Elasticsearch, Logstash, and Other Data

John Sellens
jsellens@syonex.com

Cascadia IT, 2015
March 13, 2015

Notes PDF on USB or at http://www.syonex.com/notes/
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preamble and Introduction</td>
<td>3</td>
</tr>
<tr>
<td>The ELK Stack</td>
<td>9</td>
</tr>
<tr>
<td>Elasticsearch</td>
<td>13</td>
</tr>
<tr>
<td>Installation and Configuration</td>
<td>18</td>
</tr>
<tr>
<td>Command, Control, Management</td>
<td>32</td>
</tr>
<tr>
<td>Monitoring and Management</td>
<td>44</td>
</tr>
<tr>
<td>Logstash</td>
<td>49</td>
</tr>
<tr>
<td>Installation and Configuration</td>
<td>53</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add Ons for Logstash</td>
<td>77</td>
</tr>
<tr>
<td>Monitoring and Management</td>
<td>82</td>
</tr>
<tr>
<td>Nginx Front End</td>
<td>84</td>
</tr>
<tr>
<td>Wrap Up</td>
<td>89</td>
</tr>
</tbody>
</table>
Preamble and Introduction
Overview

- Elasticsearch – a search engine
- Logstash – filters inputs to outputs
- Kibana – web interface to Logstash/Elasticsearch
- The ELK Stack
- Introduction, Installation, Configuration
  - How to get things in and out

Notes:

- Both Cascadia IT and I will very much appreciate your feedback
Outline/Timetable

- Preamble / Introduction / Outline
- Elasticsearch
  - Overview and concepts
  - Installation, configuration, care and feeding
- Logstash
  - Overview and how it fits together
  - Installation, configuration, management
  - The ins and outs
- Kibana and its use
- Break 3:30 to 4:00, Wrap up 4:50pm

Notes:

- Scheduled for 1:30 - 4:50pm with one half hour break
- I’m hoping the timing fits together as I hope it will
Questions?

- Got a Question?
- A Clarification?
- Some Confusion?
- A Point of Interest?
- Ask!

Notes:

- This slide is here to be even more explicit that questions and comments are more than welcome, and that interactivity is good.
- Get my attention through any appropriate means, but if you’re throwing something, please lob, and keep it light.
- Though please consider the time we have available before you start on a long, involved anecdote of what once happened to a friend of yours.
About the Instructor

- John Sellens
- 25+ years as UNIX system administrator
- University of Waterloo, UUNET, managed services, FreshBooks, NightingaleMD . . .
- Long time USENIX and LISA attendee and speaker
  - And elsewhere too . . .
- Occasional writer and author

Notes:

- Feel free to contact me here or by email if you have any questions
Viewpoints and Religion

- I like simple
  - And like making my job easier, not harder
- Multiple cooperating component parts are good
- AKA The UNIX Philosophy
- Not too crazy about the bleeding edge
- Solve any problem in computer science with another level of indirection
  - But not too much of that today

Notes:

- With that viewpoint, generally I think Elasticsearch and friends are pretty cool
- I’m not generally a Java fanatic or anything like that, but these tools seem well implemented
The ELK Stack - Elasticsearch Logstash Kibana

- No master plan to take over the world?
- General need for scalable text search
  - Elasticsearch built on Lucene
  - Nice distributed, reliable database
- Hey! That might be a good place to collect log files!
- How about a convenient way to query the log data?
- Seems to have gained prominence fairly quickly

Notes:

- Or at least this is my impression of how things might have happened
  - I could be full of nonsense of course
The ELK Ecosystem

- Primary components are open-source
- Developers formed Elasticsearch the company
  - Services, support, add-ons
  - Upcoming more “enterprisey” tools
- Logstash and Kibana joined in
- Lots of people doing tools, docs, blogs, ...
- Starting to generate other products
  - e.g. Nagios Log Server

Notes:

- elasticsearch.org
- elasticsearch.com
  - Formed in 2012, seems well-funded
  - “Simple is best”
- Seems like a healthy environment to hitch your wagon to
- Standard docs and repositories for RPM-ish and APT-ish systems
One or Many

- ELK can be self-contained, on a single machine
  - As we shall see, laptop willing . . .
- Most components can be split to multiple machines
  - Elasticsearch clusters
  - Logstash shippers, brokers, indexers
- Some useful related parts are “missing”
  - e.g. Security and access controls

Notes:

- The standard packages install and start with a usable configuration
- Most people will want to do something more advanced than a single machine
  - But you can easily put a demo system together
- Can be installed and configured with configuration management tools
  - Puppet and the like
- Typically we run ELK on UNIX-ish servers
  - But it’s java, so it can run anywhere, right?
  - I think logstash can run on Windows, and grab from the eventlog
  - Though using something like nxlog on Windows might make more sense
Elasticsearch
What is Elasticsearch?

