common types while building your own. You can also [register as a developer](#) to let us know what you're updates to our APIs.

Post to Channel
Start by choosing a channel where your Incoming Webhook will post messages to.

#cloud-operations-mon

Add Incoming WebHooks integration

By creating an incoming webhook, you agree to the [Slack API Terms of Service](#).

A WebHook is created with the Webhook URL to use to integrate Logstash with Slack:

Setup Instructions

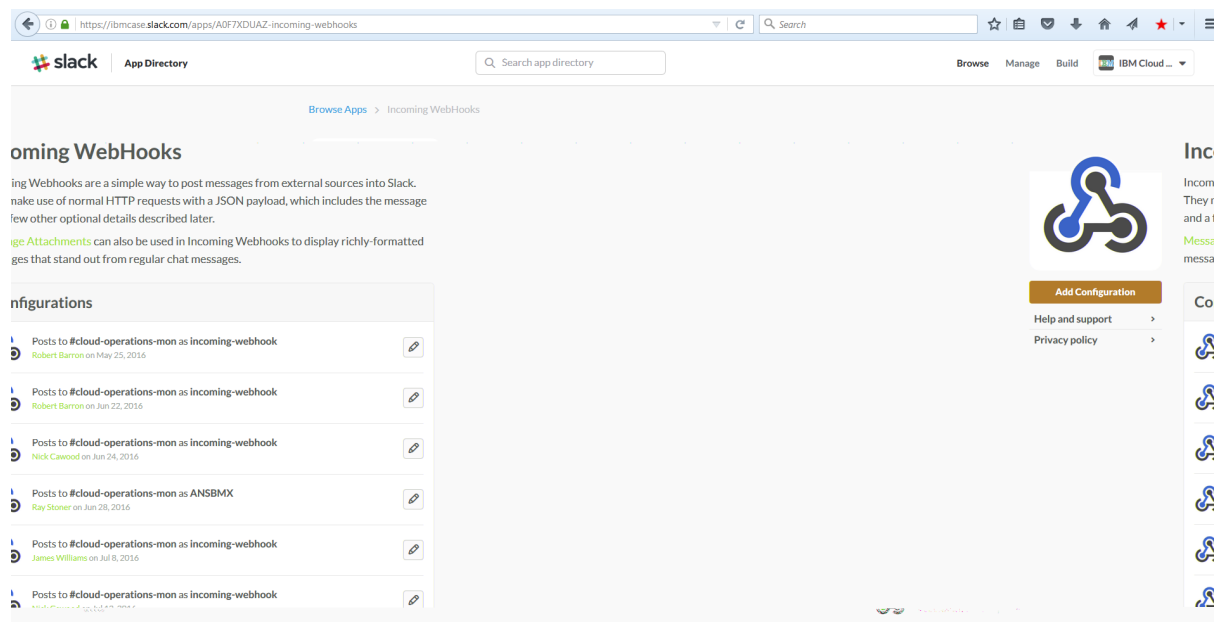
We'll guide you through the steps necessary to configure an Incoming Webhook so you can start sending data to Slack.

close

Webhook URL

<https://hooks.slack.com/services/T0NKYQPG/B1QTQ3H9P/R9ZRlmQUAi1JH5s3Ur5AgS55>

The new WebHook is listed in the App directory:



The generated URL can be used in the Logstash output Slack plugin specifying the Slack channel, the Slack user, and a custom message (using Logstash fields `%{message}` the raw record, `%{data1}` the Bluemix app name etc. from the grok output):

```
if "servicename_alarm_stopped" in [tags] and
"servicename_throttled" not in [tags] {

  slack {

    url => "https://hooks.slack.com/services/T0NKYQPGB/
B1QTQ3H9P/R9ZRWmQUAiJ1H5s3Ur5AgSs5"

    channel => "#cloud-operations-mon"

    username => "@nick_cawood"

    icon_emoji => ":interrobang:"

    #icon_url => [icon url, would be overridden by
icon_emoji - optional]

    format => "ELK alert: Stopped message reported for
%{data1} on %{host}: %{message}"

    #attachments =>

  }

}
```

For icon details, check the Internet for usable values, for example, the [Emoji Cheat Sheet](#) provides a list of emoticons.

In Slack, the alerts look like this:

