splunk> .conf2017

How splunkd works

splunkd: Pipelines, Processors, Queues

Inputs: File, Network, Script, HEC, S2S, …

Debugging: Metrics, Monitoring Console

by Amrit Bath, Abhinav Nekkanti

Forward-Looking Statements

During the course of this presentation, we may make forward-looking statements regarding future events or the expected performance of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results could differ materially. For important factors that may cause actual results to differ from those contained in our forward-looking statements, please review our filings with the SEC.

The forward-looking statements made in this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, this presentation may not contain current or accurate information. We do not assume any obligation to update any forward looking statements we may make. In addition, any information about our roadmap outlines our general product direction and is subject to change at any time without notice. It is for informational purposes only and shall not be incorporated into any contract or other commitment. Splunk undertakes no obligation either to develop the features or functionality described or to include any such feature or functionality in a future release.

splunk> .conf2017

Agenda

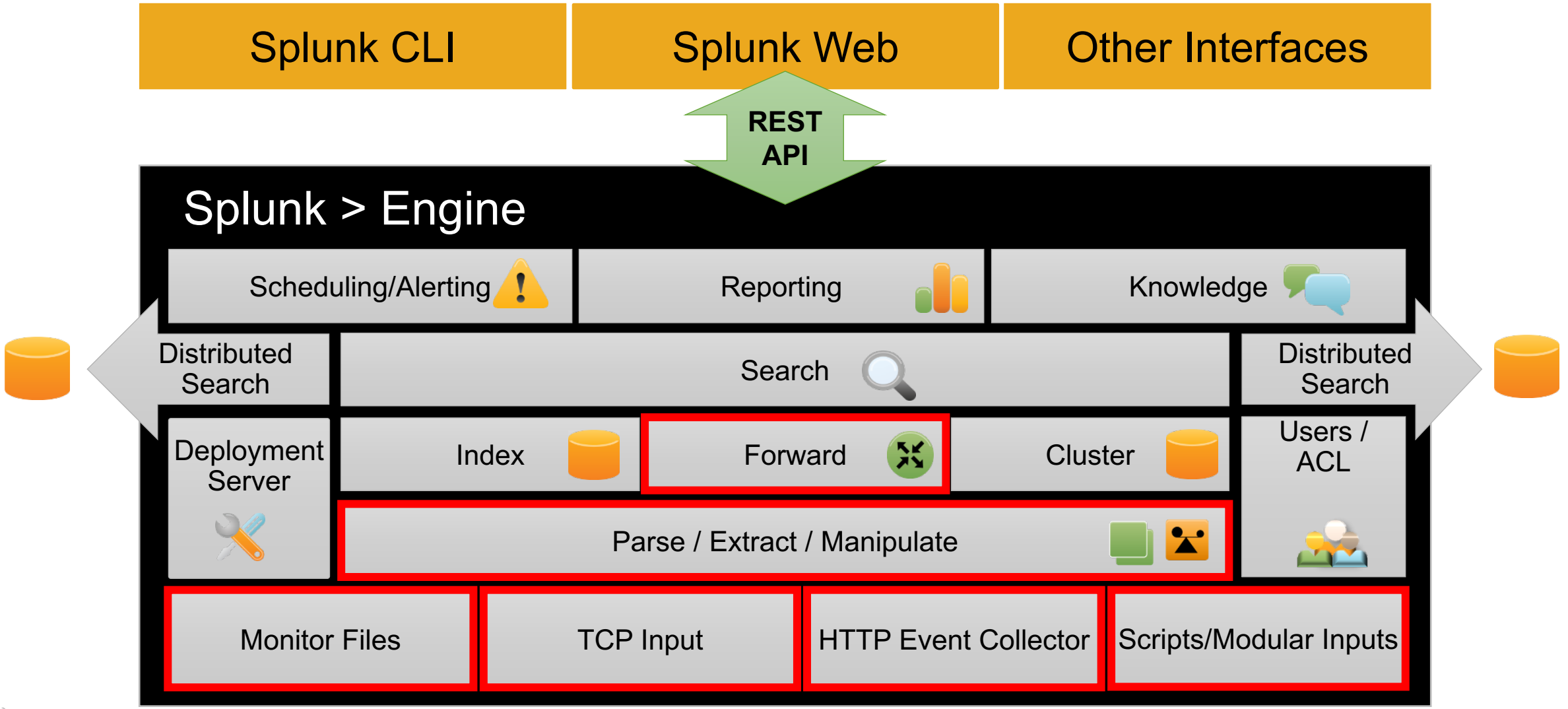
- splunkd building blocks:
  - Pipelines, processors, queues
  - Not here: Indexing/clustering, searching.
- Where data goes & how
- Where data comes from
- Debugging/optimizing
  (What Would Octavio Do?)