- Elasticsearch is a search engine
  - Distributed, scalable, resilient, HA
  - RESTful API, JSON, HTTP
  - Built on Apache Lucene

- Stores **documents**

- Organized by **type**

- In an **index**

Notes:

- The reference docs are
  - And worth a read (or a perusal, or ...)

- And the glossary
Documents, Types, Indexes

- Documents are JSON documents
  - So they can have some structure to them

- In RDBMS terms:
  - Index – database
  - Type – table
  - Document – row

- Documents have a document id
  - "indexname/type/id" is the unique identifier

- Documents can have version, TTL, parent/child

Notes:

- JSON – list of keyword : value pairs, plus more!
- More information about document attributes is in the documentation for the index API
How About That Index?

- An index can be created implicitly or explicitly
  - i.e. You can just start shoving documents in

- An index is divided into **shards**
  - Each shard is a lucene instance

- And may have replicas of the shards
  - Replicas for reliability and read performance

- Mappings give hints about data types of fields
  - To help make indexing and searching more efficient

- Aliases are like database views
  - Combine multiple indexes, select with a filter

Notes:

- The index API adds or updates a document to an index, and will automatically create an index when first used
- The create index API lets you explicitly create an index and set its attributes
- Mappings will be created automatically if not specified
  - A mapping is like a schema definition in a relational database
- More later on deciding on the number of shards and replicas
All About the API, Not the CLI

- At first, I was confused
  - Where are the administration commands?

- Everything is a RESTful API call
  - i.e. It's all through HTTP interaction

- Much admin-type stuff is done with curl or similar

- Lack of access controls make this much simpler
  - Which may or may not be a feature

```
curl 'http://es01:9200/
curl 'http://es01:9200/idxname/_status?pretty'
```

Notes:

- Hopefully, I’m less confused now
Planning Ahead

- It’s worthwhile to consider your environment
  - Will affect networking, configuration, shards, replicas . . .

- If a single machine will do it all, it’s easy

- Large index size may require more shards
  - A shard must fit on a single node

- Need many nodes for performance?

- Need replication to a failover location (or rack)?

- Need to control access or network traffic?
  - Need a private subnet?

Notes:

- Elasticsearch is happy to start up and get going
  - And the defaults may work just fine for some

- But for reliability and performance you should think ahead
  - e.g. You can’t add shards to an index
  - But you can change the number of replicas (up or down) dynamically
Installing Elasticsearch

- Basic install is straightforward
  - Use provided package repos
  - Needs java – suggest openjdk 1.7
  - Install package, optionally config, start service
- Default behaviour is nodes find each other, work together
  - cluster.name: elasticsearch
  - Multicast, 224.2.2.4, all interfaces
  - Other discovery methods are configurable
- Nodes listen for HTTP API calls on port 9200
- Cluster nodes communicate on port 9300

Notes:
- http://www.elasticsearch.org/overview/elkdownloads/
- Current is 1.4.4 (Feb 19 2015)
- See setting up repositories link on that page
- Installs into /usr/share/elasticsearch
  - On Centos 7 at least ...
- Discovery is called “zen discovery” – see the docs
- Once discovered, I think multicast traffic is not used further
Since I Mentioned “Cluster”

- Elasticsearch servers are called **nodes**
  - And choose a “random” name for themselves
- Nodes form a **cluster**
- And elect a master node
- And distribute shards across nodes
- And everything mostly “just works”
- Lots of configuration settings
  - Cluster settings, shard allocation and recovery
  - Can distribute shards based on node attributes

Notes:

- I think Elasticsearch takes available space into account when distributing shards
  - But making all nodes the same has a certain appeal
- Having nodes choose “random” names by default is kind of interesting
  - But I don’t understand why they didn’t use hostname
  - Perhaps on the assumption that it’s sometimes hard to be tidy with names, especially “in the cloud”
What’s In A Name?

- Consider naming before configuring or creating indexes
- You might want predictable cluster and node names
- Index names should be meaningful
  - Should you divide your data into multiple indexes?
  - Like data shards: users-a, users-b, users-c, ...
  - Index aliases can make multiple indexes look like one
- Time series indexes should use `name-yyyy.mm.dd`
  - Which matches Logstash and various tools

Notes:

- Long ago, we used to say that naming is always the hardest part
- These days, they say that choosing a colour for the bike shed is the hardest part
Configuration Files

- Two text files, YAML

- `elasticsearch.yml` – controls Elasticsearch
  - Controls node behaviour, clustering, defaults, recovery, etc.
  - Default file comes with lots of comments
  - And the defaults are quite reasonable

- `logging.yml` – defines how Elasticsearch logs
  - `log4j` – I think
  - Logs to local files by default

Notes:

- The default file says: “Most of the time, these defaults are just fine for running a production cluster.”