!? @nick_cawood BOT 11:58 AM
ELK alert: error message reported for ms-catalog-postpyretic-pomiculture on 198.11.234.72: 938 <14>1 2016-07-12T10:58:56.847907+00:00 loggregator
0c677e82-5bfc-4e38-9155-e416249e3ee6 [APP/0] - -
{\"v\":0,\"level\":20,\"name\":\"newrelic\",\"hostname\":\"19n32r6pk9v\",\"pid\":62,\"time\":\"2016-07-12T10:58:56.847Z\",\"msg\":\"Calling metric_data on collector API with:
[\"112457174520792430\",1468321076.845,1468321136.8470001],[{\"name\":\"Memory/Physical
\",[12,732.984375,732.984375,61.08203125,61.08203125,44772.17449951172]},{\"name\":\"Events/wait\"},
[4,0.005529625000000001,0.005529625000000001,0.001364589,0.0013977619999999999,0.000007644941136881]],[{\"name\":\"Supportability/Events
/Customer/Dropped\"},[0,0,0,0,0]],[{\"name\":\"Supportability/Events/Customer/Seen\"},[0,0,0,0,0]],[{\"name\":\"Supportability/Events/Customer/Sent\"},
[0,0,0,0,0]],[{\"name\":\"Supportability/Events/TransactionError/Seen\"},[0,0,0,0,0]],[{\"name\":\"Supportability/Events/TransactionError/Sent\"},
[0,0,0,0,0]]]\",\"component\":\"remote_method_invoke\"}

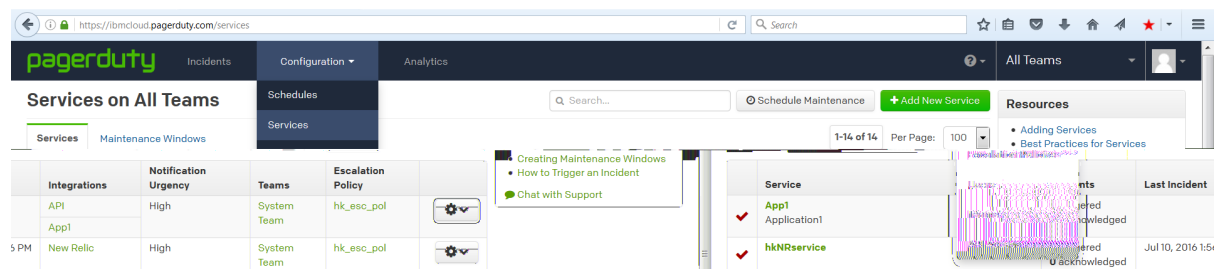
Integrate Logstash with PagerDuty

To send generated alerts to PagerDuty, complete these steps:

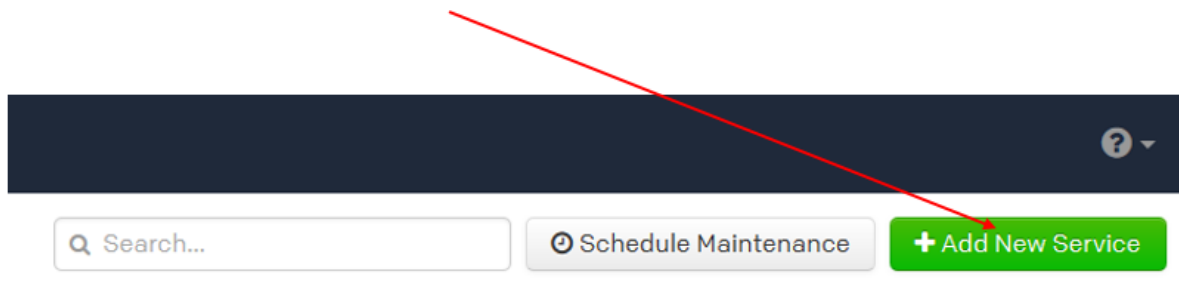
1. Create a PagerDuty integration.
2. Use the [pagerduty plugin](#) in the Logstash output.

To create the PagerDuty integration, follow these steps:

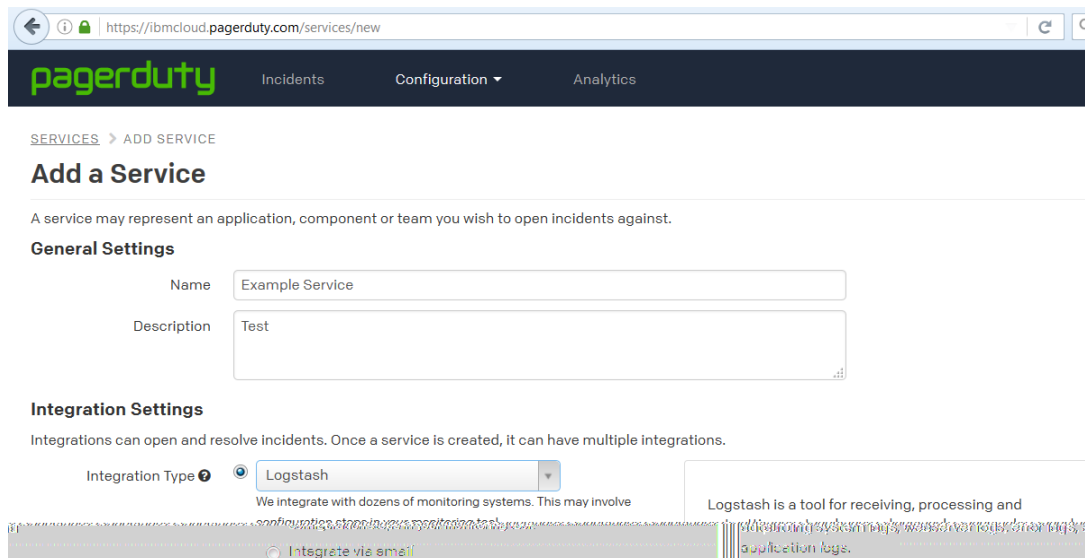
1. Log in to PagerDuty and select **Configuration** → **Services**:



2. Click **Add New Service**:



3. Fill in a name, description, and select **Logstash** for the Integration Type:



https://ibmcloud.pagerduty.com/services/new

pagerduty Incidents Configuration Analytics

SERVICES > ADD SERVICE

Add a Service

A service may represent an application, component or team you wish to open incidents against.