Visual approximation of Octavio

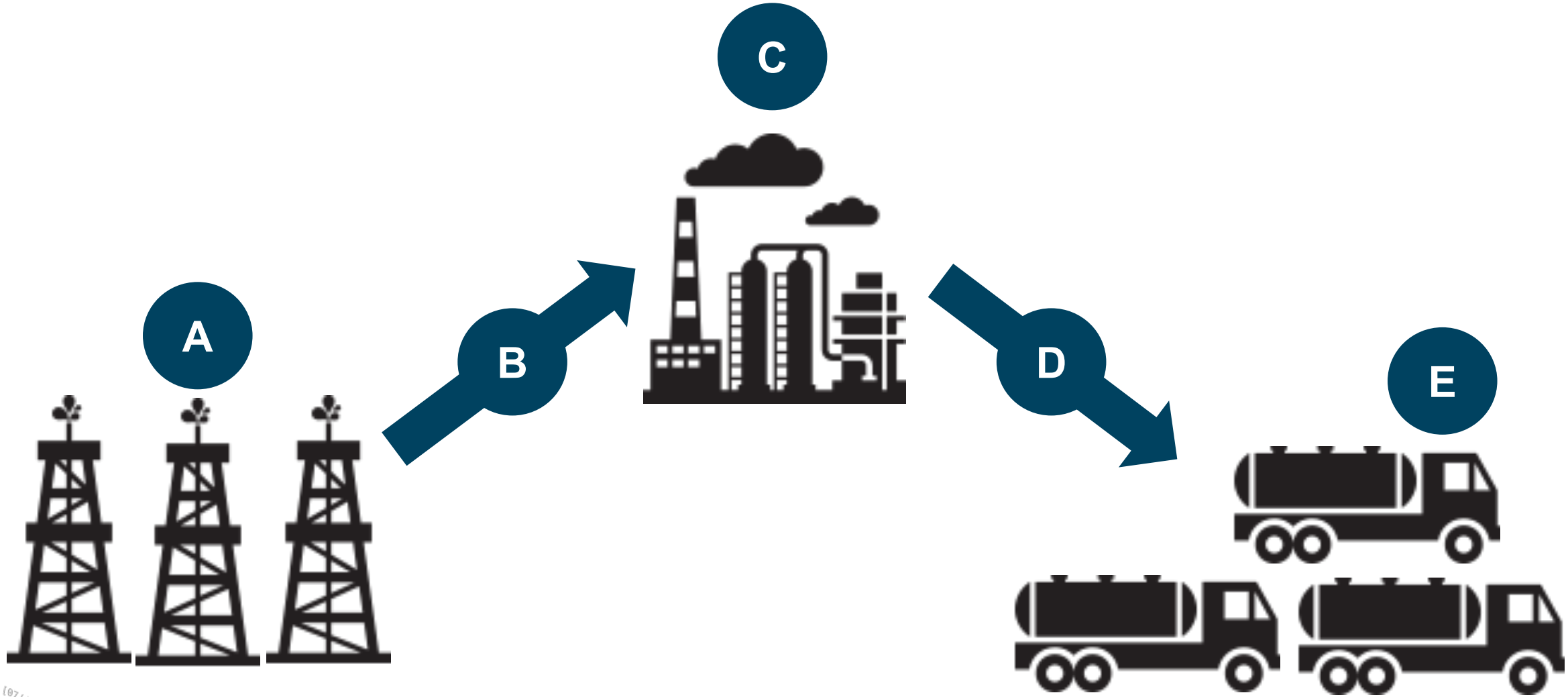
About Us

- Amrit Bath (2005) – Engineer
  - CLI, Deployment Server, Tailing, REST API, Universal Forwarder, Indexed Extractions, Cloud, SHC, Metrics, …
  - Previously: College
  - Hates working on cars
  - Owns Bud Light pants
- Abhinav Nekkanti (2013) – Engineer
  - Tailing, Pipeline Parallelization, Alerts improvements, Preview improvements, Cloud
  - Previously: Citrix (Goto products)
  - Size 14 shoe
  - Overly enthusiastic about Bud Light tallboys

