ELK

Tips, Tricks, and Lessons Learned

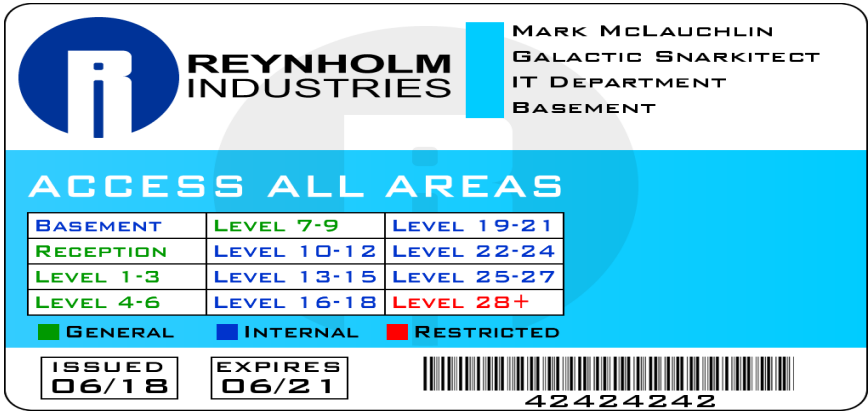
Agenda

- Introductions
- Why Care?
- Basics
- Our Setup
- ELK components
- Automation
- Standardization
- Monitoring
- Troubleshooting
- Miscellaneous
- Questions

Intro – Mark

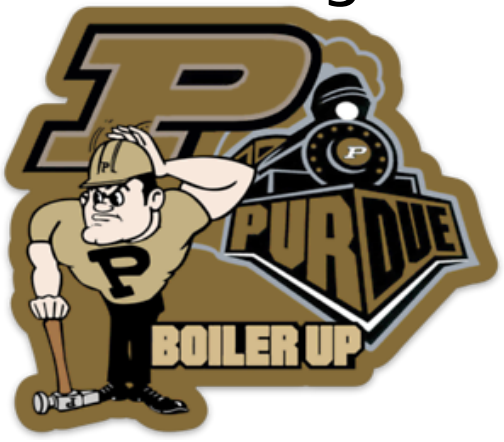
- Senior Systems Engineer, Security Tools @ GM, 7+ years
- Healthy interest in Info Sec, Raspberry Pi's, and Arduinos
- Twitter - @whats6times9
- https://www.linkedin.com/in/markdmclauchlin

#iworkforgm



Intro - Nico

- Systems Engineer, Security Tools @ GM, 6 Years



FIFA WORLD CUP
RUSSIA 2018

Technical University of Denmark DTU

https://www.linkedin.com/in/nicolas-taina-5586663b/

- #iworkforgm

Also ……

The story

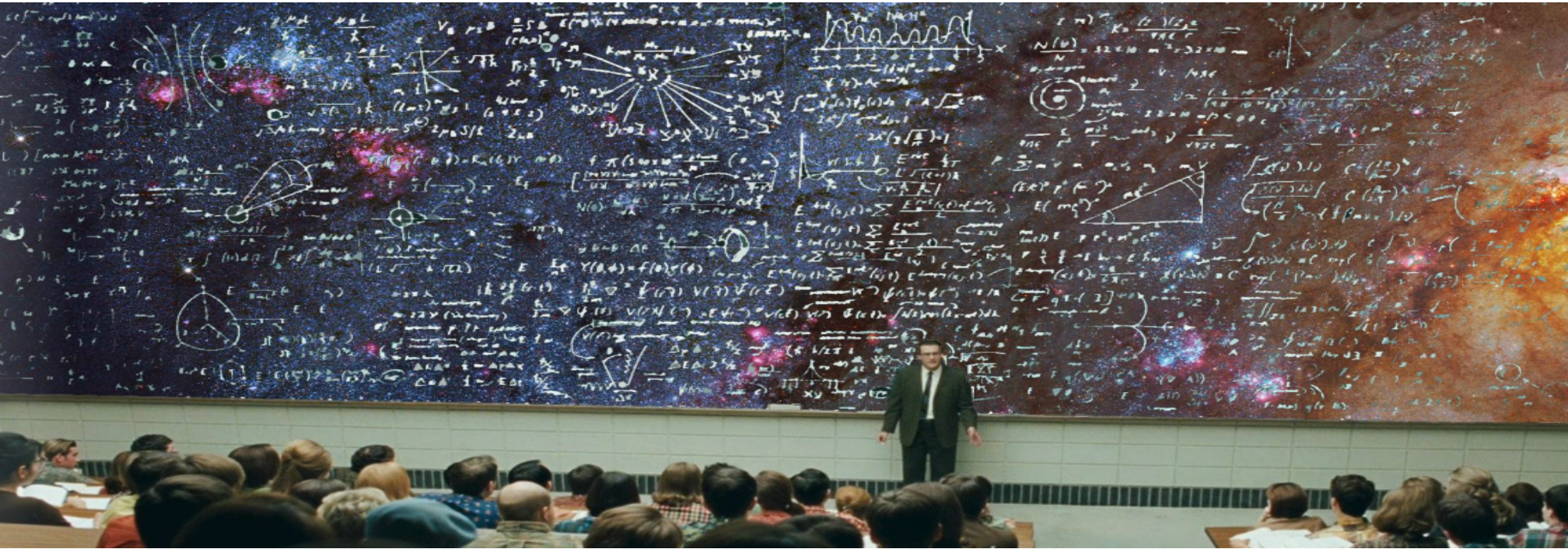
"Can you check the errors from yesterday between 15.02 and 15.07 ?!"

