Parsing Logs via ELK

BSides Austin 2019

Presenter: Mark McLauchlin

Github: https://github/macatak/ELK/

Overview

Goal – Learn how to get information out of the data in logs

How we'll do that
•Use the open source ELK stack to ingest and enrich logs

What we'll cover
• Setup and configuration of an open source ELK stack running on the localhost

What we won't cover
• Any of the paid features of ELK

Who Am I?

- Systems Engineer at a small automobile manufacturing IT start up

Degrees and security certifications that have nothing to do with this talk

- MS-IT focusing on InfoSec
- CEH / ECSA

Who are you?

- Name
- Little bit about yourself
- What you hope to get out of this training

Story of the 3 Little pigs

Once upon a time there were 3 little pigs
The 1st little pig did not use any centralized logging like ELK. One day the marketing guy called and said the internet was down. The 1st little pig logged on many different servers trying to find the issue. He knew what to look for but didn't know exactly where. By the time he found it his company was out of of business and the big bad wolf ate him on his way to the unemployment office.

Story of the 3 Little pigs

The 2nd little pig did use centralized logging. He did as little possible to get the logs into ELK. One day the marketing guy called and said the internet was down. The 2nd little pig looked and looked in Kibana to find the problem. However, he has too many logs coming in and was having a hard time finding the problem. He knew what to look for but had too much data that was unrelated. By the time he found it his company was out of of business and the big bad wolf ate him on his way to the unemployment office.

Story of the 3 Little pigs

The 3rd little pig did use ELK for centralized logging. He put forth a little effort to ensure logs came in and were parsed in a uniform manner. One day the marketing guy called and said the internet was down. The 3rd little pig already knew there was problem because he was it on his awesome Kibana dashboard and already let operation know there was a problem. The marketing guy was impressed and bought him lunch.

Morale of the story? Put a little effort into your log parsing and avoid the unemployment line where you might get eaten by the big bad wolf

Tell a Story

Licensing

What We Will Cover

- ELK Components
  - Overview
  - Installation
  - Configuration
- Deeper Dives
  - Filebeat
  - Logstash
  - Kibana

What is the ELK Stack?

Base Stack

- ***E***lasticsearch - NoSQL database based on the Lucene search engine
- ***L***ogstash - Log pipeline app that accepts event inputs and can enrich the data
- **K**ibana - Visualization layer and search

The rest of the players

- Filebeat – Lightweight log shipper, can also enrich event data
- Other beat packages that will not be covered.

Event Data Flow Model (simplified)

1) Filebeat lives on the server generating the event data and sends it to Logstash
2) Logstash takes the Filebeat input, parses it, and sends it to Elasticsearch
3) Elasticsearch stores the data
4) Kibana retrieves the data out of Elasticsearch

ELK Stack 101’s

Filebeat 101

What it does
- Monitors one or more logs
- Can enrich event data (add tags or key:value pairs)
- Forwards to Logstash
  - Can also send straight to Elasticsearch along with a few other apps

How it does it
- Sort of ‘tails’ a file, not real-time
- Keeps a pointer in a file to keep track of last read

Why we use it
- It is easily configurable
- It not resource intensive
- It works great

Logstash 101

What it does

- Wide variety of support for many input and output technologies
- Capability to parse and enrich data

How it does it

- I/O supported through plugins
- Parsing/enrichment through filters

Why we use it

- Power of slicing and dicing data
- Event enrichment

Elasticsearch 101

What it does
- Stores data data as an JSON document

How it does it
- NoSQL Lucene database (same as Apache Solr)

Why we use it
- Very fast searches
- API support (don’t need Logstash or Kibana)
- Horizontally scalable

Kibana 101

What it does
- Provides easy access to data in Elasticsearch
- Can do basic reporting, dashboards, and more

How it does it
- Web UI (Node JS)

Why we use it
- Search capabilities
- Dashboards

Basic VM Setup Info

1) Create a folder for sample log files
   example - /home/<username>/log_samples
   This is where we'll put our sample logs
2) Most ELK installs will place the YAML files here:
   1) /etc/<app_name>
   2) Logstash has a YML file but we won't use it
3) Make a backup of the Elasticsearch and Kibana YAML's
   1) /etc/elasticsearch.yml
   2) /etc/kibana.yml
4) Create an /etc/logstash/conf_files folder
   This is where we'll put our Logstash parsers

Lay the Foundation

Downloads

MISC

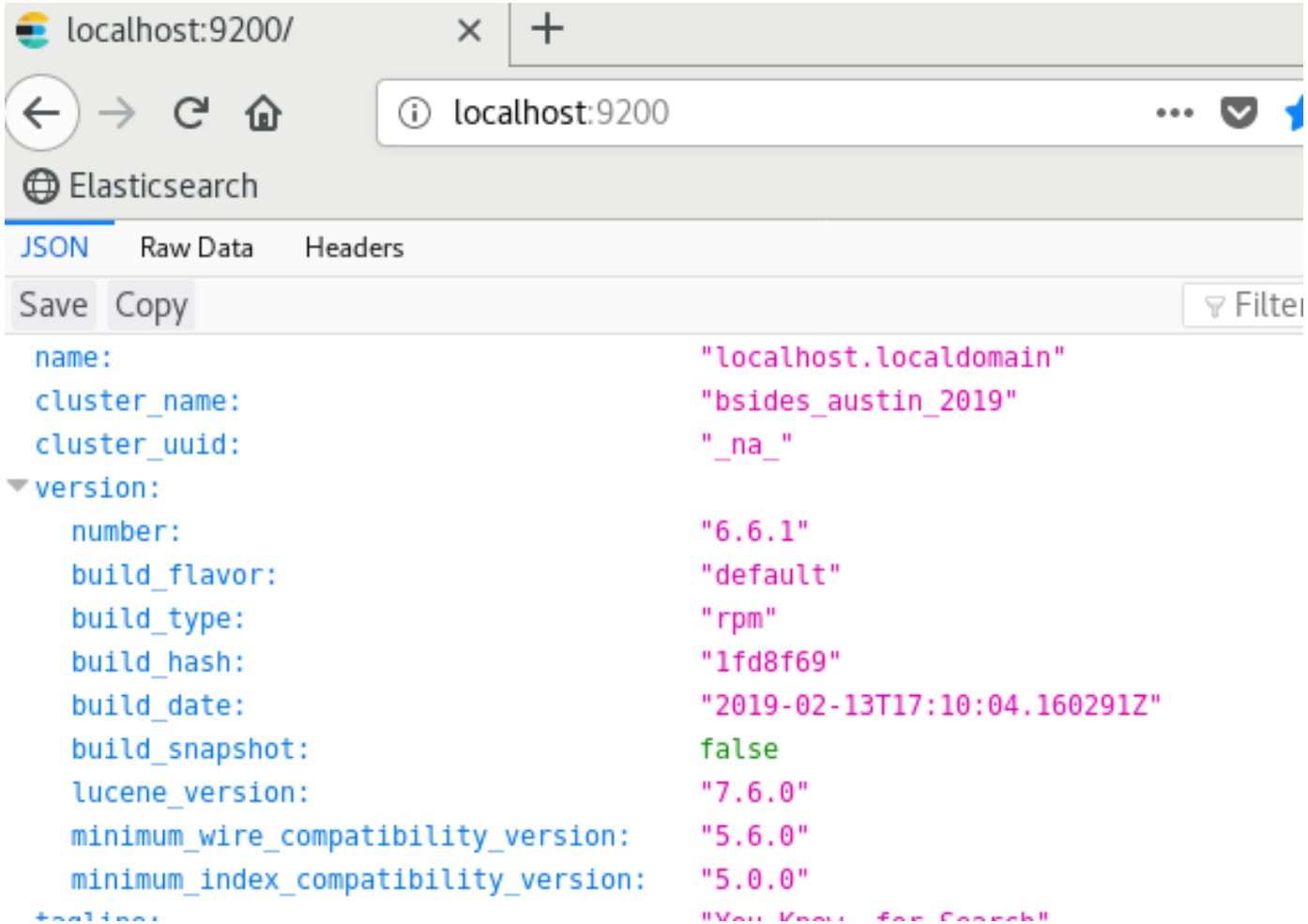
- Copy the GitHub repo
  - https://github.com/macatak/ELK/tree/master/AustinBSides2019
- Install RPM/DEB package, version 6.6.1
- https://www.elastic.co/downloads/past-releases
  - Filebeat
  - Logstash
  - Elasticsearch
  - Kibana