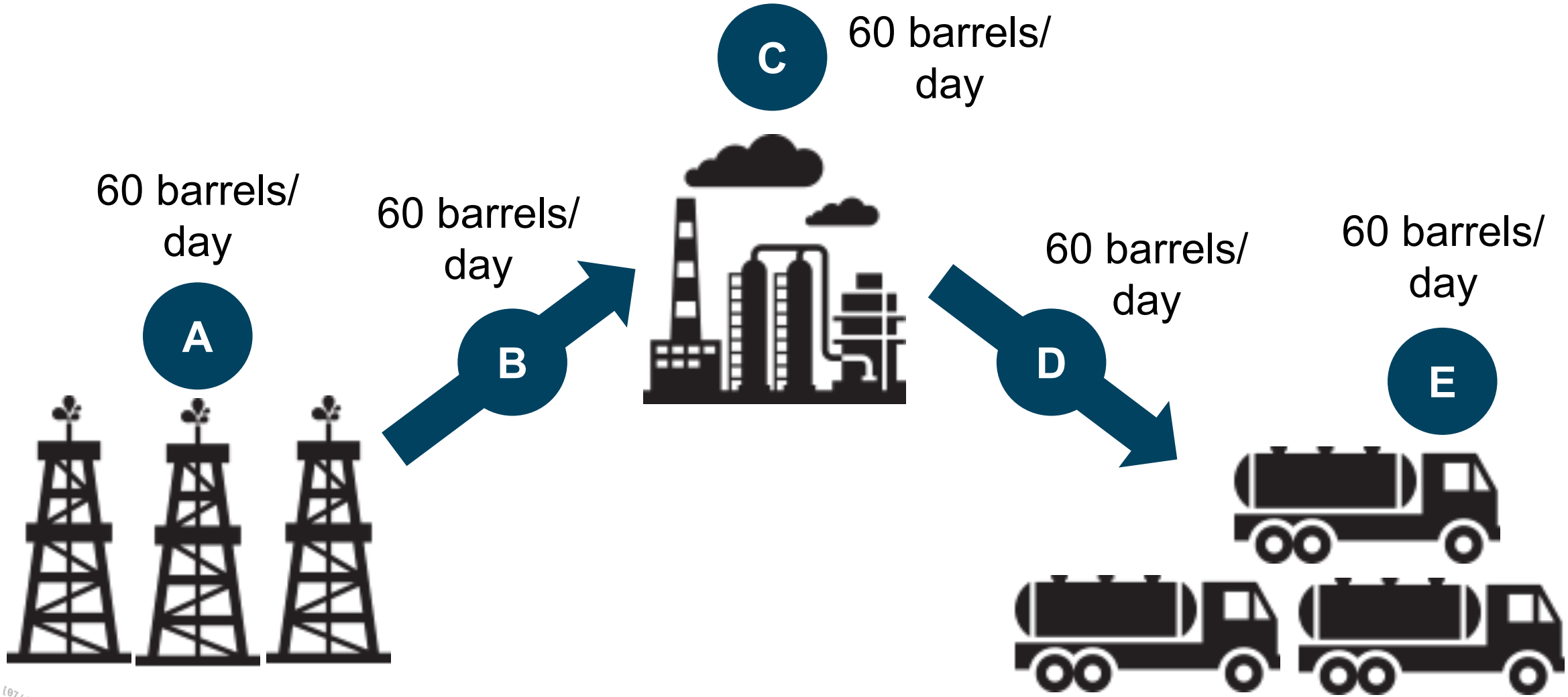
Splunk Architecture



It’s all a pipeline

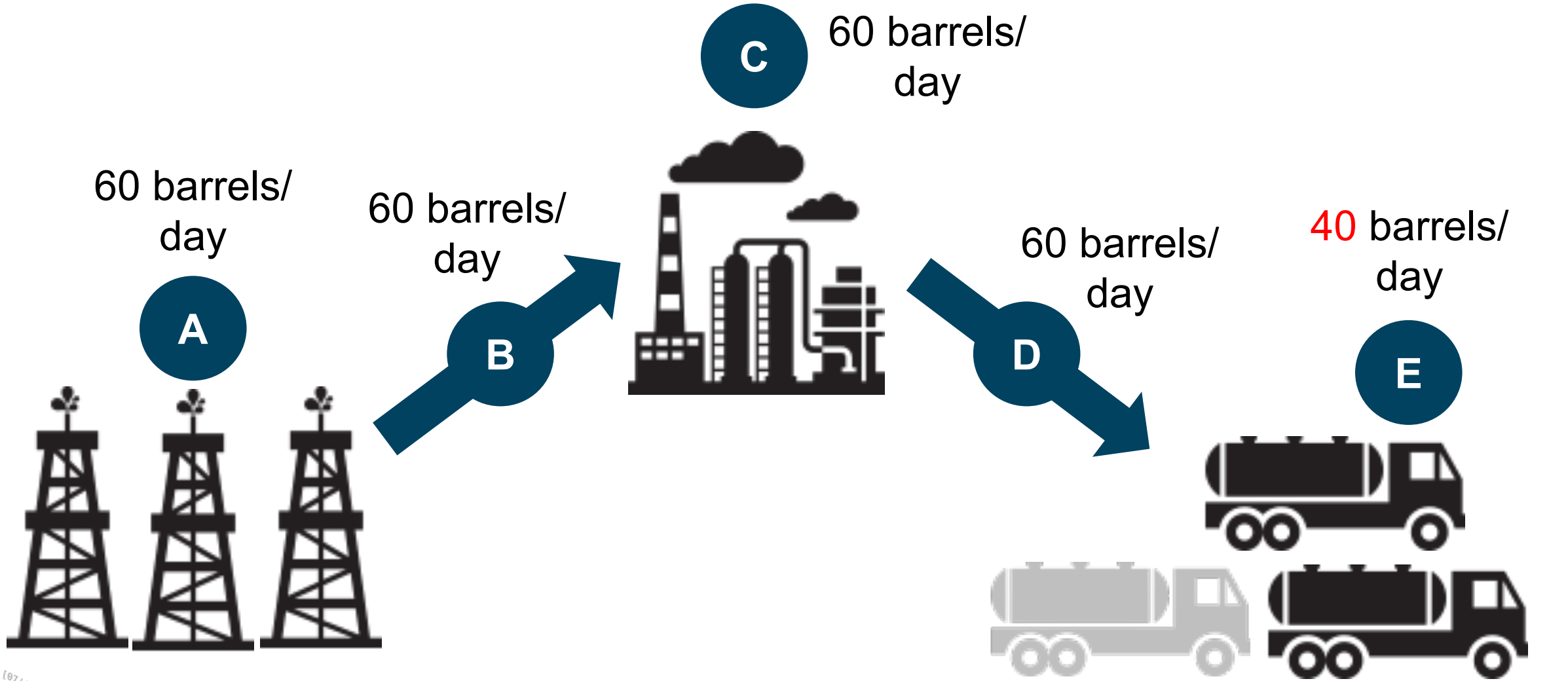