Why ELK (and why are we here)?

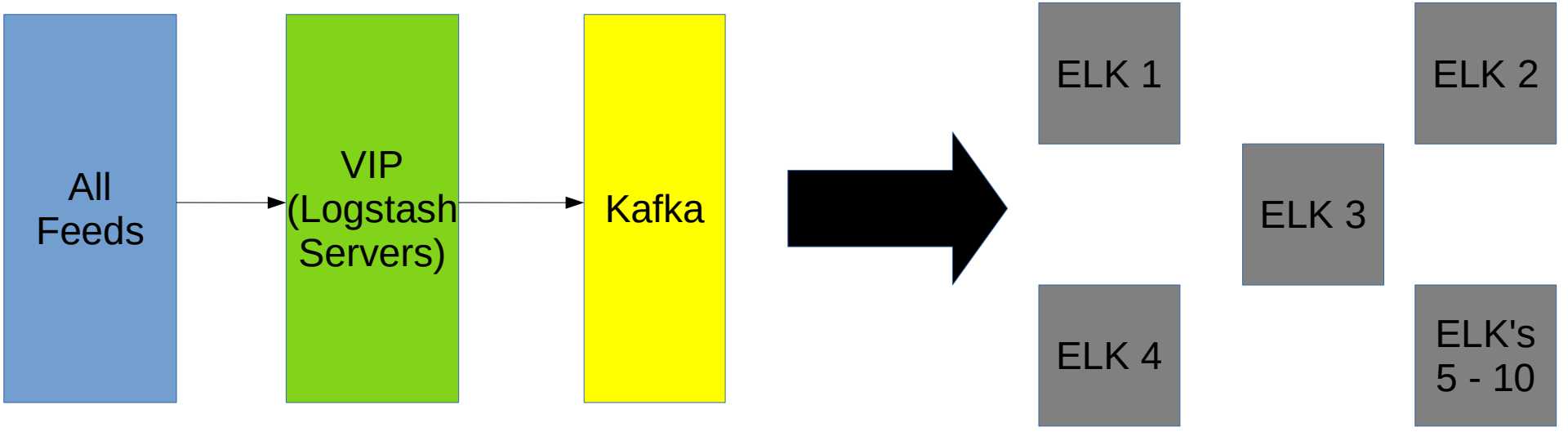
ELK can serve multiple purposes

- Log Management
  - ELK Stack
- SIEM
  - Elastic Basic License
- File Integrity Monitoring
  - Audit Beat
- Netflow
  - Elastiflow
- Search Solutions
  - Elasticsearch API's

… and it’s not hard but not intuitive

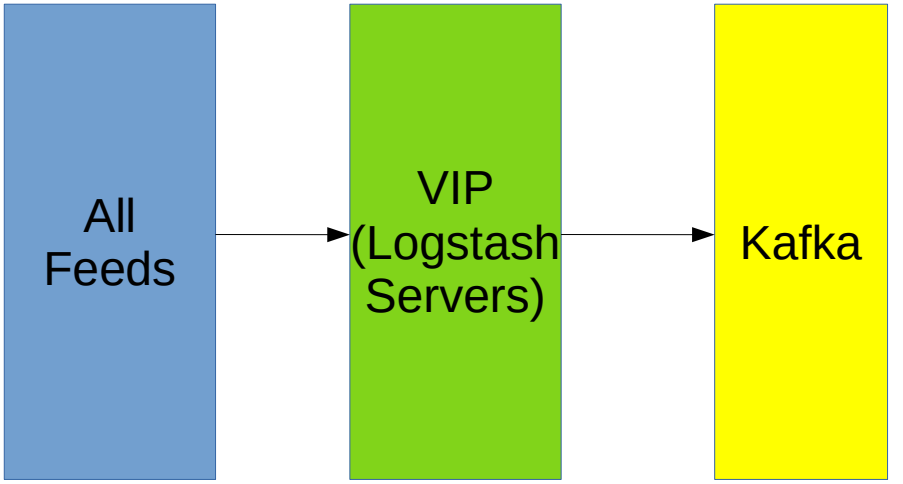


How We Use ELK



“Collect Once, Use Many”