Avoid using .0 (x.0 or x.x.0) releases, quality is sometimes lacking

Elasticsearch Setup

Exercise 1: Elasticsearch Install

- Follow the steps in the /exercise/ex1.txt to install Elasticsearch
- Validate it is running



elasticsearch.yml file

cluster.name: bsides_austin_2019

You can name this whatever you want

node.name: ${HOSTNAME}

This will use the host name as the node name

path.data: /var/lib/elasticsearch

Where the data will be stored

path.logs: /var/log/elasticsearch

Log file location

network.host: localhost

Required for localhost setup

http.port: 9200

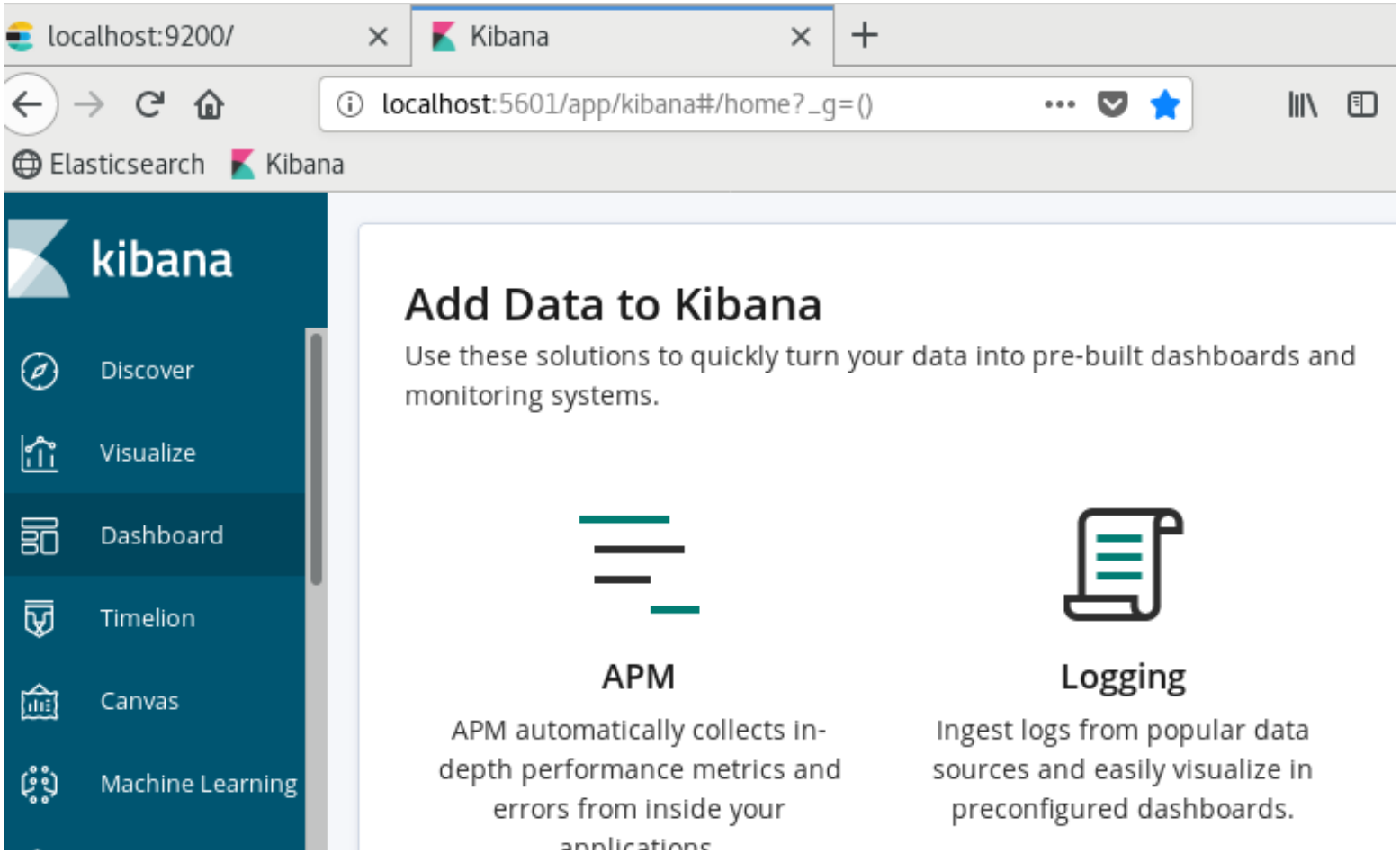
Port to be used by Elasticsearch


Elasticsearch Questions?

Kibana Setup

Exercise 2: Kibana Install

1) Follow the steps in the /exercise/ex2.txt to install Kibana
2) Validate it is running



Kibana.yml File

server.port: 5601

Port where Kibana can be accessed

server.host: "localhost"

Setting for localhost setup

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

Where Elasticsearch is running

These are not required but can clean up the interface

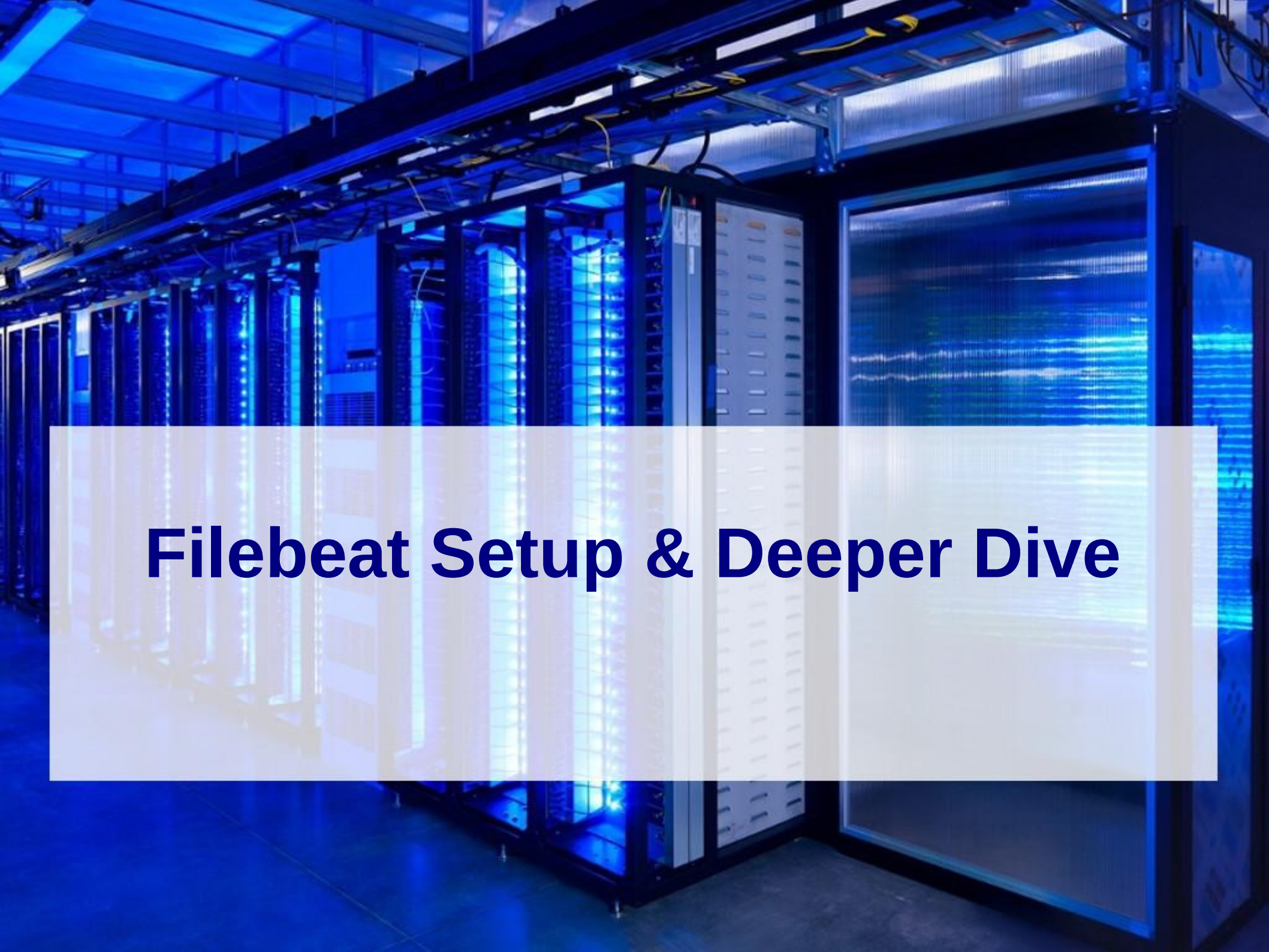
#xpack.apm.ui.enabled: false

#xpack.apm.enabled: false

#xpack.ml.enabled: false

#xpack.graph.enabled: false


Kibana Questions?