- You will of course manage these files (and everything else), with your configuration management tool of choice, right?
elasticsearch.yml Suggestions

- `cluster.name` should be unique
- `node.name` is more useful as FQDN
- If larger, booleans `node.master` and `node.data`
- Generic attribute e.g. `node.rack: rack2`
- Index defaults:
  - `index.number_of_shards: 5`
  - `index.number_of_replicas: 1`

Notes:

- One replica means the primary and a replica – two copies in all
- Generic attributes can be anything you want, and are useful in shard allocation awareness
  - e.g. Put each replica in a separate physical rack
Replicas vs RAID vs Backups

- Seem similar but actually different
  - RAID guards against disk failure
  - Replicas guard against node failure
  - Backups guard against software or human failure

- Replicas help with performance
  - And can be set for offsite replication

- With few nodes, RAID is useful
  - Disk failure has no network or CPU impact

- With large data, snapshots may help with backups
  - Or at least let you go back to before your error

Notes:

- Thinking of hardware rather than software RAID here
- Not sure that ZFS is a good idea, since it competes for CPU and memory
- Reasonably confident that NFS is not a great idea
- Certain that CIFS for Elasticsearch is a bad idea
- Snapshots – Elasticsearch or OS snapshots
- Of course, you must consider the consequences of data loss when planning your strategies
  - Me, I don’t care much if I lose some logs from my home network
Networks That Work

- Elasticsearch can run on your primary network
  - But by default, no access control
  - Can mitigate somewhat with host firewalls
- A separate subnet/VLAN is nice to have
  - Isolate shard/replication traffic
  - Easier to implement access controls
- A front-end load balancer is often useful
  - IP failover on node failure
  - Access control

Notes:

- A separate subnet/VLAN is more complicated when doing offsite replication of course
- I like nginx as a front-end load balancer
  - Possibly with a two nginx's and a virtual IP or two
- A load balancer is handy for management tools and we shall see later
Network Configuration

• Specify IPs to bind to
  – `network.bind_host` – for 9200 HTTP
  – `network.publish_host` – for 9300 Elasticsearch
  – or `network.host` – for both
  – Default is 0.0.0.0

• Can override default ports
  – But you likely don’t want to

• With subnet, load balancer provides the access

Notes:

• Network layout can influence or dictate the discovery method
Discovery (of the) Network

- By default, multicast, anyone can join cluster
  - `discovery.zen.ping.multicast.enabled`: true
  - A separate subnet provides some control

- `discovery.zen.ping.unicast.hosts`: ["host1","host2"]
  - Maintain a list of exactly who can join

- Plugins for discovery in AWS, Google, Azure

- `discovery.zen.minimum_master_nodes`: 1
  - Can the cluster run if something is missing?
**Avoiding Split-Brain**

- With a two node cluster, you run the risk of split-brain
  - Both nodes think they are the only master
- If you’re serious about a cluster
  - At least three possible master nodes
  - `minimum_master_nodes` set to more than half
  - So there can only be one quorum
- With two nodes, use a non-data node as third master
  - e.g. Your Logstash server
- With offsite replication, primary site might have all masters

**Notes:**

- For offsite, active/passive, configure the nodes in the primary location to be possible masters, since that is there the writing happens
- I don’t worry about split-brain on my home network, where the Elasticsearch servers are all the the same VirtualBox host
- And I don’t care about my home log data
- Your mileage may vary
**Shard Allocation Awareness**

- Can force replicas to be on different infrastructure
- Define appropriate generic attributes for each node
- Elasticsearch will try to spread replicas for safety

```yaml
node.rack: rack1
node.zone: east
cluster.routing.allocation.awareness.attributes:
  rack, zone
cluster.routing.allocation.awareness.force:
  .zone.values: east west
```

**Notes:**

- Relatively simple configuration, but seems to be powerful enough for many uses
Logging to Syslog

- I like Elasticsearch to log to syslog
- So Logstash can put Elasticsearch’s logs in Elasticsearch

rootLogger: INFO, console, file, syslog

syslog:
  type: syslog
  syslogHost: localsyslog
  facility: local0
  layout:
    type: pattern
    conversionPattern: \
      "elasticsearch[%dISO8601][%-5p][%-25c] %m%n"

Notes:

- Add syslog to rootLogger
- Add a new appender for syslog
- These per-language logging frameworks always confuse me
  - Why isn’t syslog the default?
Command, Control, Management
Commands and APIs

- Elasticsearch control is via HTTP API calls
  - Port 9200, on Elasticsearch nodes
- Hit a URL, get HTTP code and JSON back
  - Mostly GET, some PUT and POST
- No access control or authentication. None.
- For common tasks, you likely want a script wrapper
  - Or perhaps a real programming environment

Notes:

- We shall see methods for control later on
- And network layout helps somewhat
  - Might be required to be logged in to a node
Hey Node! Who Are You?

curl http://es01:9200/

{
  "status" : 200,
  "name" : "Numinus",
  "version" : {
    "number" : "1.3.4",
    "build_hash" : "a70f3cccb5220...c6597448eb3e45",
    "build_timestamp" : "2014-09-30T09:07:17Z",
    "build_snapshot" : false,
    "lucene_version" : "4.9"
  },
  "tagline" : "You Know, for Search"
}
And How About You, Cluster?

curl 'http://es01:9200/_cluster/health?pretty=true'

{
  "cluster_name" : "elasticsearch",
  "status" : "green",
  "timed_out" : false,
  "number_of_nodes" : 4,
  "number_of_data_nodes" : 3,
  "active_primary_shards" : 24,
  "active_shards" : 48,
  "relocating_shards" : 0,
  "initializing_shards" : 0,
  "unassigned_shards" : 0
}
Time For Only So Many Examples

- There are many, many API calls
- Manage cluster, nodes, indexes
- Insert, delete, update, query documents
- Elasticsearch plugins are through the web as well
  - Though some of those are more typical web pages
- Let's look at some of the APIs
- And some conventions

Notes:

- And seeing a million curl commands might not be that entertaining either
**Standard Options**

- An index reference can usually in a list, wildcard or _all
- Query options modify output format
  - `?pretty` – pretty-print the JSON
  - `?format=yaml` – return YAML
  - `?human=true` – human readable statistics values
- And more – see the docs

**Notes:**

- Multiple indices
- Common options
- You can also pretty-print JSON with: `python -mjson.tool`
Document APIs

- Index (insert), Get, Delete, Update
  - And multi-document versions

```
% s="http://es01:9200"
% curl -XPUT "$s/prod/users/1" \
   -d '{ "first" : "Bob", "last" : "Dobbs" }'
{"_index":"prod","_type":"users","_id":"1",
 "_version":1,"created":true}

% curl -XPOST "$s/prod/users/" \
   -d '{ "first" : "Jane", "last" : "Doe" }'
{"_index":"prod","_type":"users",
 "_id":"syRPUhWWRkuULqyYoll_Xw",
 "_version":1,"created":true}
```

Notes:

- Documents can have parent/child relationships, ttl (time to live), versions, etc.
Document APIs – cont’d

% curl "$s/prod/users/1" | json-pp
{
  "_id": "1",
  "_index": "prod",
  "_source": {
    "first": "Bob",
    "last": "Dobbs"
  },
  "_type": "users",
  "_version": 1,
  "found": true
}

% curl "$s/prod/users/1/_source" | json-pp
{
  "first": "Bob",
  "last": "Dobbs"
}

Notes:

- `json-pp` is an alias for `python -mjson.tool`
- With GET, optionally select fields, etc.
Search APIs

% curl -XPOST "${s/prod/_search?q=first:Bob"
% curl -XGET "${s/prod/_search" -d '{
  "query": {
    "query_string": {
      "query": "first:Bob AND last:Dobbs"
    }
  }
}'}

- And almost arbitrarily complex queries
- Search is "(near) real-time" – inserts are not instant

Notes:

Indices APIs

- Do various things to an index
- Create, delete, get (information), update (settings)
- Open/close – can make an index inactive
- Mapping management – index schema
- Alias management – like views
- Monitoring, status, stats, etc.
- Refresh – like flush updates
- Optimize – like disk defragmentation

Notes:

Cluster and Cat APIs

- Cluster health, state, statistics
- Update cluster settings
  - Some config settings are dynamic
- Node info, stats, shutdown
  - No node restart API call . . .
- Cat API – human readable, not JSON, output

% curl "$_s/_cat/shards?v"

```
index shard prirep state docs store ip node
users 4 p STARTED 5659 2mb 1.2.3.4 es01
users 4 r STARTED 5659 2mb 1.2.3.5 es02
```

Notes:

- I’m not sure that the cat API is really good for typing on the command line except for one-offs
  - A cover script might be nice
Elasticsearch Access Control

- Short answer: there is none
- Do you need some?
  - You might not, but most likely will
- Using a private network mitigates some of the risk
- Putting a gateway in front can help
  - Especially for ad-hoc and reporting access
- For some uses, nginx and basic auth can help
- Stunnel, firewall rules, etc.