- ***This is the best thing we ever did!***
- Highly scalable, future proof
- Apps outside of ELK can pull events from Kafka (or any pub/sub messaging system)
- Logstash does nothing but send events to a topic
- All events are pushed, no pulls
- Events queue on Kafka for upgrades, no data loss
- Events can queue if there are Kafka issues



Best Practices - Elasticsearch

- Get your sharding right
  - 50-100GB is ideal
  - Over sharding is a resource waste
  - Under sharding can take your stack down
    - We speak from experience
  - Index Life-cycle Management can help but has its downsides

Best Practices - Elasticsearch

- Look to object store (S3 or compatible) for long term storage
  - Cheaper than disk storage (ask your storage team)
  - Automate snapshots

- Python can be your friend
  - Allows you to easily automate routine tasks
  - 2 Packages to get to know
    - elasticsearch
      - Low level package but great for getting stack metrics and information
    - elasticsearch-dsl
      - High level package, geared more for searches but still very useful

Optimize Filters by Names

Numbering/Ordering is important

- Filters will be stitched together in alphanumeric order
- Inputs should be first, filters second, Outputs last
- Allows control over the execution order of filters
  - Some of our environments have 100+ filters
- Use prefix numbering to accomplish this
  - 100- for Inputs
  - 200 for basic wide filters (remove syslog headers)
  - 300 for app specific filter(parsing)
  - 400 for post parsing wide filters (drops)
  - 900- outputs

Optimize YML for Performance

---------- Pipeline Settings ----------
pipeline.workers: 8
pipeline.batch.size: 2000

- Pipeline workers to equal hosts CPU cores. Can be temporarily upped to double during high throughput times.
- Pipeline Batch Size is the number of events to collect before process. Will increase throughput at the cost of memory. Tuning is needed to find optimal value. We found 2000 to be a happy medium for 16GB VM's.
- JVM heap size = half system memory
- JVM Out of Memory issues
  - Increase thread stack size in logstash.yml
    -Xss4m

Best Practices - Logstash

Dissect vs Grok

Criteria	Dissect Filter	Grok Filter
Simplicity	• Easy to implement and readable	• May be preferred if familiar with regex
Delimiter varies line to line	• Dissect is better if the lines are reliably repetitive.	• Grok is a better option if flexibility is needed
Performance	• Because of simplicity, it uses less resources; hence improves the performance	• Being bases on regex, the matching can be more accurate
Robust	• More robust or less fragile than Grok as is not based on Regex	• Failures to match can quickly bog down resources

Best Practices - Logstash

Dissect Example

- Dissect works what separates the values

```
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}
```

Best Practices - Pipelines

Pipeline 101

- The main pipeline is probably the weakest link in Logstash.
  - If the pipeline goes down Logstash stops processing
  - The service will still be running which means most default monitors will not catch this issue
    - This can be caused by a filter
    - This can be caused by an output not responding
      - If Logstash can’t talk to Kafka it will stop trying
        - It ‘*looks*” like it is, but it’s not
        - Automated Python script can monitor/fix this