Filebeat Setup & Deeper Dive

Filebeat YAML Settings

Things you can set in the YAML (but will not covered)

- Include/Exclude RegEx’s
  - Very useful in the real world
- Exclude Files
- Multiline Parsing
  - Also very useful in the real world
- Modules
- Reload Period
- Dashboard Imports
  - Nice to have but only if you need them
- SSL/TLS Implementation
- Processors
- Xpack

Exercise 3: Filebeat Setup

1) Follow the steps in the /exercise/ex3.txt to install filebeat
2) Make sure you change the path in the yml to the valid path on your VM
   /home/bikeride/sample_logs/access_5.log

There is a filebeat.reference.yml with all available settings (there are a LOT of them)

Check the filebeat.yml File

Syntax check on the configuration

- Command : /usr/share/filebeat/bin/filebeat test config -c filebeat.yml
  NOTE: '-c filebeat.yml' is not really really required
- Assumes current working directory is /etc/filebeat/

```
[root@localhost filebeat]# /usr/share/filebeat/bin/filebeat test config -c filebeat.yml
Config OK
[root@localhost filebeat]#
```

- Issues
  - Syntax only, will not catch things like mangled or bad paths
  - Indication of these types of errors is no output

Run Filebeat (Command Line)

Normally Filebeat would be run as a service but for development it can be run from the command line with the output directed to the console.

- Command

  /usr/share/filebeat/bin/filebeat -c filebeat.yml

  - CTRL+C to stop


```
{
  "@timestamp": "2019-02-24T18:05:14.561Z",
  "@metadata": {
    "beat": "filebeat",
    "type": "doc",
    "version": "6.5.4"
  },
  "host": {
    "name": "localhost.localdomain"
  },
  "source": "/home/bikeride/log_samples/access_5.log",
  "offset": 241,
  "message": "[08/Jul/2017:17:36:57 -0400] \"GET /manager/html HTTP/1.1\" 403 3420",
  "prospector": {
    "type": "log"
  },
  "input": {
    "type": "log"
  },
  "beat": {
    "name": "localhost.localdomain",
    "hostname": "localhost.localdomain",
    "version": "6.5.4"
  }
}
```

Run Filebeat (Command Line)

Run the command again

What happens?

Why?

What can we do to fix it?

Run Filebeat (Command Line)

The Issue

- Filebeat keeps track of where it last read a file
- We need to ‘clear’ the memory
  - Need to delete the ‘registry’ file
- Normally one of two places, depending on how Filebeat was run
  - As a service
    /var/lib/filebeat/registry
  - From the command line
    /usr/share/filebeat/bin/data/registry
- If in doubt search for the file
  - find / -name registry | grep filebeat

Filebeat Help

There is a wealth of information online but some items of note:

- Debugging a YML
  - https://www.elastic.co/guide/en/beats/filebeat/current/enable-filebeat-debugging.html
  - Command: filebeat -e -d "publish"
    - -e is an override command
    - -d is debug mode
    - "publish" displays all the "publish" related messages.

- YAML tips and gotchas
  - https://www.elastic.co/guide/en/beats/filebeat/current/yaml-tips.html

Why is this Important?

- The event enrichment is very powerful
  - You can use the fields and tags set in the Filebeat YML in your Logstash filters
    - Easier parsing rules
    - Set destination index
- It is an easy configuration and low maintenance


Filebeat Questions?


Logstash Deeper Dive

Logstash Inputs

- Inputs are based on plugins
- An input plugin enables a specific source of events to be read by Logstash.
  - ~around 50
  - https://www.elastic.co/guide/en/logstash/current/input-plugins.html
- Plugins of note
  - Beats
    - Supports the beat packages (like Filebeat)
  - File
    - Supports reading from a file
  - Syslog
    - Supports syslog input
  - Generator
    - Generates log events

Logstash Filters

- Filter plugin performs processing on an event.
  - https://www.elastic.co/guide/en/logstash/current/filter-plugins.html
- Can add, transform, or remove event data
  - This is where the power of Logstash to slice and dice data comes from
- Filter plugins of note:
  - Dissect - Extracts unstructured event data into fields using delimiters or what separates the values
  - Grok - Parses unstructured event data into fields using RegEx
  - Geo - Adds geographical information
  - Mutate - Performs mutations on fields

Logstash Outputs

- Plugin oriented (like Inputs)
  - https://www.elastic.co/guide/en/logstash/current/output-plugins.html
- Logstash can send data to 1 or more outputs
- Output plugins of note:
  - Elasticsearch – Sends data to Elasticsearch
  - File – Writes to a file
  - Stdout – Writes to the console

Logstash Codecs

- Codec plugins change the data representation of an event
  - https://www.elastic.co/guide/en/logstash/current/codec-plugins.html
- Basically stream filters that can operate as part of an input or output
- Codecs of note:
  - Json - Reads JSON formatted content, creating one event per element in a JSON array
  - Json_lines - Reads newline-delimited JSON
  - Multiline - Merges multiline messages into a single event
  - Nmap - Reads Nmap data in XML format
  - Rubydebug – Prints out using the Ruby Awesome print library

Notes on Conf Files

- Logstash automatically reads any and all conf files in the (default) /etc/logstash/conf.d folder
- Files are read in alphabetical order
- If there are numerous conf files Logstash will build a configuration out of them
- For what we are doing we will run Logstash from the command line

For multiple conf files use a number naming convention so inputs are read 1st, followed by filters, then outputs

Why is this Important?

- Logstash is where you can really organize and parse the incoming event data
- The event enrichment is very powerful
- The plugins are vast and varied.
- There are community plugins available that can be installed
- Bottom Line – The more work you do here the more benfits you will get out of thre data sent to Elasticsearch.

Logstash Filebeat Hands-On

Overview: logstash_filebeat_basic.conf

```
input {
  beats {
    port => 5044
  }
}

filter {
   # filtering goes here
}

output {
   stdout { }
   #elasticsearch { hosts => ["localhost:9200"] }
}
```

Filebeat Conf File Syntax Check

1) Check the syntax of the logstash_filebeat_basic.conf file.
   1)‘/usr/share/logstash/bin/logstash -f logstash_filebeat_base.conf -t’
   2) The ‘-t’ is for test
   3) The ‘-f’ tells Logstash to use a specific file
2) Output is verbose but near the bottom you should see “Configuration OK”

<snipped>
Ignoring the 'pipelines.yml' file because modules or command line options are specified
Configuration OK
[INFO] 2019-03-07 13:45:01.117 [LogStash::Runner] runner - Using config.test_and_exit mode. ***Config Validation Result: OK***. Exiting Logstash

3) This just a syntax check so an “OK” does not mean it will work the way you expect (but is a very good practice to get into doing)

Logstash Filebeat Run

1) Start Logstash from the command line
   1) /usr/share/logstash/bin/logstash -f logstash_filebeat_base.conf

```
{
        "source" => "/home/bikeride/sample_logs/access_5.log",
    "@timestamp" => 2019-03-07T19:55:41.697Z,
         "input" => {
        "type" => "log"
    },
          "tags" => [
        [0] "beats_input_codec_plain_applied"
    ],
          "host" => {
        "name" => "localhost.localdomain"
    },
    "prospector" => {
        "type" => "log"
    },
        "offset" => 63,
      "@version" => "1",
        "fields" => {
        "target_index" => "test"
    },
       "message" => "[08/Jul/2017:17:36:51 -0400] \"GET /bodgeit HTTP/1.1\" 302 -",
          "beat" => {
         "version" => "6.5.4",
        "hostname" => "localhost.localdomain",
            "name" => "localhost.localdomain"
    }
}
```

Why is this important

- In the real world a lot of feeds will come from Filebeat.
- We will see later on how Logstash can use the enrichment Filebeat provides in very useful ways
- Having said that, we will detour from the Filebeat → Logstash model


Logstash Questions?