General Settings

Name: Example Service

Description: Test

Integration Settings

Integrations can open and resolve incidents. Once a service is created, it can have multiple integrations.

Integration Type: Logstash

We integrate with dozens of monitoring systems. This may involve configuration steps on your monitoring tool.

Logstash is a tool for receiving, processing and storing application logs.

☐ Integrate via email

4. Add an integration name, select an escalation policy (use existing or create a personalised policy via Configuration → Escalation Policies → New Escalation Policy), select a notification urgency, and tune the incident behaviour for your requirements.

5. Click **Add Service**:

Integration Name: ExampleLogstashIntegration

Incident Settings

Escalation Policy: NickELK

The policy specifies **who will be notified** when this service is triggered.

Notification Urgency: Low - use low-urgency notification rules and do not escalate

Urgency determines how PagerDuty will notify a user of incidents.

Incident Behavior

☒ Incident Acknowledgement Timeout

Acknowledged incidents will time out after 30 minutes, and return to the Triggered state. The incident will then re-notify the person to whom it is assigned.

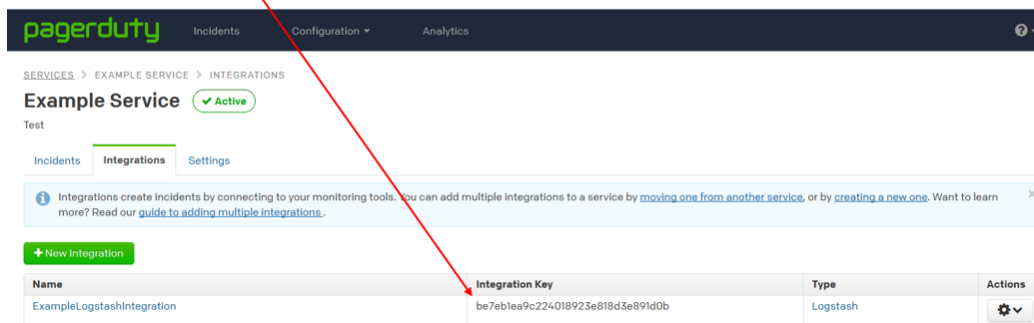
☒ Incident Auto-Resolution

Open incidents will auto-resolve in 4 hours

We recommend turning this option on, to ensure you do not forget incidents in the Triggered or Acknowledged states.

Add Service Cancel

The service is added and an integration key is created to be used in the Logstash configuration:



The generated key can be used in the Logstash output pagerduty plugin using Logstash fields `%{message}` the raw record, `%{data1}` the Bluemix Appname etc. from the grok output to customise the alerts sent to PagerDuty:

```
if "servicename_alarm_stopped" in [tags] and "servicename_throttled"
not in [tags] {

  pagerduty {
    event_type => "trigger"
    description => "%{data1} alert reported on %{host}"
    details => {
      timestamp => "%{@timestamp}"
      message => "%{message}"
    }
    service_key =>
"be7eb1ea9c224018923e818d3e891d0b"
    incident_key => "logstash/%{host}/%{type}"
  }
}
```

Alerts sent from Logstash to Pagerduty appear similar to this one on the PagerDuty Incidents page:

<input type="checkbox"/>	Urgency	#	Created On	Details	Service	Assigned To	Status	
<input type="checkbox"/>	Low	3505	at 6:32 AM	ms-catalog-postpyretic-pomiculture alert reported on 169.55.12.229	ms-catalog-postpyretic-pomiculture	Nick Cawood	Triggered	Details +
<div> <div>Description</div> <div>ms-catalog-postpyretic-pomiculture alert reported on 169.55.12.229</div> <div>Details</div> </div>								
<div> <div>message</div> <div> <pre> 927 <14>1 2016-07-12T13:31:57.240829+00:00 loggregator 0c677e82-5bfc-4e38-9155-e416249e3ee6 [APP/0] - - {"x":0,"level":20,"name":"newsclirc","hostname":"19n32rsgk9","nid":62,"type":"2016-07-12T13:31:57.2397","msg":{"callio metric_data on collector API with: [{"112457174520792430","1468330257.2350001,1468330317.239,[{"name": {"name":"Memory/Physical"},[12,727.078125,727.078125,60.58984375,60.58984375,44053.54998779297],[{"name": /wait"},[4,0.005609202000000001,0.005609202000000001,0.001367914,0.001448952,0.00000786922874618],[{"name":"Supportability/Events/Customer/Dropped"},[0,0,0,0,0,0],[{"name":"Supportability/Events/Cust [0,0,0,0,0,0],[{"name":"Supportability/Events/Customer/Sent"},[0,0,0,0,0,0],[{"name":"Support /TransactionError/Seen"},[0,0,0,0,0,0],[{"name":"Supportability/Events/TransactionError/Sent"}, [0,0,0,0,0,0]]],"component":"remote_method_invoke"} </pre> </div> <div>timestamp</div> <div>2016-07-12T13:31:57.270Z</div> </div>								


```

    msgid => "%{tstamp}"
    procid => "%{data1}"
    rfc => "rfc5424"
    sourcehost => "%{host}"
}