Best Practices - Kibana

Dev Tools is extremely useful

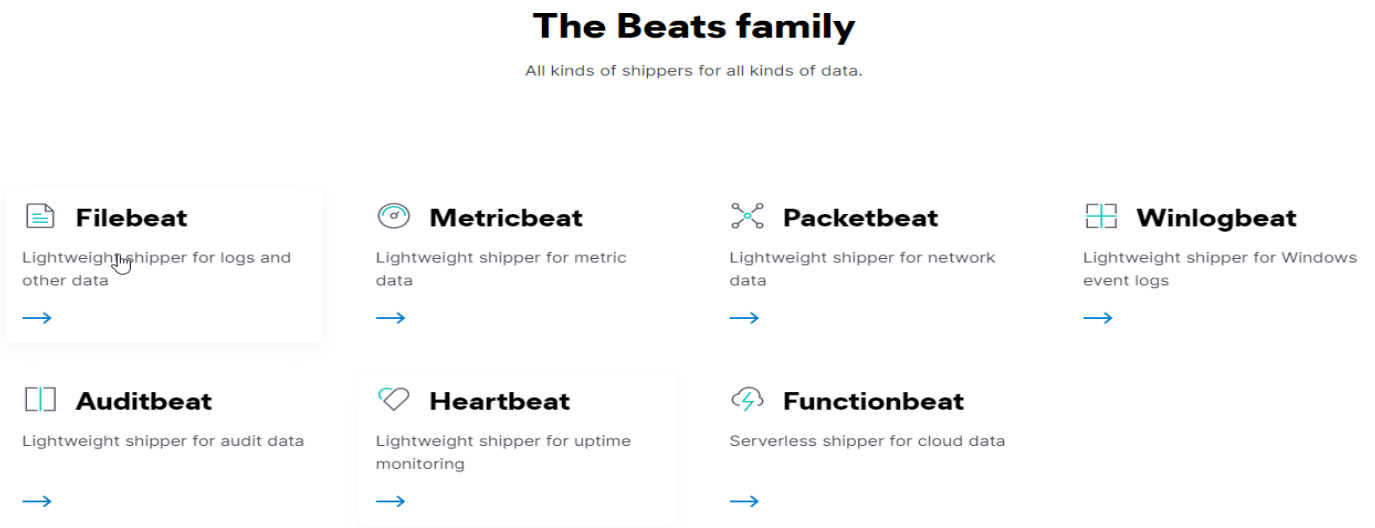
- “?help” shows all the keys that supported and there are a LOT of them
  - Ex - GET _cat/indices?help
- Tweak basic commands using headers to target keys and sorting to sort
  - Can be used in automation scripts
  - Ex - GET _cat/indices?v&h=index,store.size&s=store.size:desc
    - Returns index names and the storage size
      - &h=index,store.size
    - Sorts by the storage size, largest first.
      - &s=store.size:desc

Best Practices - Kibana

- Using the ‘Copy as cURL’ function is very valuable
  - Sometimes Kibana may not be accessible, how are you going to fix a read-only stack, problem index or other problems?
  - Get a list of useful cURL commands and save them somewhere
  - VERIFY they work
    - Just like backups, you should know they work before you need them.
    - Unlike backups validating them is easy
- Beware of using wildcards in deletes
  - Very bad things happen if you delete .kibana
  - Yes, we’ve both done that
  - Lesson Learned, snapshot .kibana indices at regular intervals