Logstash File Hands-On

Logstash and Reading from Files

- Open logstash_file_base.conf file in a text editor
- Change the path => setting to match the path on your VM
  - /home/bikeride/sample_logs/access_5.log

Overview: logstash_file_basic.conf

```
input {
  file {
    #type => "log"
    # this is the path to the log file
    path => "/home/bikeride/sample_logs/access_5.log"
    # following 2 lines will configure Logstash to always read from the start
    # this is not something you would do in the real world
    # but very useful for development
    start_position => "beginning"
    sincedb_path => "/dev/null"
  }
}

filter {
  # filtering goes here
}

output {
  stdout { }
}
```

Logstash File Conf File Syntax Check

1) Check the syntax of the logstash_file_basic.conf file.
   1)‘/usr/share/logstash/bin/logstash -f logstash_file_base.conf -t’
   2) The ‘-t’ is for test
   3) The ‘-f’ tells Logstash to use a specific file
2) Output is verbose but near the bottom you should see “Configuration OK”

<snipped>
Ignoring the 'pipelines.yml' file because modules or command line options are specified
Configuration OK
[INFO] 2019-03-07 13:45:01.117 [LogStash::Runner] runner - Using config.test_and_exit mode. ***Config Validation Result: OK***. Exiting Logstash

3) This just a syntax check but is a very good practice to get into doing (Yes, I am purposely repeating that).

Logstash File Basic Run (stdout)

1) Start Logstash from the command line

/usr/share/logstash/bin/logstash -f logstash_file_base.conf

```
{
       "message" => "[08/Jul/2017:17:36:51 -0400] \"GET /bodgeit/ HTTP/1.1\" 404 1078",
          "path" => "/home/bikeride/sample_logs/access_5.log",
    "@timestamp" => 2019-03-07T19:07:53.610Z,
      "@version" => "1",
          "host" => "localhost.localdomain"
}
{
       "message" => "[08/Jul/2017:17:36:55 -0400] \"GET / HTTP/1.1\" 200 11250",
          "path" => "/home/bikeride/sample_logs/access_5.log",
    "@timestamp" => 2019-03-07T19:07:53.610Z,
      "@version" => "1",
          "host" => "localhost.localdomain"
}
{
       "message" => "[08/Jul/2017:17:36:57 -0400] \"GET /manager/html HTTP/1.1\" 403 3420",
          "path" => "/home/bikeride/sample_logs/access_5.log",
    "@timestamp" => 2019-03-07T19:07:53.610Z,
      "@version" => "1",
          "host" => "localhost.localdomain"
}
```

Logstash File Basic Run (Elasticsearch)

1) Open logstash_file_base.conf
2) Uncomment the Elasticsearch output
3) Save the conf file
4) Syntax check?
5) Start Logstash
   1)/usr/share/logstash/bin/logstash -f logstash_file_base.conf
6) Should see the same output
7) Let’s see what’s in Elasticsearch via Kibana

Validation in Kibana

1) Open Kibana in a browser
2) Open the Discover tab (far left)
3) Why is there no data?
4) Create the index pattern
   1) Use logstash*
   2) Set the timestamp to @timestamp
   3) Click “Create Index Pattern”
5) Go back to the Discover tab
6) The events from the log are displayed

Why is this important?

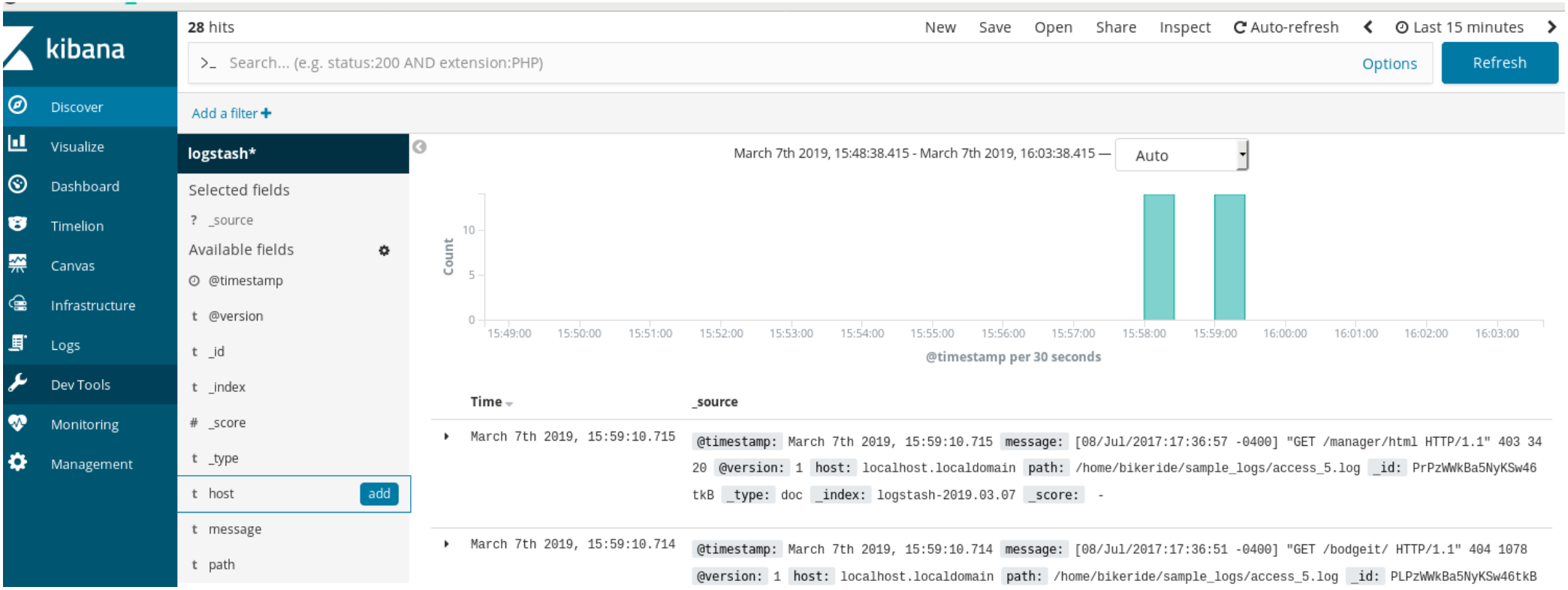
- Logstash is where ‘the rubber meets the road” when it comes to parsing
- Conf file development is easier to accomplish and less error prone against a sample log file
- The sample log file just needs enough lines to be accurate
- Filebeat is useful but a hindrance when it comes to filter development due to the registry


Questions?


Kibana Overview

Kibana Overview

1) Open a browser
2) Go to localhost:5601



Kibana Short Tour

- The Time Filter lets you set a time range to search over.
  - Quick – Select from various predefined ranges
  - Relative – Select relative periods based on time ranges
  - Absolute – Define a start and end time range
  - Recent – A recently used time range
- Auto Refresh
  - Sets the Discover page to auto refresh at specific intervals
- Search Bar
  - Allows searches based on the Apache Lucene Query Parser Syntax
  - Supports logical operator (AND/OR/NOT)
  - Can save, open, and share searches

Kibana – Short Tour

- Fields List
  - Shows available fields
  - Can set which fields to display
  - Provides specific information on the fields
  - Can add filters on field values to include or exclude
- Document Data
  - Shows a graph of events over time.
  - Can drag over to drill down
- We will circle back to this once we some meaningful data

Why is this important?

- Kibana is not hard but not intuitive.
- Knowing the different ways to drill down on data will be very useful.
  - Time
  - Positive/negative filters
  - Many more (as we'll see)


Kibana Questions?