Notes:

- There are commercial products coming available
- Shield: Security for Elasticsearch
  http://www.elasticsearch.com/products/shield/
- Nagios Log Server has access controls
  http://go.nagios.com/logserver
- Complicated by web tools that go direct from browser to Elasticsearch
- Turns out that adding controls after the fact is not quite as easy as you might hope
- But see notes on an Nginx front end on page 85
Monitoring and Management
Monitoring Elasticsearch

- Servers – all the typical things
  - CPU, disk space, memory, hardware
- Cluster health
  - Get /_cluster/health and look for “status”
  - Green, yellow, red – a human can look for why
- Index health
  - Get /indexname/_status and look at “_shards”
  - Any failed?
- Can get more details, but a dashboard might be useful

Notes:

- Simple Nagios checks are easy to wrap around a curl command and awk
Dashboard – Marvel

- Cluster health dashboard from Elasticsearch
  - Same toolkit as Kibana
  - Free for development use
- Really quite slick
- Needs to talk to Elasticsearch

% elasticsearch/bin/plugin -i \
elasticsearch/marvel/latest
% service elasticsearch restart

http://es01:9200/_plugin/marvel/

Notes:

- http://www.elasticsearch.org/overview/marvel/download/
- Talks directly from browser to Elasticsearch
  - So if you have nginx front-end, likely need to adjust URL in `config.js` file
  - Just like Kibana
Dashboard – elasticsearch-head

- Not as slick, but free
- Allows arbitrary calls to api through web . . .

% elasticsearch/bin/plugin \\n  -i mobz/elasticsearch-head

http://es01:9200/_plugin/head/

Notes:

- https://github.com/mobz/elasticsearch-head
- http://mobz.github.io/elasticsearch-head
Dashboard – Elasticsearch Paramedic

- Another free dashboard
- And you can try it online with no install
  - Enter your Elasticsearch URL from your browser’s point of view
  - I had trouble getting the demo to work well

```bash
% elasticsearch/bin/plugin \
  -i karmi/elasticsearch-paramedic

http://es01:9200/_plugin/paramedic/
```

Notes:

- [https://github.com/karmi/elasticsearch-paramedic](https://github.com/karmi/elasticsearch-paramedic)
What is Logstash?

- Logstash filters inputs to outputs
  - Connect “this” to “that” or “these” to “those”
- Takes input **message** and might translate it with **codec**
- Adds some fields – timestamp, host
- Might transform or add fields with filters
- Produces output, possibly via a codec
- Works very well with log files and Elasticsearch
  - But can do almost arbitrary things

Notes:

- Well, “arbitrary things” perhaps over states it
- www.logstash.net/docs/1.4.2/tutorials/getting-started-with-logstash
Sample Simple Logstash Config

```
input {
    stdin { type => "echo" }
}
filter {
    if [type] == "echo" {
        grok {
            match => {
                "message" => "(?<greeting>(hello|bye))"
            }
        }
    }
}
output {
    stdout { codec => json }
}
```

Notes:

- Simple text file(s)
- Hash sign comment indicator
Sample Simple Logstash Execution

```
% echo hello \
    | logstash -f sample.conf \ 
    | json-pp

{
    "@timestamp": "2014-11-08T05:27:29.550Z",
    "@version": "1",
    "greeting": "hello",
    "host": "ls01.t0.syonex.com",
    "message": "hello",
    "type": "echo"
}
```
Installing Logstash

• Basic install is straightforward
  – Use provided package repos
  – Needs java – suggest openjdk 1.7
  – Typically “logstash” and “logstash-contrib” packages

• Create configuration file(s)
  – Usually a directory of files

• Start the service
  – Runs as user logstash

Notes:

• http://www.elasticsearch.org/overview/elkdwnloads/

• Current is 1.4.2, 1.5.beta1 December 2014

• Information on APT and YUM repositories
  http://www.logstash.net/docs/1.4.2/repositories

• Actually written in JRuby but distributed as jar files

• Installs into /opt/logstash
  – Again, on Centos 7 at least

• A directory of configuration files is read in “alphabetical” order
  – Perhaps that’s dependent on locale and/or language settings
Logstash Connectivity Overview

- Inputs, filters and outputs – directed acyclic graph?
  - Inputs and outputs are edges
  - Filters and other services are nodes
  - Or it's all just Tintertoy ...
- Lets you connect various sources and sinks
  - A crossbar switch?
- Inputs and outputs provide various protocols
- Codecs provide translation

Notes:

- Start with a single input and single output
- Then expand the connections into a network
- Inputs/outputs can implement secure and reliable transport
  - e.g. Lumberjack is SSL between logstash instances
- A little glossary
  - Plugin – an input, filter, or output method
  - Setting – a plugin configuration parameter
  - Value – setting values can have specific types; string, boolean, etc.
  - Codec – encodes data in an input or output
Organizing Config Files

- Use a directory that gets glob’d
  - Usually want inputs, then filters, then outputs
  - Consider a numbering/naming scheme to make ordering obvious

- Text files, random-ish whitespace, hash-sign comments

- Test your configs before restarting
  - No config reload, only restart

```
logstash -f /etc/logstash/conf.d --configtest
```

Notes:

- There are likely times when you might want to violate the inputs / filters / outputs ordering
  - But that would likely be complicated and/or convoluted