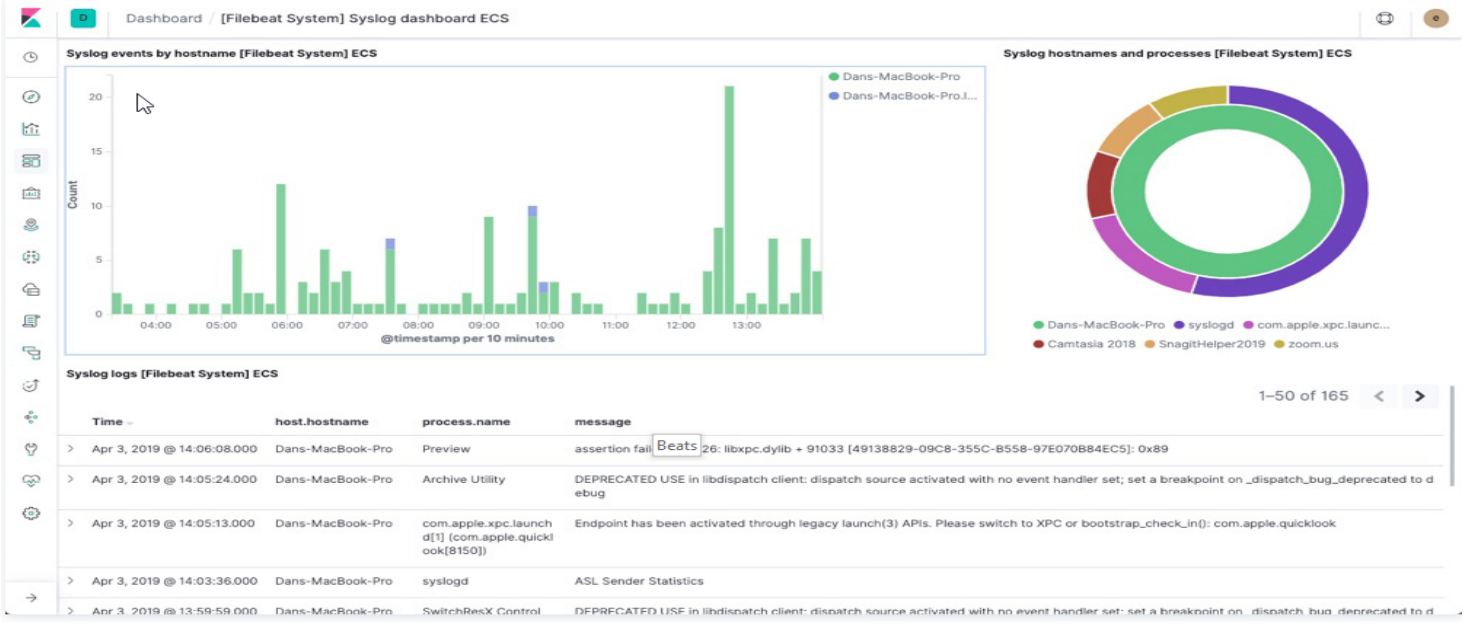
Beats

- Simplifies and streamline getting your data to Elasticsearch
- Wide variety covers a lot of use cases



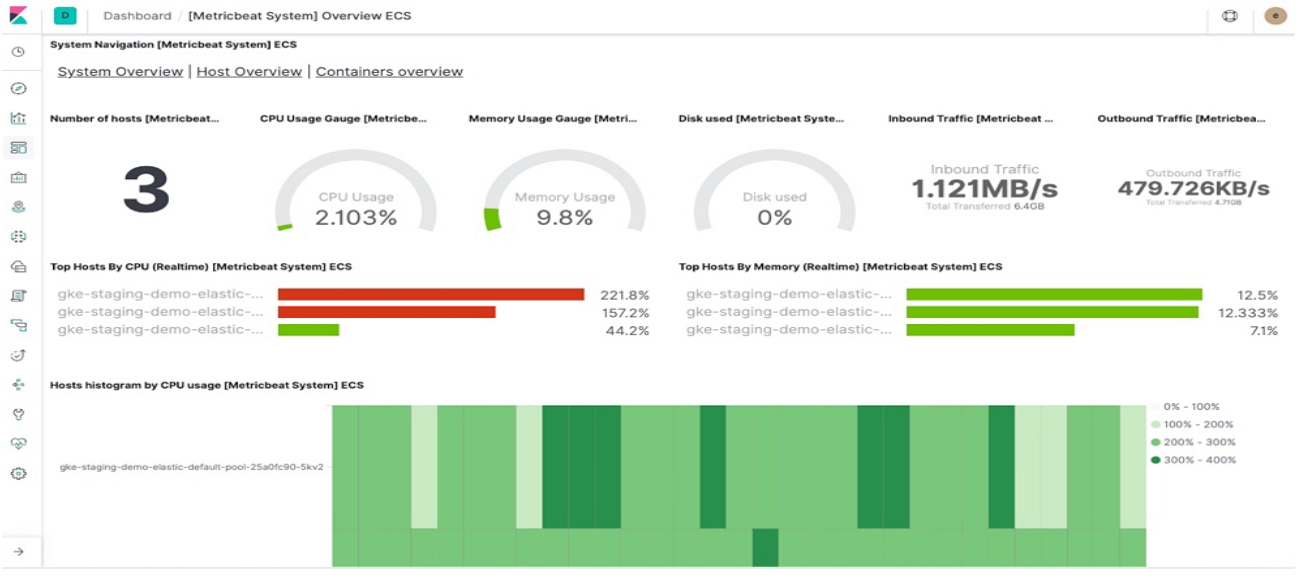
Filebeats

- Aggregate, “ tail -f ” & search
- Can take advantage of Modules, which can have default paths and basic parsing



Metricbeat

- Has many modules available
  - MongoDB, Kubernetes, Docker
- Includes modules for all the ELK stack components
- Can be customized
  - HTTP API is very powerful



Metricbeat- HTTP Module

Example (Elastic Basic License)

```
- module: http
  metricsets: ["json"]
  period: 5s
  hosts: ["https://localhost:9200/_cluster/health"]
  namespace: “health"
```

Example (Using OpenDistro Performance Analyzer)

```
- module: http
  metricsets: ["json"]
  period: 5s
  hosts: ["http://localhost:9600/_opendistro/_performanceanalyzer/metrics?metrics=Heap_Max,Heap_Used&agg=avg,avg"]
  namespace: "heap"
  processors:
  - script:
      lang: javascript
      id: my_filter
      source: >
       function process(event) {
          var jason = event.Get("http.heap");
          var cutjson = JSON.stringify(jason);
          cutjson= cutjson.slice(2, 24);
          var recordsfield = "http.heap."+cutjson+".data.records";
          event.Rename(recordsfield, "records");
          var jsrecords = event.Get("records");
          var heapmax = jsrecords[0][0]; //returns 'someVal'
          var heapused = jsrecords[0][1]; //returns 'someVal'
          parseFloat(heapmax);
          parseFloat(heapused);
          // var typep = typeof heapp;
          event.Put("heapmax", heapmax);
          event.Put("heapused", heapused);
          event.Delete("http.heap");
       }
```