Logstash Filters

Logstash Filter Development Process

1) Look at the event log and determine what needs to be parsed (key:values)
   1) This will probably be an iterative process
2) Start simple
   1) Get input and output section right
   2) Once these are done start with a simple filter
3) Use console out for validation
   1) This does increast the time to parse large logs as we'll see,
   2) There is also an codec => dots output that you may like better
4) Keep in mind that there may be multiple conf files

Lets get started

Logstash Parsing – Adding data

1) Let’s parse the UofS log
2) Open u_of_s.conf in a text editor
3) Uncomment whichever log you want
4) Run Logstash in the console
5) What happened?
6) Look at the data in Kibana
   1) The important part is the message
   2) Set to display the message only
   3) That is what we need to parse

Logstash Parsing - Grok

- Grok development can be done online or locally with Logstash
  - Locally running from the command line
  - There are online grok testers
    - http://grokconstructor.appspot.com/do/match
    - Online testers can sometimes have mixed results I.e just because it works online doesn't mean it work in Logstash
- You need a small log sample
- There are many prebuilt grok patterns
  - https://github.com/logstash-plugins/logstash-patterns-core/blob/master/patterns/grok-patterns
  - https://grokdebug.herokuapp.com/patterns
  - These are very useful

Groking grok

Given this log event what do we want to parse?

137.189.160.206 - - [31/Jan/2019:23:29:39 -0600] "GET /~ladd/virginia_pisu.html" 200 915

- Source
- Timestamp
- HTTP method
- Request
- Response
- Bytes returned

Total – 6 Keys

Grok baby steps (online method)

1) Basic grok pattern syntax:
   1) %{PATTERN:FieldName}
   2) Pattern is always upper case
   3) GREEDYDATA matches anything but is not a good practice
2) Go to http://grokconstructor.appspot.com/do/match
3) Paste the contents of uofs_5.log into the input
4) Start with %{GREEDYDATA:spam}
5) Click Go

ix-oly-wa2-11.ix.netcom.com - - [31/Dec/1995:23:57:00 -0600] "GET /~scottp/publish.html" 200 271	
MATCHED	
spam	ix-oly-wa2-11.ix.netcom.com·-·-·[31/Dec/1995:23:57:00·-0600]·"GET·/~scottp/publish.html"·200·271
freenet.npiec.on.ca - - [31/Dec/1995:23:58:06 -0600] "GET /~scottp/free.html" 200 23067	
MATCHED	
spam	freenet.npiec.on.ca·-·-·[31/Dec/1995:23:58:06·-0600]·"GET·/~scottp/free.html"·200·23067
moondog.usask.ca - - [31/Dec/1995:23:58:29 -0600] "GET / HTTP/1.0" 200 3890	
MATCHED	
spam	moondog.usask.ca·-·-·[31/Dec/1995:23:58:29·-0600]·"GET·/·HTTP/1.0"·200·3890
moondog.usask.ca - - [31/Dec/1995:23:58:37 -0600] "GET /search.html HTTP/1.0" 200 1838	
MATCHED	
spam	moondog.usask.ca·-·-·[31/Dec/1995:23:58:37·-0600]·"GET·/search.html·HTTP/1.0"·200·1838
ns1.maf.mobile.al.us - - [31/Dec/1995:23:59:28 -0600] "GET /~scottp/free.html" 200 23067	
MATCHED	
spam	ns1.maf.mobile.al.us·-·-·[31/Dec/1995:23:59:28·-0600]·"GET·/~scottp/free.html"·200·23067

Grok baby steps (online method)

1) Looking at grok patterns we can use NOTSPACE for the requestor

1)%{NOTSPACE:requestor}%{GREEDYDATA:spam}

MATCHED	
requestor	ix-oly-wa2-11.ix.netcom.com
spam	·-·-·[31/Dec/1995:23:57:00·-0600]·"GET·/~scottp/publish.html"·200·271

2) We don’t care about ‘ - - [‘ so don’t assign a pattern to them

3) %{NOTSPACE:requestor} - - [%{GREEDYDATA:spam}

4) That went boom. Any idea why?

5) Special charaters are escaped by a ‘\’

1) %{NOTSPACE:requestor} - - \[%{GREEDYDATA:spam}

• We can see the characters we don’t care about are dropped.

Grok Baby Steps - Timestamps

- Timestamps can be a pain. Look for a prebuilt pattern that fits (and there are a LOT of them).
- You can always build one by adding multiple parsed item into one
  - Format: (?<field_name>%{Key1}%{Key2})
  - (?<DTS>%{DAY}\/%{MONTH}) etc

Grok Baby Steps – Timestamps (cont)

For this timestamp - 31/Dec/2018:23:57:00 -0600 lets build a dts (Date Time Stamp) field

1) Paste just the time field in the online grok tester.
2) Getting started
   (?<dts>%{MONTHDAY})
3) Develop the rest of the parser

Grok Baby Steps – Timestamps (cont)

Solution timestamp - 31/Dec/2018:23:57:00 -0600

(?<dts>%{MONTHDAY}\/%{MONTH}\/%{YEAR}:%{TIME} %{ISO8601_TIMEZONE})

- That was probably not very easy and more than a little error prone.
- Had we searched the patterns better we would have realized there is a prebuilt pattern for this, HTTPDATE
- However, knowing how to build a field out of different pieces of an event log can be a very powerful tool.

Grok Baby Steps (cont)

Grok syntax

```
filter {
    grok {
       match => { 'message' => "<GROK FILTER HERE>" }
     }
}
```

Grok Baby Steps (cont)

Back to our full UofS log
Time permitting complete the grok pattern

Grok Baby Steps (cont)

Full grok pattern:

%{NOTSPACE:requestor} - - \[%{HTTPDATE:dts}\] \"%{WORD:http_method} \/%{GREEDYDATA:resource}\" %{INT:response_code} %{NUMBER:bytes_returned}

moondog.usask.ca - - [31/Dec/1995:23:58:29 -0600] "GET / HTTP/1.0" 200 3890	
MATCHED	
requestor	moondog.usask.ca
response_code	200
bytes_returned	3890
http_method	GET
resource	·HTTP/1.0
dts	31/Dec/1995:23:58:29·-0600

Dissect Filters Overview

- Where grok uses RegEx's to match what's in a field dissect uses what separates a field
- Benefits
  - Easier to develop
  - Not as resource intensive as grok
    - RegEx matching can be resource intensive
    - GREEDYDATA is expensive on the resource side
  - Less error prone
    - If you have an INT and you get an alpha the parser breaks

Drawback

- No online tester that I know of

Dissect Filters Overview (cont)

- Many of the same capabilities as grok
  - Can drop data
    - %{}
  - Can combine in several fields
    - %{dts} %{+dts} %{+dts}
    - Combines the 3 fields into a DTS field

- Back to our full UofS log

Dissect Filters Hands-On

Similar to grok but parses on separators
What separates the values we want?

ix-oly-wa2-11.ix.netcom.com - - [31/Dec/1995:23:57:00 -0600] "GET /~scottp/publish.html" 200 271

%{requestor} - - [%{dts}] "%{http_method} /%{resource}" %{response_code} %{bytes_returned}

Note – only characters that need to be escaped are ?, +, &

Dissect Filters Hands On

1) Run Logstash from the command line using the UofS_dissect.conf
2) If stdout looks good edit the conf file and uncomment the Elasticsearch output
3) Run it again
4) Go to Kibana and view the results
5) What is the problem?
6) Stop Logstash

Kibana Dev Tools Detour

- Open the Kibana Dev Tools tab
- Dev tools allow you to interact with the Elasticseach backend
- Very useful when developing filters among other things.
- Some useful commands
  - GET _cat/indices – Shows information on all of the indices
    - Health Status Index Name UUID Primary Shards Replicas Docs Deleted Docs Size Total Size
    - yellow open logstash-2019.03.07 Xd2iFfGrT0uZDvmsnODPJg 5 1 1028 0 239.3kb 239.3kb
    - Why is the Logstash index yellow?
    - Why is the .kibana index green?
  - Delete <index name>
    - This is what we want. Delete the Logstash index so we only have the latest data
      - DELETE logstash-2019*
    - This is VERY powerful, use with care, especially wildcards

Dissect Filters Hands On

1) Rerun the command line Logstash
2) Back to our timestamp problem
3) By default Logstash uses its timestamp and not the timestamp in the log which is not want we want.
4) Enter the Logstash Date filter

Logstash Date Filter

- Date filter will let us use a field as a date in Elasticsearch
- Follows the joda format
  - http://joda-time.sourceforge.net/apidocs/org/joda/time/format/DateTimeFormat.html
- The dts format
  - 31/Jan/2018:23:59:28 -0600
- Joda equivalent ?