It’s all a pipeline

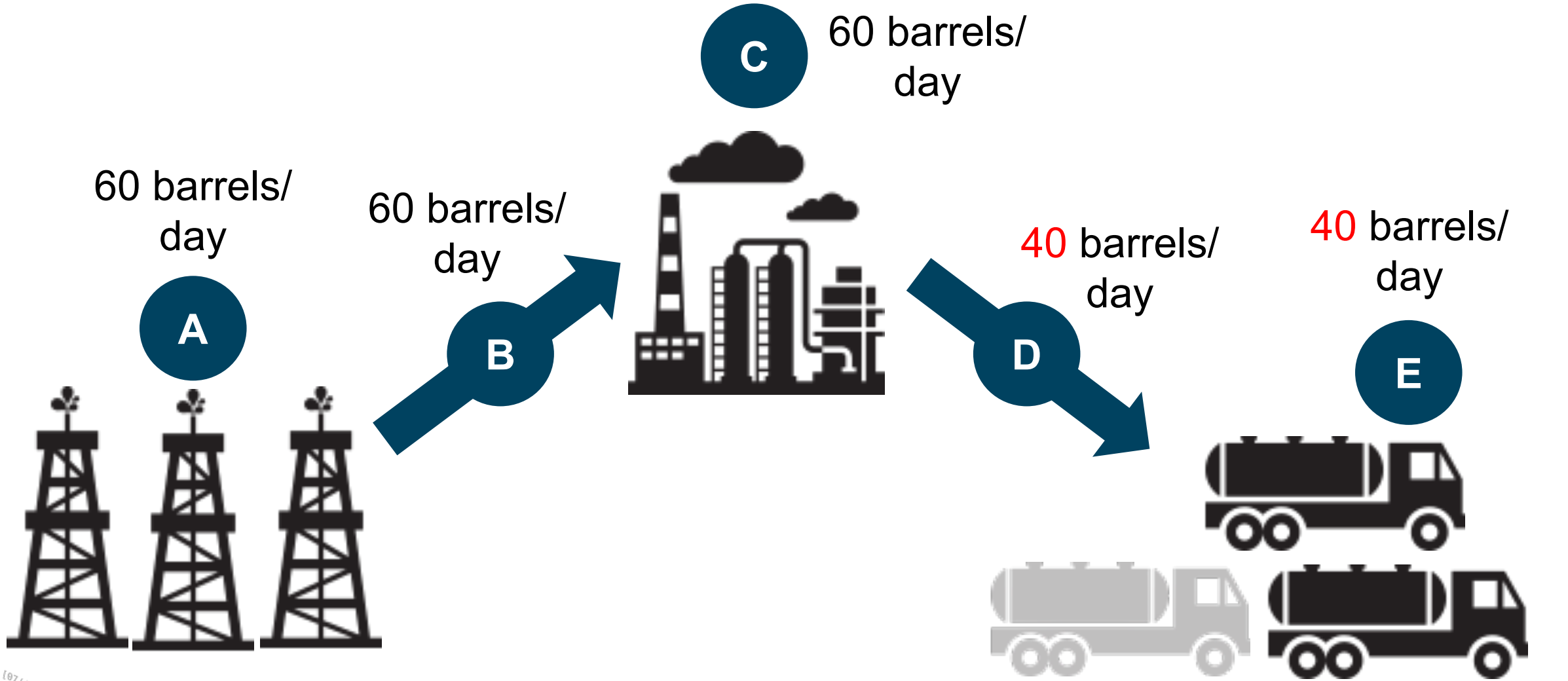


splunk> .conf2017