```

For the severity, the pagerduty plugin has the following values:

```

severity string, one of ["emergency", "alert", "critical", "error",
                        "warning", "notice", "informational", "debug"]

```

The Omnibus instance in the scenario's project environment has been set up to receive the severity as follows:

```

if ($Severity <= 3) {
  @Severity = 5
} else if ($Severity = 4) {
  @Severity = 4
} else if ($Severity = 5) {
  @Severity = 3
} else {
  @Severity = 2
}

```

This setup means the Logstash severities are translated into the Omnibus UI as follows:

Logstash severity	Shown in Omnibus as:
alert	Indeterminate
critical	Warning
debug	7
emergency	Clear
error	Minor
informational	6
notice	Critical
warning	Major

Therefore, you should select the Logstash severity based on the value users would expect to see in Omnibus for that specific alert condition.

Once alerts are being sent from Logstash to Omnibus they are shown as follows:

https://159.122.249.116311/bm/console/webtop/eventviewer/eventViewer.jsp?xq=(Agent=Logstash)&transientname=logstashevents&viewname=C											
logstashevents CASE_Integrations 0 171 0 0 0 2											
Group	Count	Sev	Severity	Node	Ack	Location	PagerDuty	Slack	CASE SlackChannel	Summary	LastOccur
All	173			ms-catalog-postpyretic-pomiculture	No		Not Sent	Not Sent		ms-catalog-postpyretic-pomiculture 2016-07-12T14:20:57.467668+00:00 - ("v":0,"level":	7/12/16, 10
Syslog Probe	173			ms-catalog-postpyretic-pomiculture	No		Not Sent	Not Sent		ms-catalog-postpyretic-pomiculture 2016-07-12T14:28:57.438514+00:00 - ("v":0,"level":	7/12/16, 10
				ms-catalog-postpyretic-pomiculture	No		Not Sent	Not Sent		ms-catalog-postpyretic-pomiculture 2016-07-12T14:27:57.434729+00:00 - ("v":0,"level":	7/12/16, 10
				ms-catalog-postpyretic-pomiculture	No		Not Sent	Not Sent		ms-catalog-postpyretic-pomiculture 2016-07-12T14:26:57.431763+00:00 - ("v":0,"level":	7/12/16, 10
				ms-catalog-postpyretic-pomiculture	No		Not Sent	Not Sent		ms-catalog-postpyretic-pomiculture 2016-07-12T14:24:57.357766+00:00 - ("v":0,"level":	7/12/16, 10
				ms-catalog-postpyretic-pomiculture	No		Not Sent	Not Sent		ms-catalog-postpyretic-pomiculture 2016-07-12T14:23:57.357789+00:00 - ("v":0,"level":	7/12/16, 10
				ms-catalog-postpyretic-pomiculture	No		Not Sent	Not Sent		ms-catalog-postpyretic-pomiculture 2016-07-12T14:22:57.356874+00:00 - ("v":0,"level":	7/12/16, 10
				ms-catalog-postpyretic-pomiculture	No		Not Sent	Not Sent		ms-catalog-postpyretic-pomiculture 2016-07-12T14:21:57.355788+00:00 - ("v":0,"level":	7/12/16, 10
				ms-catalog-postpyretic-pomiculture	No		Not Sent	Not Sent		ms-catalog-postpyretic-pomiculture 2016-07-12T14:20:57.453310+00:00 - ("v":0,"level":	7/12/16, 10
				ms-catalog-postpyretic-pomiculture	No		Not Sent	Not Sent		ms-catalog-postpyretic-pomiculture 2016-07-12T14:19:57.452606+00:00 - ("v":0,"level":	7/12/16, 10
				ms-catalog-postpyretic-pomiculture	No		Not Sent	Not Sent		ms-catalog-postpyretic-pomiculture 2016-07-12T14:18:57.446556+00:00 - ("v":0,"level":	7/12/16, 10
				ms-catalog-postpyretic-pomiculture	No		Not Sent	Not Sent		ms-catalog-postpyretic-pomiculture 2016-07-12T14:17:57.418104+00:00 - ("v":0,"level":	7/12/16, 10
				ms-catalog-postpyretic-pomiculture	No		Not Sent	Not Sent		ms-catalog-postpyretic-pomiculture 2016-07-12T14:16:57.656549+00:00 - ("v":0,"level":	7/12/16, 10
				ms-catalog-postpyretic-pomiculture	No		Not Sent	Not Sent		ms-catalog-postpyretic-pomiculture 2016-07-12T14:15:57.432124+00:00 - ("v":0,"level":	7/12/16, 10
Initial 4/2 Selected: 0 3:50 PM 3 row(s) modified and 4 row(s) deleted											