Logstash Date Filter - Hands on

Develop that date filter (time permitting)

- Date filter info
  - https://www.elastic.co/guide/en/logstash/current/plugins-filters-date.html
- Logstash conf file filter
  - Syntax:
    match => ["field name", "Joda format"]
    target => "dts"
- This gets added in the conf file filter section under the dissect filter

```
date {
      match => ["dts", "<your Joda filter here>"]
      target => "dts"
   }
```

- Use local Logstash to validate

Logstash Date Filter

Solution

```
date {
    match => ["dts", "dd/MMM/YYYY:HH:mm:ss Z"]
    target => "dts"
 }
```

- Check Kibana to see if it pulled the right date
- If not what is the problem and how do we fix it?

Grok Summary

- Custom fields
- Drop a filed by matching a pattern but do not give it a key
  - Create a key value - %{HTTPDATE:dts}
  - Drops the data - %{HTTPDATE}
- GREEDYDATA will match everything until it gets to the end or matches the next pattern
  - “%{GREEDYDATA:eggs}” matches everything
  - “%{GREEDYDATA:eggs}%{INT:response}” will stop matching when a number is seen

Back to Kibana

- Elasticsearch does a best guess on field types when it builds an index for the first time or when it sees a new field.
- What you see in the Kibana Index Pattern is what Elasticsearch picked
- We can tell Elasticsearch what types we want by using a template
- If we want to use our DTS field as the timestamp we need to delete the current index and push the data through again.
- When we recreate the index pattern we can use the DTS field as the timestamp

Back to Kibana

Deleting an index pattern

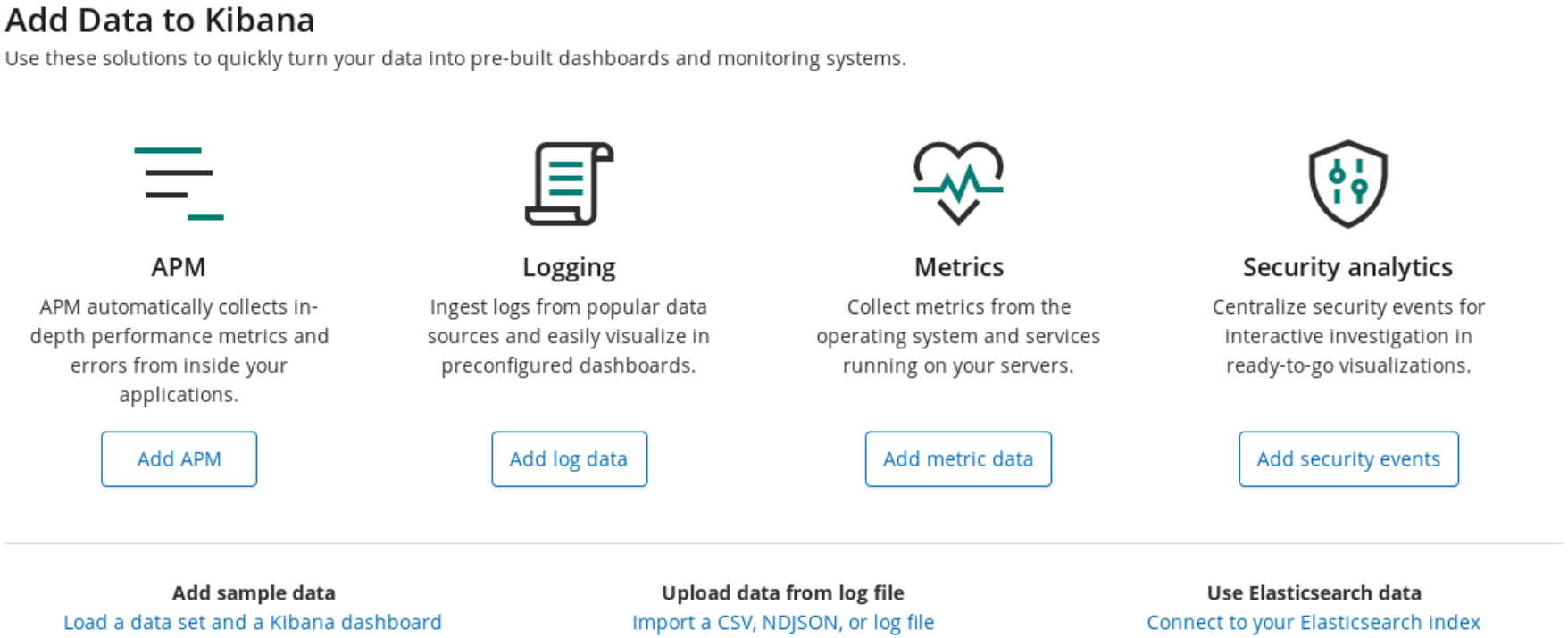
1) Go to Kibana → Management → Index Patterns
2) Select the Logstash* index
3) Click on the red trash can in the upper right
4) Rerun Logstash
5) Recreate the Index Pattern in Kibana but choose the DTS field as the timestamp
6) Validate that the timestamp is the log's timestamp

Dissect Summary

- Dissects are
  - Faster to process
  - Use fewer system resources
  - More robust
  - Easier to develop
- When to use a dissect over a grok?
  - If the data is uniform go with dissect
  - If the data is not uniform then you'll probably have to sue a grok

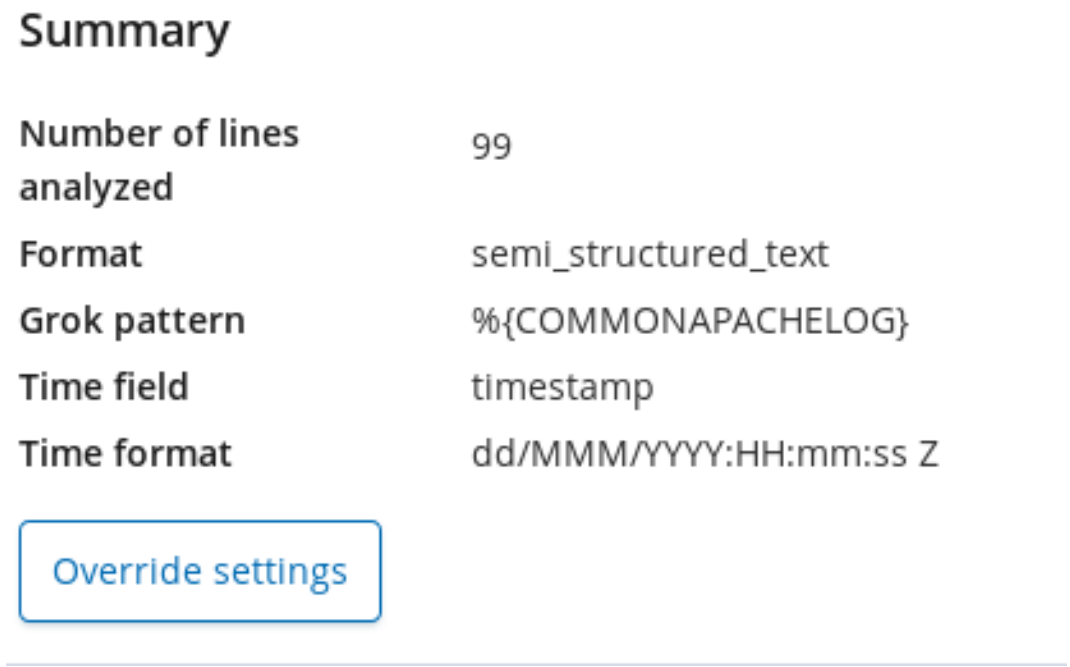
And now, the REALLY easy way

- Kibana added a File Data Visualizer feature.
- It does a best guess for a parser and shows data on a selected log file.
- http://localhost:5601/app/kibana#/home?_g=()



And now, the REALLY easy way (cont)

1) Click the “upload data from log file”
2) Select the uofs_100.log
3) The machine learning will parse the log, give some details, and a grok pattern



And now, the REALLY easy way (cont)

Grok filter based on Kibana:

```
filter {
   grok {
      match => { 'message' => "%{COMMONAPACHELOG}" }
    }
}
```

Why is this important?

- Knowing how to write filters is critical to parsing logs.
- Knowing the options (and there are more we did not cover) will give you a better chance of success
  - Spend some time and play around with some of the other filter options
  - The more tools you have in your toolbox the better
- Choosing the right approach does take time and experience
- The ‘easy way’ is a Machine Learning feature.
  - It’s currently beta, not supported
  - Machine Learning is part of the paid package


Questions?