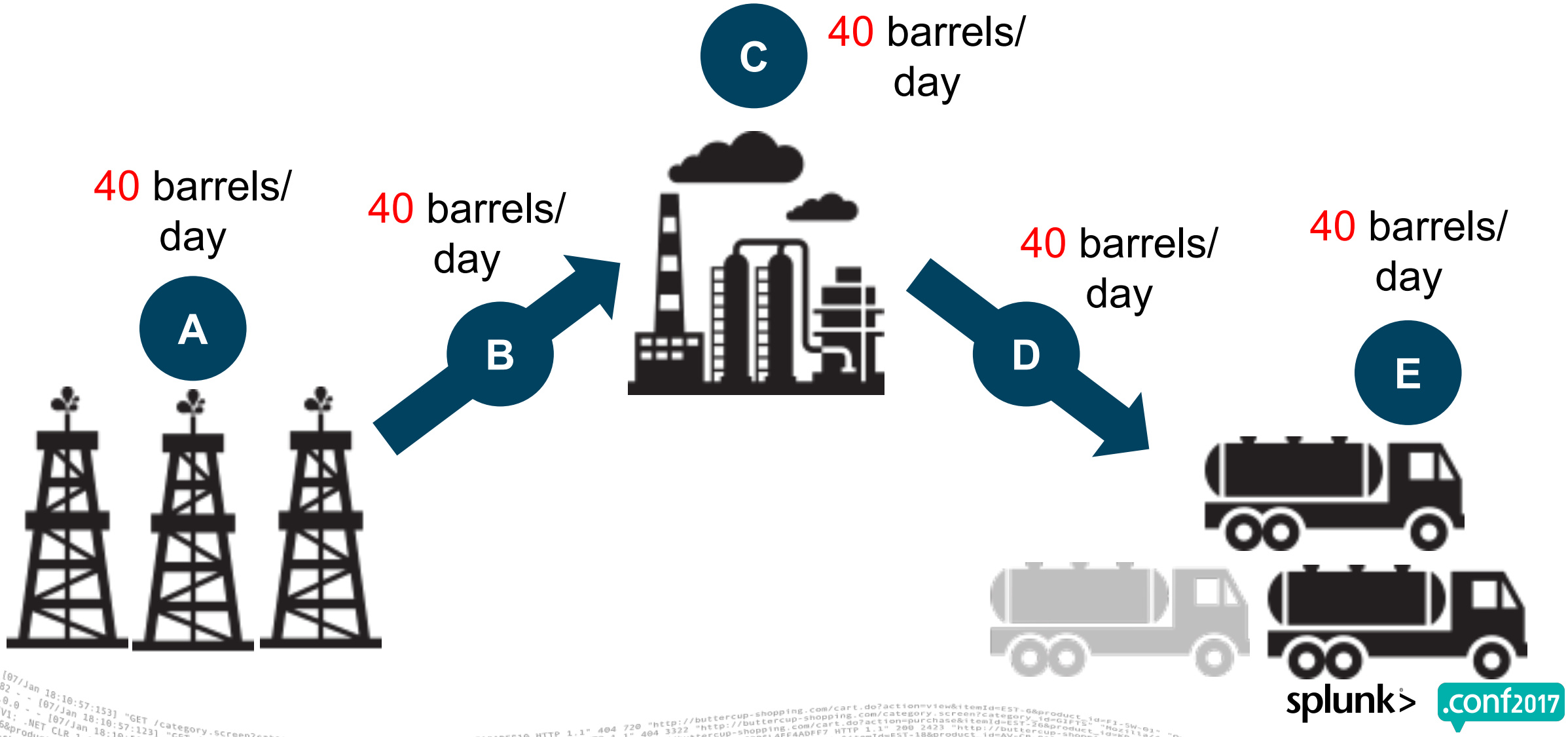
It’s all a pipeline



It’s all a pipeline

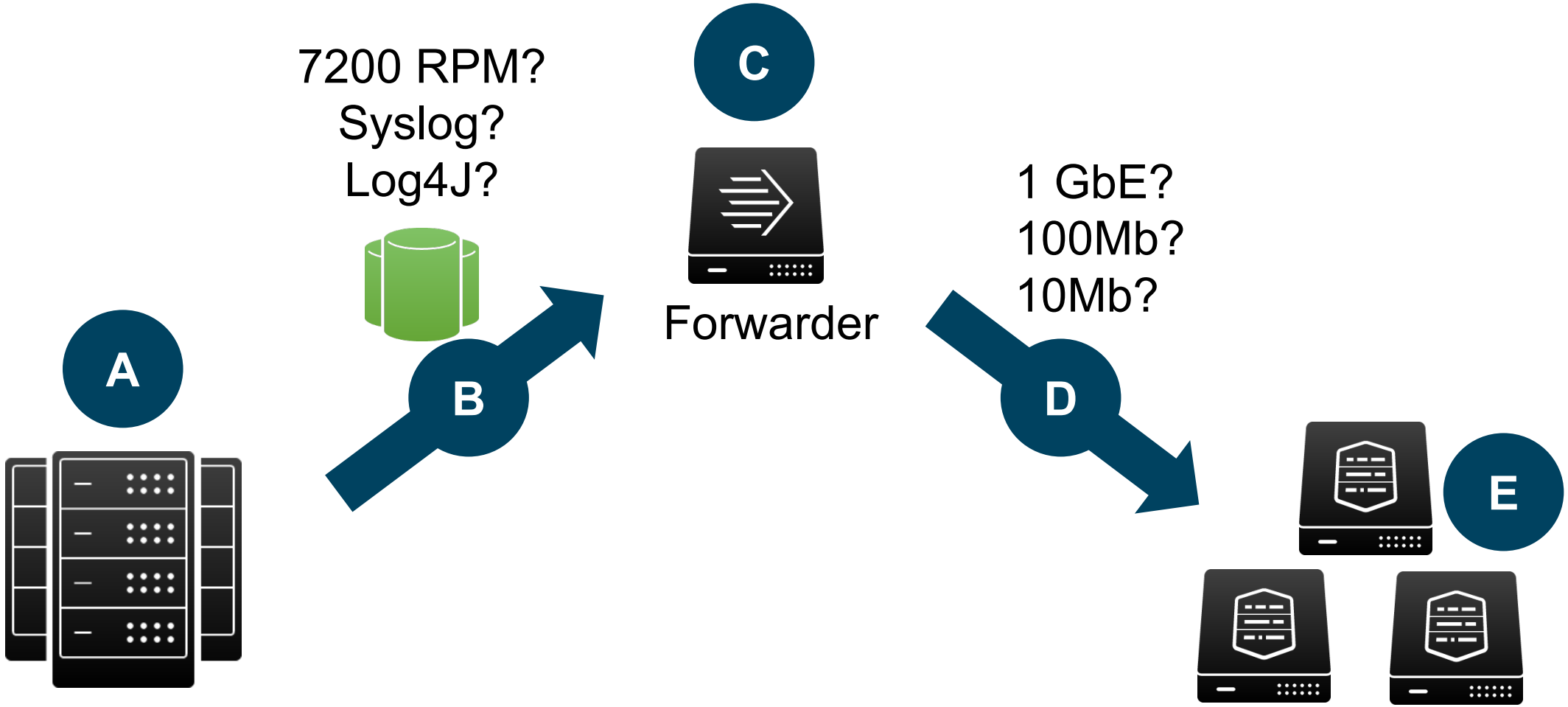