- You can run multiple copies of logstash on one machine
  - Which can make things more comprehensible
  - At the cost of some memory

- Restarting for new configs could disrupt processing
  - Multiple servers, or a message broker might help
Quick Configuration Notes

- Three sections – input, filter, output – can be interleaved
  - Applied to an event in config file order
- Lots of plugins, lots of settings
  - String, boolean, number, array, hash
- Refer to field values with [fieldname]
  - In strings with "sprintf format"
  - path =>
    "/var/log/%{type}.%+yyyy.MM.dd.HH"
- if / then / else – with curly braces
- “The Logstash config language aims to be simple"

Notes:

- www.logstash.net/docs/1.4.2/configuration
- With nested fields, full square bracket path is required
- Date format is not strftime(3) style but something more java-ish
- No looping statements, no case, etc.
Codecs – Inline Filters

- Codecs convert data on input or output
  - Effectively a filter attached to an input or output
- Input codecs convert input to fields
  - Which can then be used in filters
- Output codecs convert the internal format into external format
  - i.e. the format whatever you’re sending to expects

Notes:

- Standard codecs are listed on the main docs page
  www.logstash.net/docs/1.4.2/
**Codecs – A Few Examples**

- **plain** – line of input becomes the message field
  - Can also convert character sets, or parse with sprintf format

- **json** – read a JSON document, convert to fields
  - On output, print fields as a one line JSON document

- **multiline** – combine multi-line messages into a single event
  - Patterns, join previous/next

- **graphite** – handle graphite metrics

- **collectd, netflow, . . .**

**Notes:**

- There are lots of codecs – check the docs
- I’ll mention just a few of the most interesting/useful
- The json codec can also change characters sets
- multiline – e.g. those huge java stack traces
Inputs – Getting Stuff Into Logstash

- Define input to collect events from a data stream
- Lots of different input plugins (methods)
  - e.g. file, syslog, stdin, tcp, . . .
- Typically set the “type” field on an input, or set “tags” array
  - Filters can use the type to know where the event came from
- Codec converts to internal format and fields
- Can declare many inputs
- Can use the same plugin multiple times

Notes:

- If type or tags is already set, the value is not changed
  - If we don’t already know any better, it’s this
- I don’t know if the field name “type” is special, or just a convention
- Similar inputs, as long as the settings are not exactly the same
  - For example, it would likely be an error, or at least ill-advised, to have two inputs reading the same file
Inputs – A Few Particular Ones

- file – like `tail -F` on a file or glob pattern
  - Excludes, other options
- syslog – act as a syslog server
- stdin, pipe, exec – exec runs command periodically
- tcp, udp, unix – listen on a socket
- imap – reads mail via IMAP and creates events
- irc, twitter, xmpp
- redis, rabbitmq, zeromq – read from queues
- snmptrap, collectd, graphite, . . .

Notes:

- There are all sorts of useful settings for inputs
- And a bunch more I didn’t mention
- I think adding a new input involves writing some Ruby
  - And of course there’s always the pipe input
  - So you can write in language of your choice
Filter Facts

- Filters modify events – add/modify fields, etc.
- Usually invoked based on type or tags
- Parse messages, set fields based on values, etc.
- if/then/else and config file ordering dictate behaviour
- Wide variety available
  - Though I don’t think there’s “pipe”

Notes:
- Again, typically Ruby code
Some Notable Filters

- anonymize – replace field values with their hash
- csv – split CSV field into separate fields
- date – parse various data formats
- dns – lookup name from IP address
- drop – delete an event
- geoip – add geographic info based on IP address
- grep – modify based on regular expressions
- metrics – count matching events over interval

Notes:

- There are many more . . .
grok – Pattern Match

- Pattern match a field and act
  - Whole library of patterns included, extensible
- Create new fields, remove, drop, etc.
- "%{IP:client}" – match IP address, save as “client”
- Standard use case: parse entire line
  - e.g. Split an apache log file entry into fields
- Arbitrary regexp: (?<newfield>regexp)
- If fails, sets tag “_grokparsefailure”

Notes:

- Regular expressions are Origuruma syntax
  - I think this is now the default Ruby style
- Online tools for regular expression debugging
  - Rubular http://rubular.com/
  - Grok debugger http://grokdebug.herokuapp.com/
- e.g. Test if grok succeeds
  if ! ( "_grokparsefailure" in [tags] ) {
    # do something
  }
- www.logstash.net/docs/1.4.2/filters/grok
Some Sample Standard Patterns

USERNAME [a-zA-Z0-9._-]+  
USER %{USERNAME}  

INT (?:+)?(?:[0-9]+)  