Logstash – Logic & Enrichment

Logic & Enrichment - Goals

- Use logic in a conf file to add relevant data.
- Enrich an event by adding fields and values
- Clean up the feed so we are only saving what we need to save
- Use the data to send to a specific index

Logic & Enrichment – Using Logic

- Logstash supports the following conditionals:
  - Logic
    - if, else if, and else statements
  - Equality
    - ==, !=, <, >, <=, >=
  - Inclusion
    - ‘In’ and ‘not in’
  - RegEx
    - =~ and !~
  - Boolean
    - and, or, xor, and nand (not “and”)

Logic & Enrichment – Using Logic

- IF formats

```
if [field name] == expression {
    do this
    }

if [term] in [field name] {
    do this
    }
```

- IF example

```
if [response_code] == 404 {
    mutate { add_tag => “we failed”
    }
```

Logic & Enrichment – Mutate

- Mutate filter allows us to add, remove, replace, rename, and modify fields
- It operates on an order of precedence or the order of the mutate filters in the conf file.
  - When in doubt break out the mutate calls in the conf file
- Complete list:
  - https://www.elastic.co/guide/en/logstash/current/plugins-filters-mutate.html
- Useful for optimizing what is sent to Elasticsearch for storage
  - Only send what you need to store.
    - If you parse the entire message do you really need to store the message field?
  - Add (enrich) by adding tags or fields.

Logic & Enrichment – Mutate

Mutate Examples

- Add Field

```
mutate {
   add_field => [ "field_name", "field_value" ]
}
```

- Remove Field

```
mutate {
   remove_field => [ "field_name1", "field_name2" ]
}
```

- Lowercase
  - useful if a field is going to be part of an index name which is a requirement for index names

```
mutate {
   lowercase => [ "fieldname" ]
}
```

Logic & Enrichment – Index Names

- As we've seen, the default index from a Logstash feed is logstash-yyyy.mm.dd
  - This will not be very useful but is easy to fix
- We can designate an index name in the output section of the conf file.
  - Example spam_%{YYYY.MM.dd}
  - A date is NOT required
- We can also use a field value as part of the index name
- This is VERY useful for organizing feeds which makes out life in Kibana much easier.
  - Format for this is to add a line to the Elasticsearch output
  - index => "baseName_%{*field_name*}_%{+YYYY.MM.dd}
  - Example
  - index => "spam_%{source}_%{+YYYY.MM.dd}

Exercise 6 – Write a Parser

- The h2g2.log
  - Set of quotes from the Hitchhikers Guide to the Galaxy.
  - Very little data other than what was said and who said it
- What we want
  - Who said it (source)
  - What they said (quote)
  - What their occupation is (occupation)
  - What planet they are from (home_planet)
- What we do not want
  - message
  - host
  - path
- Additional Information
  - We want the events stored in an indices with the following naming convention: h2g2_source_<year>.<month>.<day

Exercise 6 – Info you may need

- Arthur & Trillian are from Earth
  - Arthur was in radio
  - Trillian is an astrophysicist
- Ford and Zaphod are from a “small planet in the vicinity of Betelgeuse”
  - Ford is a Guide Researcher
  - Zaphod is the Ex-President of the Galaxy
- Marvin
  - From the Sirius Cybernetics Corporation
  - He is a Paranoid Android
- Deep Thought
  - From Magrathea
  - Is a supernatural computer

Logic & Enrichment – H2G2.conf

```
filter {
   # filtering goes here
   dissect {
      mapping => {
        'message' => '%{source} - "%{quote}"'
      }
    }
}
```

Why is this important?

- Enrichment is one of the key areas to organizing and cataloging event data.
- What we'll see later is by breaking out the key:value pairs and adding or removing data is the searches become easier.
- You may not have a tag the is "ThisIsTheNeedleYouSeek" but will aid you greatly as you search and visualize data


Questions?

Kibana Deeper Dive

Kibana Deeper Dive

- Kibana has some very useful features for doing useful things with the data in Elasticsearch
  - Searches
    - Can be saved for future use
  - Visualizations
    - Can be based on data or saved searches
    - There are a LOT of different types
  - Dashboards
    - Made up of visualizations
  - All of these can be drilled down based on a time period
  - Let’s focus on searching. I’ve added in depth visualizations on different sample data and if we have time we’ll go over them

Kibana – Chicago Crime Setup

Lets setup the Chicago Crime Index

This is one of the best indices I have found for sample data

1) Copy the contents of DevTools-ChicagoCrime.txt into the Dev Tools tab
2) Run the POST command
3) Run Logstash locally with the chicago_crime1year.conf file
4) Create a chicago* index pattern

Kibana – Searching 101

- Set the Time Filter to Last 1 year
- Expand the Table view and lets look at the fields
  - What looks interesting? Why?
  - What Might be interesting? Why?
  - What does not look interesting? Why?
- Visualizing data can be very helpful in deciding where to spend your efforts

Hopefully, if you're responsible for this, you have some input on what comes in, why it needs to be searchable, and how it is stored (i.e. key:values and data types).

Kibana – Searching 101

- What looks interesting? Why?
  - Beat – represents an area
  - Block – represents a location
  - Date of occurrence – Date is a key field
  - Primary Description – Main reason for arrest
  - Secondary Description – More detail on arrest
  - Time – Time of arrest is key
  - Ward – More location info
  - Coordinates – Can use range searches
  - DTS – More location info

Kibana – Searching 101

- What Might be interesting? Why?
  - Arrest – Either ‘Y’ or ‘N’, depends on use case
  - Domestic – depends on your use case
  - FBI CD / IUCR – no idea what this means but is numeric, has to be something meaningful to someone
  - Location Description – is “sidewalk” useful?

Kibana – Searching 101

What does not look interesting? Why?

- Case Number – will always be unique
- X & Y Coordinate – Useful but not for searches
- Anything that starts with an “_”. These are internal data fields
- Path – This should have been dropped
- Host – Normally meaningful but not here

Searching 101 - Visualizations

- Visualizations can be helpful in pulling meaningful data
  - Tag Cloud – shows top or bottom number of values
    - Some of those “maybe useful” would give us some info
  - Geo mapping can help on drilling down on locations

Searching 101 - Visualizations

Visualizations 101

1) Click on the Visualize menu item
2) Click the “+” to add a new Visualization
3) Select the type
4) Select the index
5) We will be using chicago* for all of these
6) Save the visualization using a useful name
   <index Name><VisualizationType><data>
   Example:
   Chicago-TagCloud-PrimaryDesc

Searching 101 - Visualizations

Tag Cloud

Lets create a Tag cloud for a few things and see if we can get an idea of what is important.

- Beat
- Primary Description
- Secondary Description
- FBI CD
- IUCR
- Location Descirption
- Ward

Searching 101 - Visualizations

Tag Cloud
Lets go through some of these

Select a Tag Cloud visualization

1) Select a term (Primary_Description.keyword)
2) Click the Play icon
3) Is that useful?
4) Set the size to 20
5) Is that better?
6) Change the order to Ascending
7) Is that useful?
8) Save them both

Searching 101 - Visualizations

Tag Cloud
For the remainder you create them

Let's try IUCR

1) Select a Tag Cloud visualization
2) Select a term (IUCR.keyword)
3) Click the Play icon
4) Is that useful?
5) Set the size to 20
6) Change the order to Ascending
7) Is that useful?
8) Should we spend time figuring out what IUCR means?
9) Save them both

Searching 101 - Visualizations

- IUCR code (Illinois Uniform Crime Reporting code) which are four digit codes that law enforcement agencies in Illinois generally have adopted to use to classify criminal incidents when taking individual reports. Other states will have their own set of code
- Explanation :

https://data.cityofchicago.org/Public-Safety/Chicago-Police-Department-Illinois-Uniform-Crime-R/c7ck-438e/data

Searching 101 – Lucene Searches

Bookmark this page -
https://lucene.apache.org/core/2_9_4/queryparsersyntax.html

- Elasticsearch (and Solr) use Lucene on the back end
- Lucene IS a search engine.
- The link above details the options we have at our disposal to perform advanced searching

NOTES

- Search terms and phrases are not sensitive to case
- Logical operators and Field names are.
- There are a lot of search options that will not be covered. The link above has all of the info

Searching 101 – Lucene Searches

General Info

- Searches are based on terms and phrases
  - ***error*** is an example of a term
  - ***“read error”*** is an example of a phrase including the quotes
- Specific fields can be searched for terms or phrases
- Syntax : FIELD_NAME:”search term or query”
- Example : BEAT:"1531"
- AND, OR, and NOT are supported logic options
- + and – are shortcuts for “must contain” or ‘must not contain’
  - Example (these are the same)
  - PRIMARY_DESCRIPTION:theft AND BEAT:1531
  - +PRIMARY_DESCRIPTION:theft +BEAT:1531
-

Searching 101 – Lucene Searches

Searches will be performed in the Kibana Discover tab using the chicago* index

- Start simple, search for **theft**
  - Note the number of hits (top left)
  - Note the wide variety of matches
    - Theft
    - Retail Theft (in secondary description)
    - Motor Vehicle Theft
    - What does ***theft +motor*** do? Why?
    - How can we find anything that is not Motor Vehicle Theft?
    - ***theft +motor -vehicle***
- Searches can be saved
  - Use a descriptive name with the index in it
  - ChicagoMotorTheft_not_Vehicle

Searching 101 – Lucene Searches

- The auto complete function can be annoying. Clicking “New” will fix that
- Let’s try some searches
  - See *exercise*ex6.txt for some searches


More Kibana – If we have time...