Standardization

- This is a key to our success because it makes automation so much easier
- What we standardized:
  - Process flow
  - Naming conventions (indices, filters, everything)
  - File stores
  - Install paths
  - Would have been easier to say EVERYTHING we could
- #1 Rule – No Snowflakes
  - Everyone follows the standards

Why Automate?

As your stacks grow traditional approaches will just create toil, automation helps prevent that

Automation

- Scripting
  - Bash/Python are languages of choice
- Avoids rolling upgrades as scripts can upgrade entire stacks in minutes
- Filter testing and placement can be automated
- Configuration management can be maintained by tools Like Chef
- Compatible with Kubernetes and OpenShift Orchestration tools

Troubleshooting - Logstash

Test the .conf file syntax
- /usr/share/logstash/bin/logstash -f /etc/logstash/conf_files/my.conf –t
- Run manually
  - <path to Logstash executable> -f <conf file location>

- Logstash supports sending to multiple destinations (Elasticsearch, stdout, etc.) if needed
- When in doubt use the stdout
  - Will print output to command line

```
output {
    stdout { }
    }
```

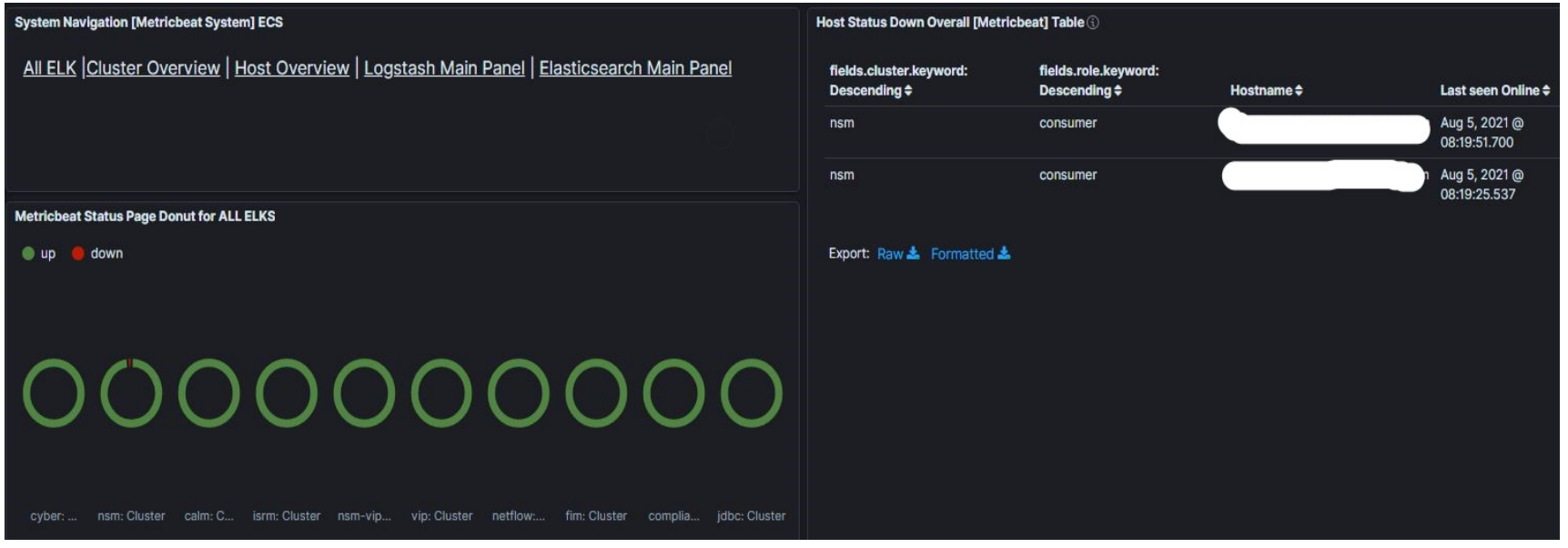
Troubleshooting - Elasticsearch

- Dev Tools and cURL commands are your friends
  - Read-only, red indices
  - Spend some time in Dev Tools to learn how to get the right data in the right form out of Elasticsearch
- Monitor Logstash Queuing Folder
  - Normal file count is ~2-4, anything over indicates an issue
  - Simple shell script to test for file count and notify
- When really bad things happen check /var/log/messages (or the equivalent)
- Services might show as being up and running but data is not flowing. Traditional monitoring/alerting will not catch these issues.