Appendix: Logstash configuration

Here is the full Logstash configuration file used as part of the scenario project ELK instance for reference:

```
input {
    tcp {
        port => 5000
        type => syslog
    }
}

filter {
    mutate {
        add_field => { "format" => "cf" }
    }

    grok {
        match => { "message" => "%{NUMBER:num1} \<%
{NUMBER:num2}\>%{NUMBER:num3} %{TIMESTAMP_ISO8601:tstamp} %
{WORD:CFsource} %{DATA:data1} \[%{DATA:data2}\] %{NOTSPACE:dash1} %
{NOTSPACE:dash2} %{GREEDYDATA:msgtosend}" }

        add_tag => ["grokked"]
    } #end grok

    translate {
        field => "data1"
        destination => "data1"
        override => "true"
        dictionary => [ "13af977a-d2e4-4814-
ab8c-29c9bef95724", "voicemailshow.servicemanagement",
                        "dc50f6e4-c052-436a-
bf27-95ba392c4228", "monverify.servicemanagement",
```

```
"f4c4bc34-856e-4867-9334-3dfd28365a9e", "microservices-catalogapi-
rstoner-1615",
```

```
"059ba88b-0e42-4e29-858a-5ebd5762389c", "Microservices-UI-
rstoner-1639",
```

```
"9b8ceb56-d952-40f7-9d84-
aa06ea4d2dbe", "Microservices-OrdersAPI-rstoner-1622" ]
```

```
    add_tag => "translated"
  }
}
```

```
# === Check Alarm Criteria ===
```

```
# Standard Key Words
```

```
if [message] =~ /.*(err).*/ {
    mutate { add_tag => "servicename_alarm_err" }
}
```

```
if [message] =~ /.*(Error).*/ {
    mutate { add_tag => "servicename_alarm_error" }
}
```

```
if [message] =~ /.*(fatal).*/ {
    mutate { add_tag => "servicename_alarm_fatal" }
}
```

```
if [message] =~ /.*(WARNING).*/ {
    mutate { add_tag => "servicename_alarm_warning" }
}
```

```
if [message] =~ /.*(STOPPED|Stopped|CRASHED).*/ {
    mutate { add_tag => "servicename_alarm_stopped" }
}
```

```
# === Alarm Exclusions ===
```

```
#if [message] =~ /.*(?i)sometextthere.*/ {
#  mutate { remove_tag => "servicename_alarm" }
#}
```

```
# Throttle alarms so we do not hit PD rate limits
```



```
if "servicename_alarm_err" in [tags] {
    throttle {
        after_count => 5
        period => 300
        key => "servicename_alarm_err"
        add_tag => "servicename_throttled"
    }
}

if "servicename_alarm_error" in [tags] {
    throttle {
        after_count => 5
        period => 600
        key => "servicename_alarm_error"
        add_tag => "servicename_throttled"
    }
}

if "servicename_alarm_fatal" in [tags] {
    throttle {
        after_count => 5
        period => 900
        key => "servicename_alarm_fatal"
        add_tag => "servicename_throttled"
    }
}

if "servicename_alarm_warning" in [tags] {
    throttle {
        after_count => 5
        period => 1200
        key => "servicename_alarm_warning"
        add_tag => "servicename_throttled"
    }
}
```

```

    }
    if "servicename_alarm_stopped" in [tags] {
        throttle {
            after_count => 5
            period => 300
            key => "servicename_alarm_stopped"
            add_tag => "servicename_throttled"
        }
    }
}

output {
    elasticsearch {
        hosts => localhost
    }

    stdout { codec => rubydebug }

    if "servicename_alarm_err" in [tags] and
    "servicename_throttled" not in [tags] {
        if [data1] == "voicemailshow.servicemanagement" {
            pagerduty {
                event_type => "trigger"
                description => "%{data1} alert reported on %{host}"
                details => {
                    timestamp => "%{@timestamp}"
                    message => "%{message}"
                }
                service_key => "<pagerduty_service_key>"
                incident_key => "logstash/%{host}/%{type}"
            }
        }

        if [data1] == "monverify.servicemanagement" {