It’s all a [smaller] pipeline

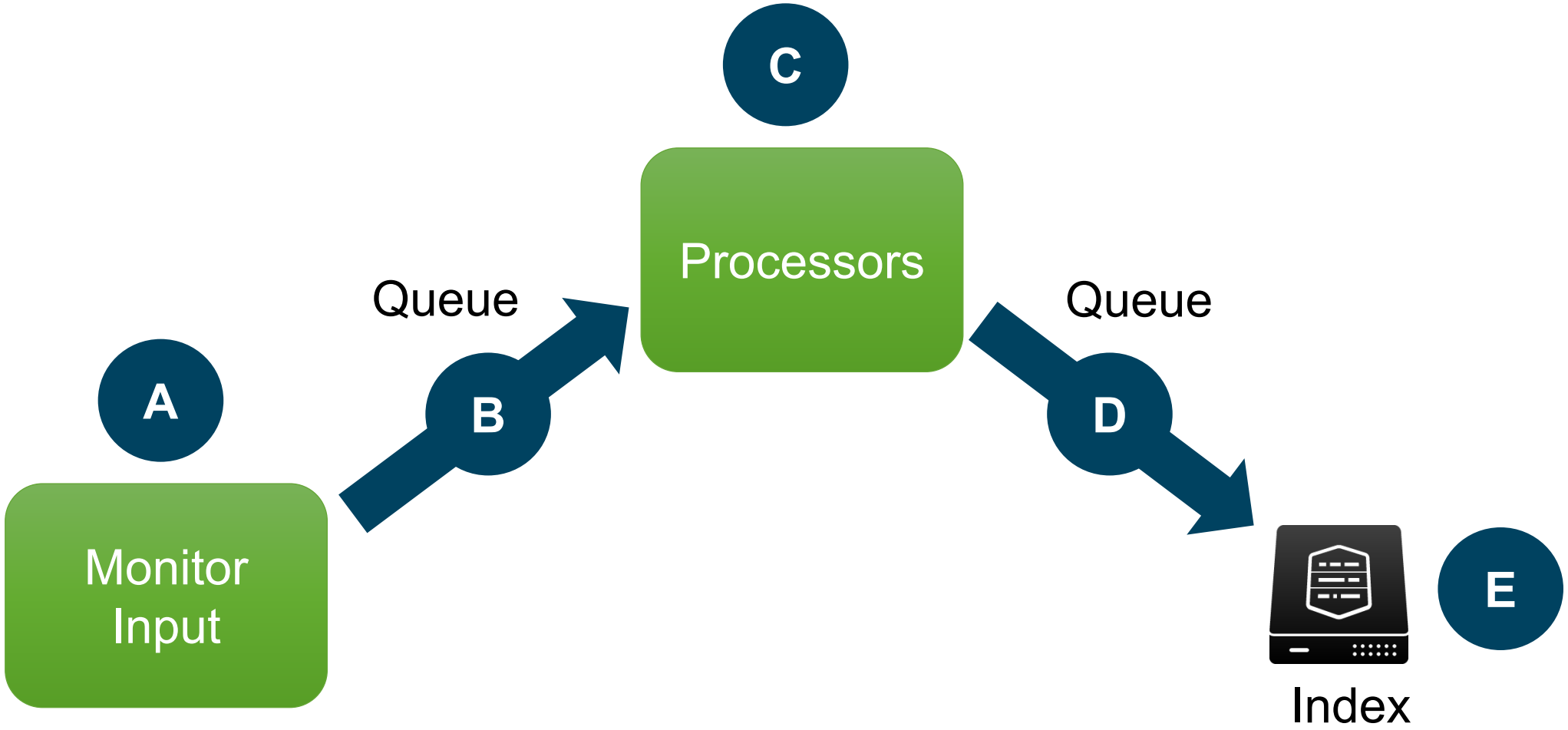


splunk> .conf2017