WORD \b\w+\b  
NOTSPACE \s+  
SPACE \s*  

MAC (?:%{CISCOMAC}|%{WINDOWSMAC}|%{COMMONMAC})  
CISCOMAC (?:(?:[A-Fa-f0-9]{4}\.){2}[A-Fa-f0-9]{4})  
WINDOWSMAC (?:(?:[A-Fa-f0-9]{2}-)\{5\}[A-Fa-f0-9]{2} \{2\})  
COMMONMAC (?:(?:[A-Fa-f0-9]\{2}\:\{5\}[A-Fa-f0-9]\{2\})

Notes:

- Extracted from patterns/grok-patterns
- I tried to find ones that would fit easily, but …
mutate – General Field Modifications

- Add/remove/change fields in various ways
  - e.g. Add new field with value based on other fields
- gsub – regexp search and replace
- split / join arrays
- strip / uppercase / lower – modify strings
- Can mutate inside if / then /else

Notes:

- www.logstash.net/docs/1.4.2/filters/mutate
Sample syslog Input and Filter

```ruby
input {
  syslog {
    type => syslog
    port => 5514
  }
}
filter {
  mutate {
    add_field => [ "hostip", "%{host}" ]
  }
  dns {
    reverse => [ "host" ]
    action => replace
  }
}
```

Notes:

- Configure syslogd to forward syslog to port 5514 on logstash server
- Note that logstash does not (typically) run as root, so can’t typically bind to the standard port 514
Sample lumberjack Input and apache Filter

```yaml
input {
  lumberjack {
    port => 5043
    ssl_certificate => "/keypath/ls-fwd.crt"
    ssl_key => "/keypath/ls-fwd.key"
    type => "lumberlogs"
  }
}

filter {
  if [type] == "apache" {
    grok {
      match => {
        "message" => "%(COMBINEDAPACHELOG)"
      }
    }
  }
}
```

Notes:

- In this case logstash-forwarder sets the type to apache before sending
Outputs – Getting Stuff Out of Logstash

- What good are events if you can’t use them?
- Filters can do data mining, notice exceptions
- There are many output plugins
- Most common/recommended is elasticsearch
- file, pipe, stdout, tcp, udp
- nagios, email, xmpp, irc, syslog, pagerduty, redmine
- redis, rabbitmq, lumberjack
- Altering/exceptions likely fire on tags
  - e.g. A filter could set the “problem” tag

Notes:

- Many of these are not meant to be “high volume” outputs
Elasticsearch Output

- Lots of settings
  - e.g. host, port, index

- Logstash can join an Elasticsearch cluster
  - Set protocol to “node”
  - Can set node_name, cluster

- Also has an embedded Elasticsearch
  - Likely only useful for very small systems

- Can even delete by document_id

Notes:

- Joining the cluster and using “node” protocol is likely the most resilient
Elasticsearch Output is Simple

```ruby
output {
  elasticsearch {
    node => "ls01"
    cluster => "prod"
    protocol => "node"
  }
}
```
Shippers, Brokers, and Topology

- Hook things together in a graph
  - If it’s useful in your environment . . .
- Shipper – logstash that collects stuff and passes to
- Broker – buffer/cache e.g. redis. AMQP, 0MQ, etc.
  - Optional, but adds scalability and resilience
- Indexer – takes from broker and inserts into elasticsearch
  - Or any other output . . .
- Gets you across networks, handle local/remote data

Notes:

- Shippers don’t strictly have to be logstash
  - syslog acts as a shipper to a port on a logstash server
- Redis seems to be the most commonly used broker
  - Ease of use – typically use a redis “list” (a queue)
  - Files and rsync could be used as a broker
- Different levels of authentication, encryption, access control for different tools
  - Connecting edge could be stunnel, etc.
- UNIX philosophy – tie different tools together easily
Stolen from The Logstash Book
Stolen from The Logstash Book
Add Ons for Logstash
Kibana Logstash Dashboard

- Web, javascript interface for Logstash data in Elasticsearch
  - Nice eye-candy for convincing the unconvinced
- Initial web server load, then javascript direct to Elasticsearch
  - Proxying, control a little more challenging
- Can also run on port 9292 with `logstash web`
- If browser can't connect to Elasticsearch, it says Upgrade Required Your version of Elasticsearch is too old ...
- Save custom dashboards, searches, etc.