```

```
pagerduty {
    event_type => "trigger"
    description => "%{data1} alert reported on %{host}"
    details => {
        timestamp => "%{@timestamp}"
        message => "%{message}"
    }
    service_key => "<pagerduty_service_key>"
    incident_key => "logstash/%{host}/%{type}"
}

if [data1] == "ms-catalog-postpyretic-pomiculture" {
pagerduty {
    event_type => "trigger"
    description => "%{data1} alert reported on %{host}"
    details => {
        timestamp => "%{@timestamp}"
        message => "%{message}"
    }
    service_key => "<pagerduty_service_key>"
    incident_key => "logstash/%{host}/%{type}"
}
}

if [data1] == "microservices-catalogapi-rstoner-1615" {
pagerduty {
    event_type => "trigger"
    description => "%{data1} alert reported on %{host}"
    details => {
        timestamp => "%{@timestamp}"
        message => "%{message}"
    }
}
```

```

        service_key => "<pagerduty_service_key>"
        incident_key => "logstash/%{host}/%{type}"
    }
}

if [data1] == "Microservices-UI-rstoner-1639" {
pagerduty {
    event_type => "trigger"
    description => "%{data1} alert reported on %{host}"
    details => {
        timestamp => "%{@timestamp}"
        message => "%{message}"
    }
    service_key => "<pagerduty_service_key>"
    incident_key => "logstash/%{host}/%{type}"
}
}

if [data1] == "Microservices-OrdersAPI-rstoner-1622" {
pagerduty {
    event_type => "trigger"
    description => "%{data1} alert reported on %{host}"
    details => {
        timestamp => "%{@timestamp}"
        message => "%{message}"
    }
    service_key => "<pagerduty_service_key>"
    incident_key => "logstash/%{host}/%{type}"
}
}

slack {
    url => "<SlackIntegrationURL>"
    channel => "#slack_channel"
}

```



```

        username => "@slack_user"
        icon_emoji => ":interrobang:"
        #icon_url => [icon url, would be overridden by
icon_emoji - optional]

        format => "ELK alert: err message reported for %
{data1} on %{host}: %{message}"

        #attachments =>

    }

    http {
        url => "http://<OmnibusIP>:<OmnibusPort>"
        http_method => "post"
        format => "json"
    }

    syslog {
        facility => "log alert"
        host => "<OmnibusIP>"
        port => "<OmnibusPort>"
        severity => "warning"
        appname => "logstash"
        message => "%{msgtosend}"
        msgid => "%{tstamp}"
        procid => "%{data1}"
        rfc => "rfc5424"
        sourcehost => "%{host}"
    }
}

if "servicename_alarm_error" in [tags] and
"servicename_throttled" not in [tags] {
    if [data1] == "voicemailshow.servicemanagement" {
        pagerduty {
            event_type => "trigger"
            description => "%{data1} alert reported on %{host}"

```



```
details => {
    timestamp => "%{@timestamp}"
    message => "%{message}"
}
service_key => "<pagerduty_service_key>"
incident_key => "logstash/%{host}/%{type}"
}
}

if [data1] == "monverify.servicemanagement" {
pagerduty {
    event_type => "trigger"
    description => "%{data1} alert reported on %{host}"
    details => {
        timestamp => "%{@timestamp}"
        message => "%{message}"
    }
    service_key => "<pagerduty_service_key>"
    incident_key => "logstash/%{host}/%{type}"
}
}

if [data1] == "ms-catalog-postpyretic-pomiculture" {
pagerduty {
    event_type => "trigger"
    description => "%{data1} alert reported on %{host}"
    details => {
        timestamp => "%{@timestamp}"
        message => "%{message}"
    }
    service_key => "<pagerduty_service_key>"
    incident_key => "logstash/%{host}/%{type}"
}
}
```



```
    }
    if [data1] == "microservices-catalogapi-rstoner-1615" {
    pagerduty {
        event_type => "trigger"
        description => "%{data1} alert reported on %{host}"
        details => {
            timestamp => "%{@timestamp}"
            message => "%{message}"
        }
        service_key => "<pagerduty_service_key>"
        incident_key => "logstash/%{host}/%{type}"
    }
    }
    if [data1] == "Microservices-UI-rstoner-1639" {
    pagerduty {
        event_type => "trigger"
        description => "%{data1} alert reported on %{host}"
        details => {
            timestamp => "%{@timestamp}"
            message => "%{message}"
        }
        service_key => "<pagerduty_service_key>"
        incident_key => "logstash/%{host}/%{type}"
    }
    }
    if [data1] == "Microservices-OrdersAPI-rstoner-1622" {
    pagerduty {
        event_type => "trigger"
        description => "%{data1} alert reported on %{host}"
        details => {
            timestamp => "%{@timestamp}"
```

```

        message => "%{message}"
    }
    service_key => "<pagerduty_service_key>"
    incident_key => "logstash/%{host}/%{type}"
}
}
slack {
    url => "<SlackIntegrationURL>"
    channel => "#slack_channel"
    username => "@slack_user"
    icon_emoji => ":interrobang:"
    #icon_url => [icon url, would be overridden by
icon_emoji - optional]
    format => "ELK alert: error message reported for %{
{data1} on %{host}: %{message}"
    #attachments =>
}
http {
    url => "http://<OmnibusIP>:<OmnibusPort>"
    http_method => "post"
    format => "json"
}
syslog {
    facility => "log alert"
    host => "<OmnibusIP>"
    port => "<OmnibusPort>"
    severity => "warning"
    appname => "logstash"
    message => "%{msgtosend}"
    msgid => "%{tstamp}"
    procid => "%{data1}"
}