Really

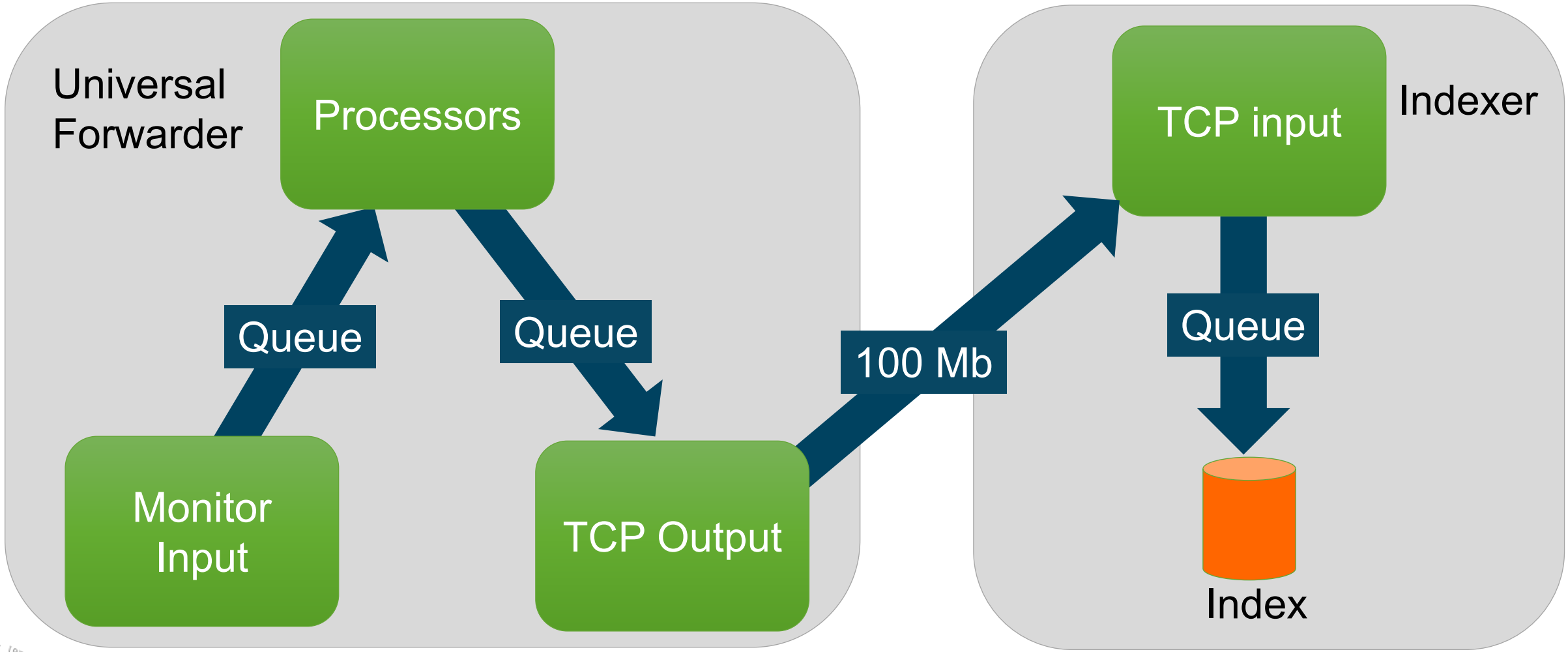


Internally, too!



And across multiple instances!