Notes:

- Kibana 4 is “all new” February 19, 2015
- It tries to do the right thing, but ...
- For proxy, or different access, modify
  `/opt/logstash/vendor/kibana/config.js` to set `elasticsearch:` to the appropriate path
- For proxying, adding `window.location.port` helps e.g.
  `elasticsearch: window.location.protocol+"//"+window.location.hostname +(window.location.port !== "" ? ":"+window.location.port : ")`,
### Logstash Forwarder

- A small, low footprint, limited Logstash
  - Intended as a shipper when full Logstash not needed
  - Written in Go, rather than Java; self-contained
- Does file and stdin inputs, lumberjack output
- Different configuration syntax than Logstash
- Uses SSL to talk, so some control
- Can configure multiple Logstash servers
  - Will choose, and failover if necessary

---

**Notes:**

- Formerly called “Lumberjack”, which lives on as the protocol name
- Now part of the Elasticsearch family of fine products
  [https://github.com/elasticsearch/logstash-forwarder](https://github.com/elasticsearch/logstash-forwarder)
Sample Logstash Forwarder Config

```
{
  "network": {
    "servers": [ "ls01:5043" ],
    "timeout": 15,
    "ssl ca": "/keypath/ls-fwd.crt",
  },
  "files": [ {
    "paths": [ "/var/apache/access.*" ],
    "fields": { "type": "apache" }
  }, {
    "paths": [ "/var/nagios/nagios.log" ],
    "fields": { "type": "nagios" }
  } ]
}
```
**Curator – Manage Time-Series Indices**

- Logstash creates date-stamped Elasticsearch indices
- Curator helps manage that
  - Delete, close, optimize, ...

  ```
  % curator delete --older-than 30
  % curator delete --disk-space 1024
  ```

**Notes:**

- Now part of the official Elasticsearch family
  - [https://github.com/elasticsearch/curator](https://github.com/elasticsearch/curator)
  - [https://github.com/elasticsearch/curator/wiki/Examples](https://github.com/elasticsearch/curator/wiki/Examples)
- A nice explanation is here
  - Nice explanation
- Can be used with any `something-yyyy.mm.dd`
  - Prefix defaults to “logstash-”
- Easy to install
  ```
  # yum -y install epel-release
  # yum -y install python-pip
  # pip install elasticsearch-curador
  ```
Monitoring and Management
Monitoring Logstash

- All the usual stuff for a server
- Monitor Elasticsearch for logstash node
- Is the process running?
- Monitor the pipeline
  - Generate syslog message from cron
  - Have logstash submit passive result to Nagios
- Remember that a config change restarts logstash
Nginx Front End

- It's useful to have all kibana web access through nginx

- Basic idea
  - Standard port 80 for basic kibana
  - Tell kibana to use http://ls01/es
  - Use rewrite and proxy_pass for Elasticsearch

- Can require some iterative testing

- Can add basic auth
  - As long as it's all on the same web server

Notes:

- These are quick sample examples

- Your mileage may vary

- See Securing kibana + elasticsearch
  http://tom.meinlschmidt.org/2014/05/19/securing-kibana-elasticsearch/
  for a nice nginx recipe

- You can also limit what things are public and what require auth
  http://www.ragingcomputer.com/2014/02/securing-elasticsearch-kibana-with-nginx
  e.g. nginx limit except GET
nginx Basic Config

# logstash kibana nginx fragment
server {
  listen * :80 default_server;

  location / {
    root /var/www;
    index index.html index.htm;
  }

  location /kibana {
    # auth_basic "Restricted";
    # auth_basic_user_file /etc/nginx/htpasswd;
    root /opt/logstash/vendor;
  }
}

Notes:

- Makes logstash kibana available
Elasticsearch, Logstash, and Other Data

Notes:

- Makes Elasticsearch available under /es
location /_plugin {
    # auth_basic "Restricted - ES";
    # auth_basic_user_file /etc/nginx/htpasswd;

    # set some headers
    proxy_http_version 1.1;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Host $http_host;

    # had to disable selinux for this to work
    proxy_pass http://es01.example.com:9200;
}

Notes:

- Makes logstash _plugin directory available
Wrap Up
Summary

• We’ve tried to hit the key areas

• We didn’t cover everything
  – Lots of choices
  – Especially for inputs / filters / outputs

• Hopefully you’ve learned some of the more interesting aspects

• And can apply them in your own implementations
Where to Get ELK Help

• The documentation is fairly good
  – The “guide” is the starting point
  – Many separate web pages for API calls, etc.

• Mailing list, IRC, videos and talks

• And of course Elasticsearch the company
  – Would be happy to provide services

• The Logstash Book

Notes:

• http://www.elasticsearch.org/guide/
  – The new starting point for documentation for ELK

• http://www.elasticsearch.org/resources/

• http://www.elasticsearch.org/community/

• http://logstash.net/docs/1.4.2/

• The Logstash Book is not bad – not very deep, but an easy read, and a good introduction, and the price is right
  http://www.logstashbook.com/
And Finally!

- Feel free to contact me directly if you have any unanswered questions, either now, or later: jsellens@syonex.com

- Questions? Comments?

- Thank you for attending!

Notes:

- Thank you very much for taking this tutorial, and I hope that it was (and will be) informative and useful for you.

- I would be very interested in your feedback, positive or negative, and suggestions for additional things to include in future versions of this tutorial, on the comment form, here at the conference, or later by email.