Stack Monitoring

Dashboards for all ELK’s

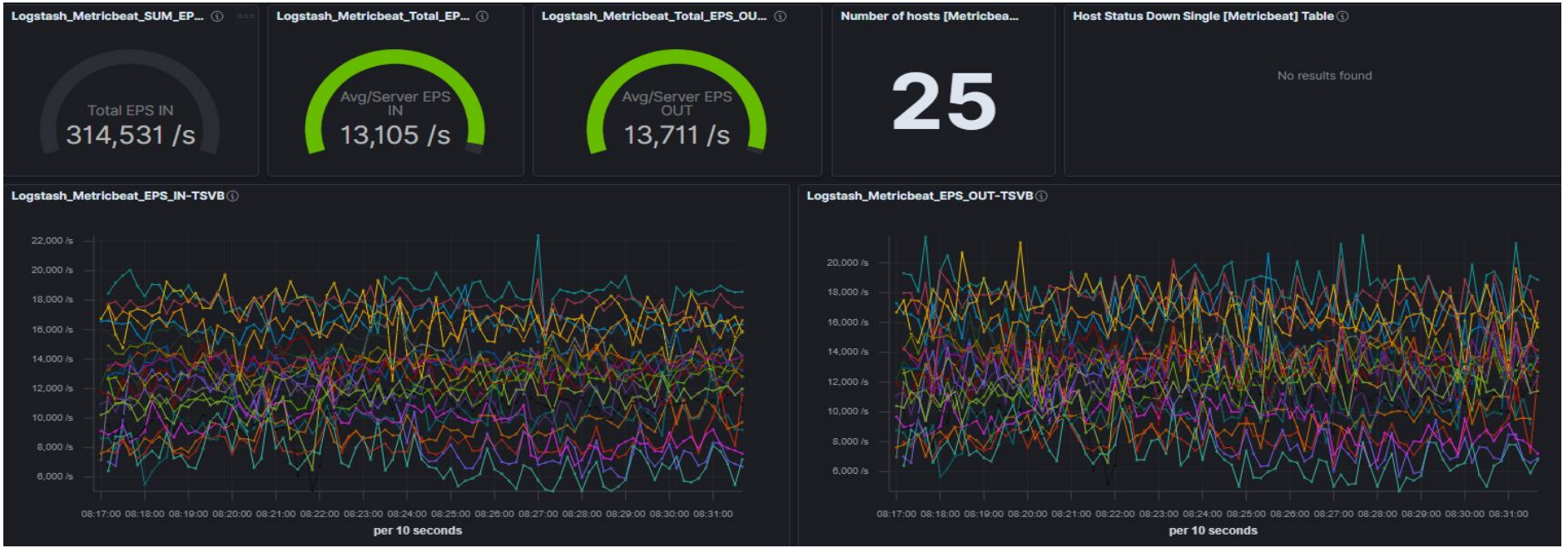
- Single pane of glass to see status of all ELK stacks
- Any issues on any stacks are shown in red
- The hosts in question are listed on the right so they can easily be remediated



Logstash Monitoring

Dashboards for Logstash

- Single pane of glass to see status of Logstash servers
- Any hosts that have not reported in are shown on the right so they can easily be remediated (none are listed below because they have all been reporting)



Elastic or Amazon

Elastic Basic (X-Pack)

- Basic free license provides SSL security, basic monitoring
  - No AD integration (comes with a paid license)
- Not true OSS, free license comes w/ restrictions
- Easy to implement comes all packaged together
- Documentation is available but deeper topics are not widely available.
- Support license is recommended and will cover almost any issue

Amazon OpenSearch (OpenDistro)

- Includes features like SSL security alerting, command line monitoring
  - AD Integration included
- True Apache 2.0 OSS license
- Plugins are all separate downloads
- Documentation is available but not quite deep and often not useful for troubleshooting.
- No support license. Support forums are not very responsive