```



```
        rfc => "rfc5424"
        sourcehost => "%{host}"
    }
}

    if "servicename_alarm_fatal" in [tags] and
"servicename_throttled" not in [tags] {
        if [data1] == "voicemailshow.servicemanagement" {
            pagerduty {
                event_type => "trigger"
                description => "%{data1} alert reported on %{host}"
                details => {
                    timestamp => "%{@timestamp}"
                    message => "%{message}"
                }
                service_key => "<pagerduty_service_key>"
                incident_key => "logstash/%{host}/%{type}"
            }
        }

        if [data1] == "monverify.servicemanagement" {
            pagerduty {
                event_type => "trigger"
                description => "%{data1} alert reported on %{host}"
                details => {
                    timestamp => "%{@timestamp}"
                    message => "%{message}"
                }
                service_key => "<pagerduty_service_key>"
                incident_key => "logstash/%{host}/%{type}"
            }
        }

        if [data1] == "ms-catalog-postpyretic-pomiculture" {
```



```
pagerduty {
    event_type => "trigger"
    description => "%{data1} alert reported on %{host}"
    details => {
        timestamp => "%{@timestamp}"
        message => "%{message}"
    }
    service_key => "<pagerduty_service_key>"
    incident_key => "logstash/%{host}/%{type}"
}

if [data1] == "microservices-catalogapi-rstoner-1615" {
pagerduty {
    event_type => "trigger"
    description => "%{data1} alert reported on %{host}"
    details => {
        timestamp => "%{@timestamp}"
        message => "%{message}"
    }
    service_key => "<pagerduty_service_key>"
    incident_key => "logstash/%{host}/%{type}"
}

if [data1] == "Microservices-UI-rstoner-1639" {
pagerduty {
    event_type => "trigger"
    description => "%{data1} alert reported on %{host}"
    details => {
        timestamp => "%{@timestamp}"
        message => "%{message}"
    }
}
```

```

        service_key => "<pagerduty_service_key>"
        incident_key => "logstash/%{host}/%{type}"
    }
}

if [data1] == "Microservices-OrdersAPI-rstoner-1622" {
pagerduty {
    event_type => "trigger"
    description => "%{data1} alert reported on %{host}"
    details => {
        timestamp => "%{@timestamp}"
        message => "%{message}"
    }
    service_key => "<pagerduty_service_key>"
    incident_key => "logstash/%{host}/%{type}"
}
}

slack {
    url => "<SlackIntegrationURL>"
    channel => "#slack_channel"
    username => "@slack_user"
    icon_emoji => ":x:"
    #icon_url => [icon url, would be overridden by
icon_emoji - optional]

    format => "ELK alert: fatal message reported for %
{data1} on %{host}: %{message}"
    #attachments =>
}

http {
    url => "http://<OmnibusIP>:<OmnibusPort>"
    http_method => "post"
    format => "json"
}

