An introduction to elk log aggregation and analysis

The Elastic Stack
The problem with logs

- Inconsistent formats
  - Even the timestamps often have different formats
- Logs are often distributed
- Enhanced access privileges often needed
- Files are difficult to search
  - Grep doesn’t scale very well
  - Got Regex?
- Expert knowledge is often required to interpret them
What can the Elastic Stack do?

• Centralized aggregation
• Easier to provide wider access to log data
• Provides ways to extract more meaning from the log data
  • Turns raw data into actionable information
  • Filters, aggregations and graphs
• Scales as needed
But what can it do?

• Case Study:
  • I get an email
  • The email contains
    • A problem
    • A SIS Emplid for a student
    • A general timeframe.
Search by Time

Search by Patterns
<table>
<thead>
<tr>
<th>Time</th>
<th>EMPLID</th>
<th>NETID</th>
<th>PVI</th>
<th>Severity</th>
<th>facility</th>
<th>host</th>
<th>message</th>
</tr>
</thead>
</table>
Elapsed time before relevant log data found

00:00:25
Grafana Integration
ELK?

- Elasticsearch
  - Near real-time search and analytics
- Logstash
  - Plumbing and glue
    - Buffers incoming log data if necessary
  - Normalizes log data
- Kibana
  - UI to search Elasticsearch as well as create visualizations
Logstash

- Logstash configuration file has three sections
  - Input
    - Defines how log data gets into Logstash
  - Filter
    - Optionally processes data
  - Output
    - Sends processed data somewhere else (typically Elasticsearch)
Logstash Input

• Getting the logs to Logstash
  • Directly read files with Logstash
  • Filebeat
    • Tails one or more logs on a server, sends the data to Logstash
• Application logging plugins
  • Logback GELF appender (JAVA)
  • gelf4net (.NET)
• There are 50 input plugins for Logstash on the Elastic website
Logstash Filter

- Convert unstructured log data into structured data
  - grok
    - Swiss army knife sort of pattern matcher
    - `{ "message" => "\%{TIMESTAMP_ISO8601:logstamp}\n\%{LOGLEVEL:log_level} (\%{PVI:pvi})? (\%{EMPLID:emplid})? (\%{WORD:netid})? \%{NOTSPACE:thread} \%{JAVACLASS:className} - \{StopWatch \%{DATA:operation}': running time \(\{\text{millis}\}\) = \%{NUMBER:elapsed}\}|\%{GREEDYDATA:message}}"}
  - extractnumbers
    - Extracts numbers from a string so that aggregations can be done on the value later

- mutate
  - Modifies parsed field values, can add fields

- useragent
  - Parses UserAgent strings into fields
  - Almost 50 filters listed at Elasticsearch
Logstash Output

- Sends processed data somewhere else for storage
  - elasticsearch
    - Just give it the server address and account credentials if needed
  - Stdout
    - Prints processed log lines to console
- Over 50 output plugins listed at Elasticsearch
Elasticsearch

• Not just for logging
• Highly searchable document database
• Fast, scalable
• Documents are organized into indexes
  • Typical Logstash configuration creates one index per day
• Underlying query protocol is JSON over REST
  • Results returned as JSON by default
• Relevance scoring, fuzzy searching, fast filtering, multi-index searches
Kibana

• Fairly lightweight web application
  • Mostly UI – composes JSON searches, displays results

• About all you need to configure is to tell it where Elasticsearch is

• Search result formats, canned searches

• Graphs and visualizations

• Interactive dashboard
Deciding what to log

- User identifiers
- Hostnames (if more than one)
- Application names (if more than one)
- Errors
- Response times
- Transaction occurrences
- State changes
- Request types
- Things that would be interesting to your stakeholders
How to log

• Try to keep a basic structure to the log lines
• Use a mapped diagnostic context library
• Put the things that every log request has first
  • Timestamp
  • Host
  • Application
  • User Identifiers (if present)
  • Operation identifiers (method names, etc)
  • The actual log message last
Questions?

- Devops@DOIT Office 365 Team
- Application Design Review Brownbags
  - Every other Friday at Noon, DoIT 3139AB
  - applicationdesignreview@office365.wisc.edu
- Brian.hill@wisc.edu