Miscellaneous

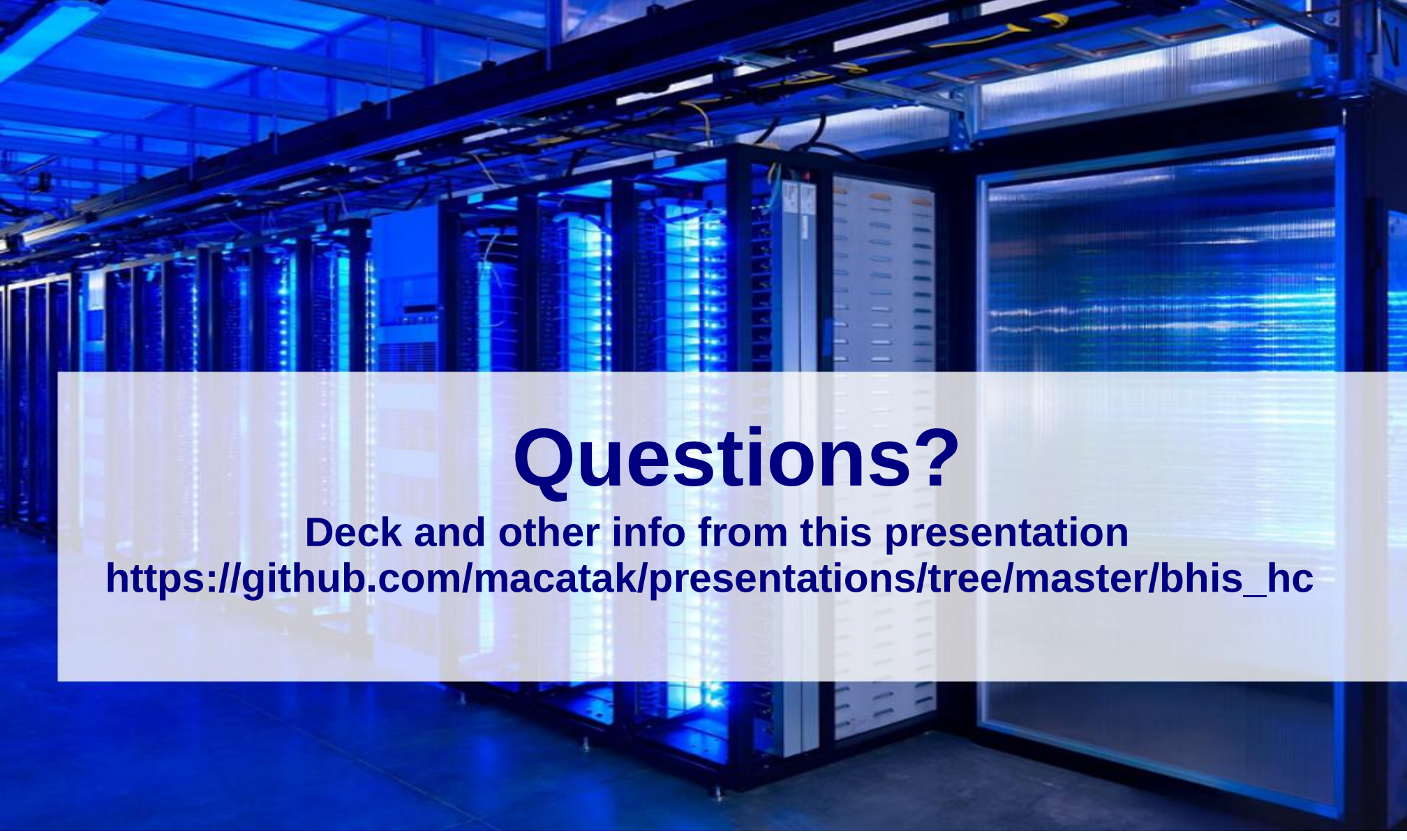
- Beware of dated material – ELK tends to have breaking changes between major version changes
  - Searches
    - Always set the search time to last 1 year or less
    - Elastic dominates search results but when troubleshooting rarely has the info you need
    - Use Advanced Search operators to eliminate results (Google specific but most search engines have them)
      - -inurl:elastic.co – Removes results from Elastic
      - -inurl:github – Error messages tend to be entered as issues
- Books
  - Check publication date or what version of ELK it covers
- Training
  - Check what version it covers

Miscellaneous

- Learn by doing
  - Virtual Machines
    - Great for trying different setups
      - Snapshots allow you to chain changes to different VM’s and evaluate
  - Docker
    - Simplest, easiest way to get up and running
  - Data.gov has tons of sample data that is perfect for learning ELK
    - Searchable by features (CSV, Geospatial)
    - Chicago Crime and NYC Vehicle collisions are great

Summary

- Automate
- Standardize
- Collect once, use many
- Dissect over grok
- No snowflakes
- Use search operators and set time for nothing over 1 year when searching for anything ELK related
- Did we mention standardize??



Questions?

Deck and other info from this presentation
https://github.com/macatak/presentations/tree/master/bhis_hc