```



```
}
syslog {
    facility => "log alert"
    host => "<OmnibusIP>"
    port => "<OmnibusPort>"
    severity => "notice"
    appname => "logstash"
    message => "%{msgtosend}"
    msgid => "%{tstamp}"
    procid => "%{data1}"
    rfc => "rfc5424"
    sourcehost => "%{host}"
}
}

if "servicename_alarm_warning" in [tags] and
"servicename_throttled" not in [tags] {
    if [data1] == "voicemailshow.servicemanagement" {
        pagerduty {
            event_type => "trigger"
            description => "%{data1} alert reported on %{host}"
            details => {
                timestamp => "%{@timestamp}"
                message => "%{message}"
            }
            service_key => "<pagerduty_service_key>"
            incident_key => "logstash/%{host}/%{type}"
        }
    }

    if [data1] == "monverify.servicemanagement" {
        pagerduty {
            event_type => "trigger"
```




```
description => "%{data1} alert reported on %{host}"
details => {
    timestamp => "%{@timestamp}"
    message => "%{message}"
}
service_key => "<pagerduty_service_key>"
incident_key => "logstash/%{host}/%{type}"
}
}

if [data1] == "ms-catalog-postpyretic-pomiculture" {
pagerduty {
    event_type => "trigger"
    description => "%{data1} alert reported on %{host}"
    details => {
        timestamp => "%{@timestamp}"
        message => "%{message}"
    }
    service_key => "<pagerduty_service_key>"
    incident_key => "logstash/%{host}/%{type}"
}
}

if [data1] == "microservices-catalogapi-rstoner-1615" {
pagerduty {
    event_type => "trigger"
    description => "%{data1} alert reported on %{host}"
    details => {
        timestamp => "%{@timestamp}"
        message => "%{message}"
    }
    service_key => "<pagerduty_service_key>"
    incident_key => "logstash/%{host}/%{type}"
}
```



```
    }
  }

  if [data1] == "Microservices-UI-rstoner-1639" {
    pagerduty {
      event_type => "trigger"
      description => "%{data1} alert reported on %{host}"
      details => {
        timestamp => "%{@timestamp}"
        message => "%{message}"
      }
      service_key => "<pagerduty_service_key>"
      incident_key => "logstash/%{host}/%{type}"
    }
  }

  if [data1] == "Microservices-OrdersAPI-rstoner-1622" {
    pagerduty {
      event_type => "trigger"
      description => "%{data1} alert reported on %{host}"
      details => {
        timestamp => "%{@timestamp}"
        message => "%{message}"
      }
      service_key => "<pagerduty_service_key>"
      incident_key => "logstash/%{host}/%{type}"
    }
  }

  slack {
    url => "<SlackIntegrationURL>"
    channel => "#slack_channel"
    username => "@slack_user"
    icon_emoji => ":heavy_exclamation_mark:"
  }
}
```



```
        #icon_url => [icon url, would be overridden by
icon_emoji - optional]

        format => "ELK alert: WARNING message reported for
%{data1} on %{host}: %{message}"

        #attachments =>

    }

    http {
        url => "http://<OmnibusIP>:<OmnibusPort>"
        http_method => "post"
        format => "json"
    }

    syslog {
        facility => "log alert"
        host => "<OmnibusIP>"
        port => "<OmnibusPort>"
        severity => "critical"
        appname => "logstash"
        message => "%{msgtosend}"
        msgid => "%{tstamp}"
        procid => "%{data1}"
        rfc => "rfc5424"
        sourcehost => "%{host}"
    }
}

    if "servicename_alarm_stopped" in [tags] and
"servicename_throttled" not in [tags] {
        if [data1] == "voicemailshow.servicemanagement" {
            pagerduty {
                event_type => "trigger"
                description => "%{data1} alert reported on %{host}"
                details => {
                    timestamp => "%{@timestamp}"
                }
            }
        }
    }
}
```

```

        message => "%{message}"
    }
    service_key => "<pagerduty_service_key>"
    incident_key => "logstash/%{host}/%{type}"
}
}

if [data1] == "monverify.servicemanagement" {
pagerduty {
    event_type => "trigger"
    description => "%{data1} alert reported on %{host}"
    details => {
        timestamp => "%{@timestamp}"
        message => "%{message}"
    }
    service_key => "<pagerduty_service_key>"
    incident_key => "logstash/%{host}/%{type}"
}
}

if [data1] == "ms-catalog-postpyretic-pomiculture" {
pagerduty {
    event_type => "trigger"
    description => "%{data1} alert reported on %{host}"
    details => {
        timestamp => "%{@timestamp}"
        message => "%{message}"
    }
    service_key => "<pagerduty_service_key>"
    incident_key => "logstash/%{host}/%{type}"
}
}

if [data1] == "microservices-catalogapi-rstoner-1615" {

```



```
pagerduty {
    event_type => "trigger"
    description => "%{data1} alert reported on %{host}"
    details => {
        timestamp => "%{@timestamp}"
        message => "%{message}"
    }
    service_key => "<pagerduty_service_key>"
    incident_key => "logstash/%{host}/%{type}"
}

if [data1] == "Microservices-UI-rstoner-1639" {
pagerduty {
    event_type => "trigger"
    description => "%{data1} alert reported on %{host}"
    details => {
        timestamp => "%{@timestamp}"
        message => "%{message}"
    }
    service_key => "<pagerduty_service_key>"
    incident_key => "logstash/%{host}/%{type}"
}

if [data1] == "Microservices-OrdersAPI-rstoner-1622" {
pagerduty {
    event_type => "trigger"
    description => "%{data1} alert reported on %{host}"
    details => {
        timestamp => "%{@timestamp}"
        message => "%{message}"
    }
}
```

```

        service_key => "<pagerduty_service_key>"
        incident_key => "logstash/%{host}/%{type}"
    }
}

slack {
    url => "<SlackIntegrationURL>"
    channel => "#slack_channel"
    username => "@slack_user"
    icon_emoji => ":interrobang:"
    #icon_url => [icon url, would be overridden by
icon_emoji - optional]

    format => "ELK alert: Stopped message reported for
%{data1} on %{host}: %{message}"

    #attachments =>
}

http {
    url => "http://<OmnibusIP>:<OmnibusPort>"
    http_method => "post"
    format => "json"
}

syslog {
    facility => "log alert"
    host => "<OmnibusIP>"
    port => "<OmnibusPort>"
    severity => "notice"
    appname => "logstash"
    message => "%{msgtosend}"
    msgid => "%{tstamp}"
    procid => "%{data1}"
    rfc => "rfc5424"
    sourcehost => "%{host}"
}

```



}
}
}