Kibana Deeper Dive – Setup the Index

- We need to tell Elasticsearch how we want the data stored, specifically:
  - We need to set a timestamp
  - We want to setup a geopoint
    - Geopoints can be setup with the following:
      - Longitude and Latitude
      - An IP
- Delete the existing Logstash index
- Delete the existing Logstash index pattern
- Run Logstash with the es_sampleData.conf file
- Set the index pattern as “logstash*”
- Set the time filter to Absolute Time Range
  - From - 2018-06-17 16:00:00.000
  - To 2014-06-17 00:08:00.000.

Kibana – Visualizations - Geo

Display the Geo location from where the requests were originated

- Go to the Visualizations tab
- Select Coordinate Map
- Select the Logstash* index
- Click on Geo Coordinates
- Select Geohash (only option)
- Select geoip.location from the Field drop down
- Click the Play icon
- Where’s the data?
  - Get used to this, it still happens to me
- Save this as LS-geo
  - Good practice to use the index name in whatever you save

Kibana – Visualizations - Geo

- Select Options (next to Data)
- Try the different Map types with the different cluster sizes
  - NOTE – You have to click Play to see the changes
- If you do not have internet access you will lose some of this
  - There are paid and free version of geo databases to use.

Kibana – Visualizations – Vertical Bar

Response Codes Over Time

1) Select a new Visualization – Vertical Bar
2) In the X Axis select Date Histogram and timestamp as the field
3) Click Add sub-buckets and select Split Series
4) Select Terms as the sub aggregation
5) Select response.keyword the field
6) Click the Play icon
7) More values can be added by increasing the count
8) Save the search as LS-verticalBar-ResponseCodes

Kibana – Visualizations – Tag Cloud

Tag Cloud – Useful as a quick way to see largest or smallest data values

- Open a new Kibana Visualize window
- Click the “+” to create a new visualization
- Select Tag Cloud
- Select the Chicago index
- Click on the Tags icon
- Select Terms for the Aggregation
- Under the Field drop down select Primary Description.
- For size select 20
- Click the Play icon
  - Did we forget something?

Kibana – Visualizations – Tag Cloud

User Agents

1) Select a new Visualization – Tag Cloud
2) Select Terms for the Aggregation
3) Under the Field drop down select useragent.name.keyword
4) Click Play
5) Save the search as LS-TagCloud-UserAgent

Kibana – Visualizations – Area Chart

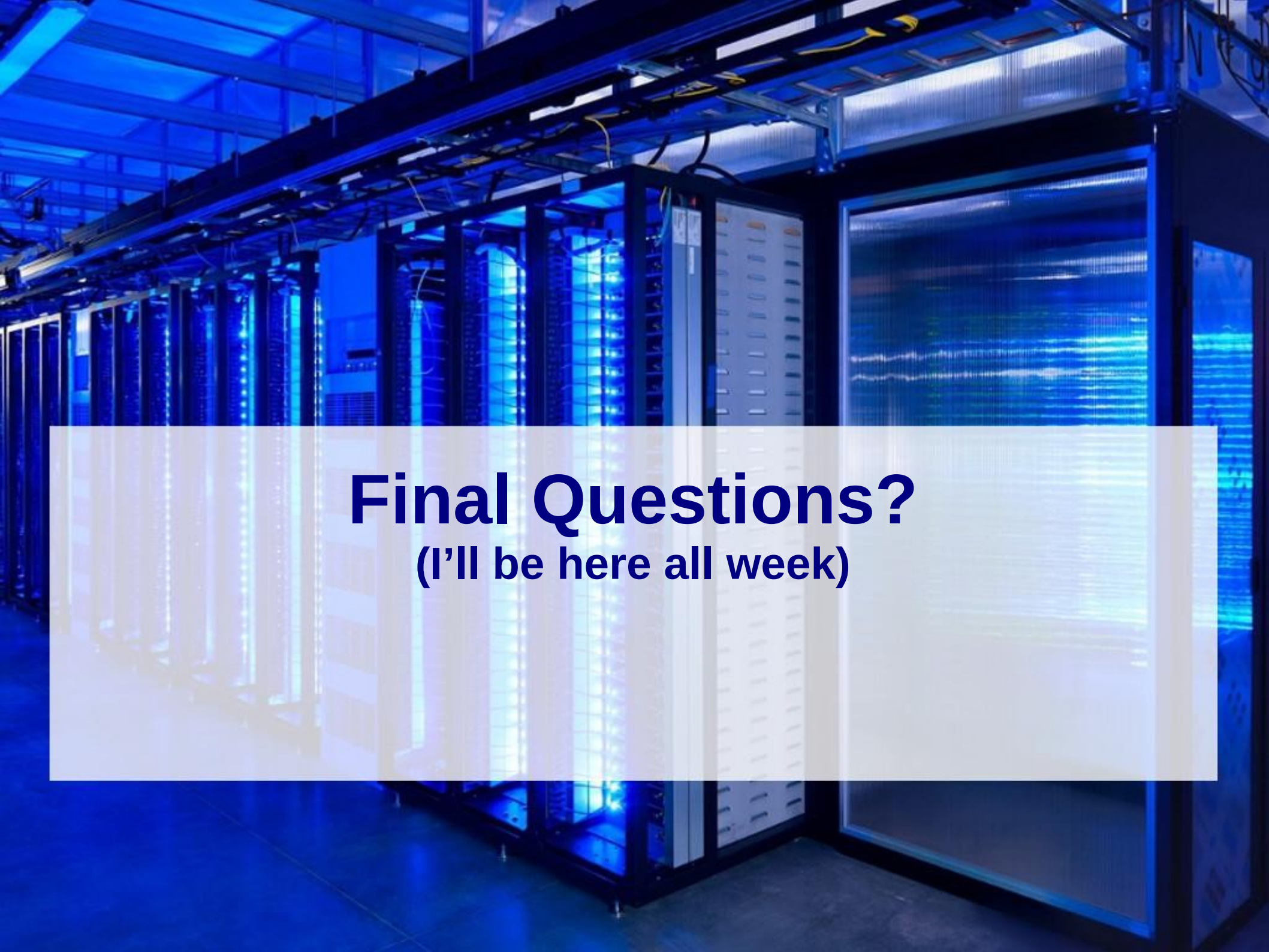
Top 5 Countries

1) Select a new Visualization – Area Chart
2) In the X Axis select Date Histogram and timestamp as the field
3) For the Y axis select the Aggregation Type as Sum with Bytes as the field.
4) For the X axis select Date Histogram with timestamp as the field.
5) Click Add Sub Buckets – Split Series
6) Select terms for the sub aggregation and geoip.country_name for the field
7) Click the Play icon
8) Save the search as LS-AreaChart-ByCountry

Kibana – Visualizations – Data Table

Top 10 URL's Requested

1) Select a new Visualization – Data Table
2) In the X Axis select Date Histogram and timestamp as the field
3) Click Add sub-buckets and select Split Rows
4) Select Terms as the sub aggregation
5) Select request.keyword as the field
6) Click the Play icon
7) More values can be added by increasing the count
8) Save the search as LS-DataTable-TopURLs


Final Questions?
(I'll be here all week)