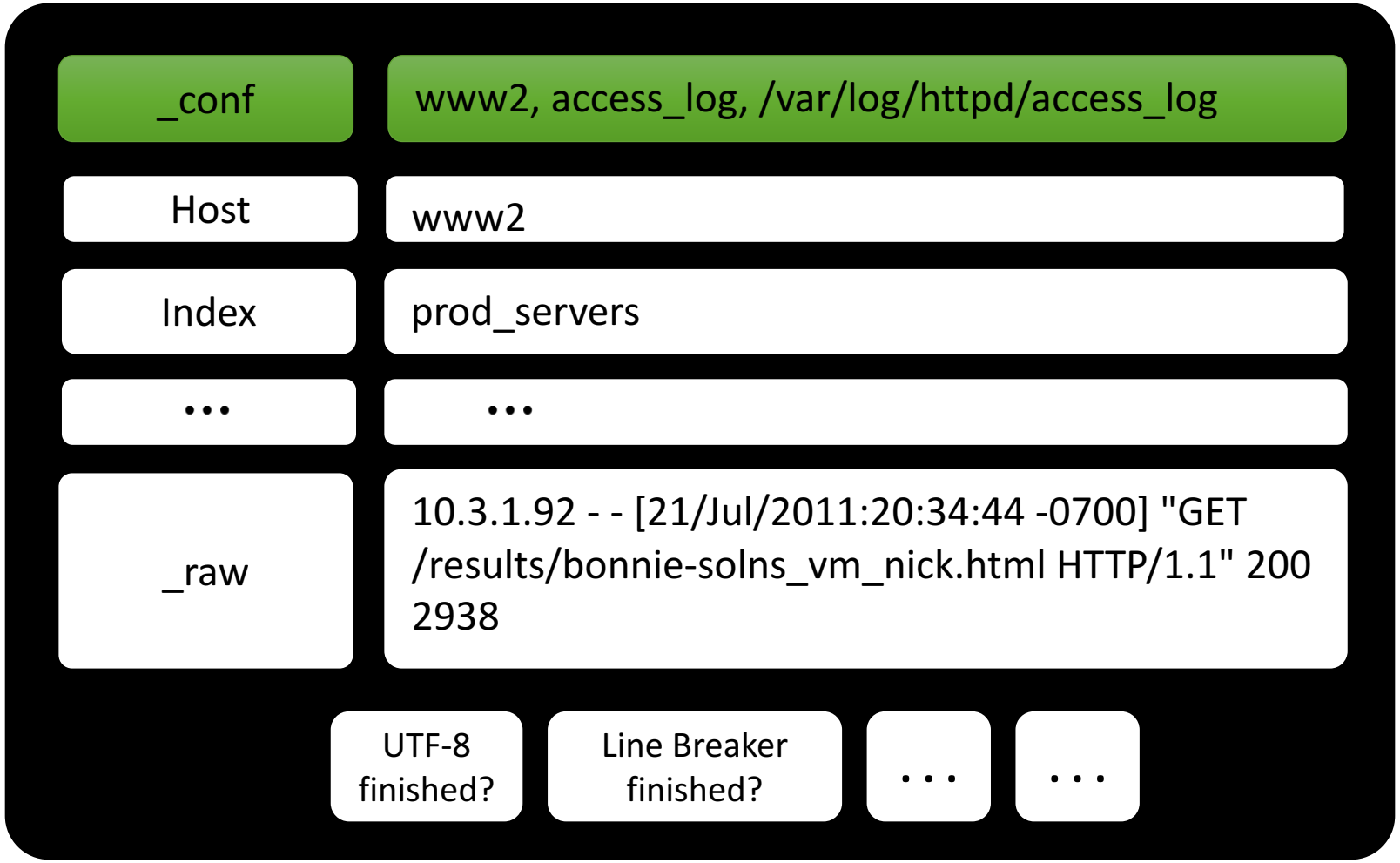
(This is the biggun')



Data structures & Routing

splunk> .conf2017

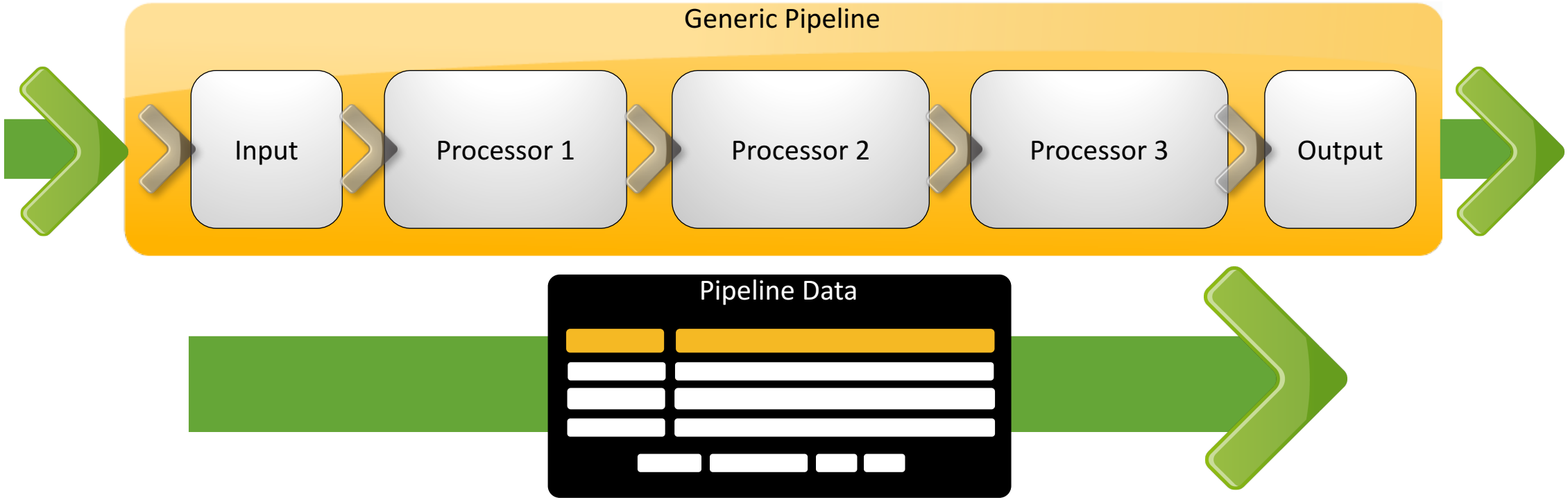
Pipeline Data



Processor

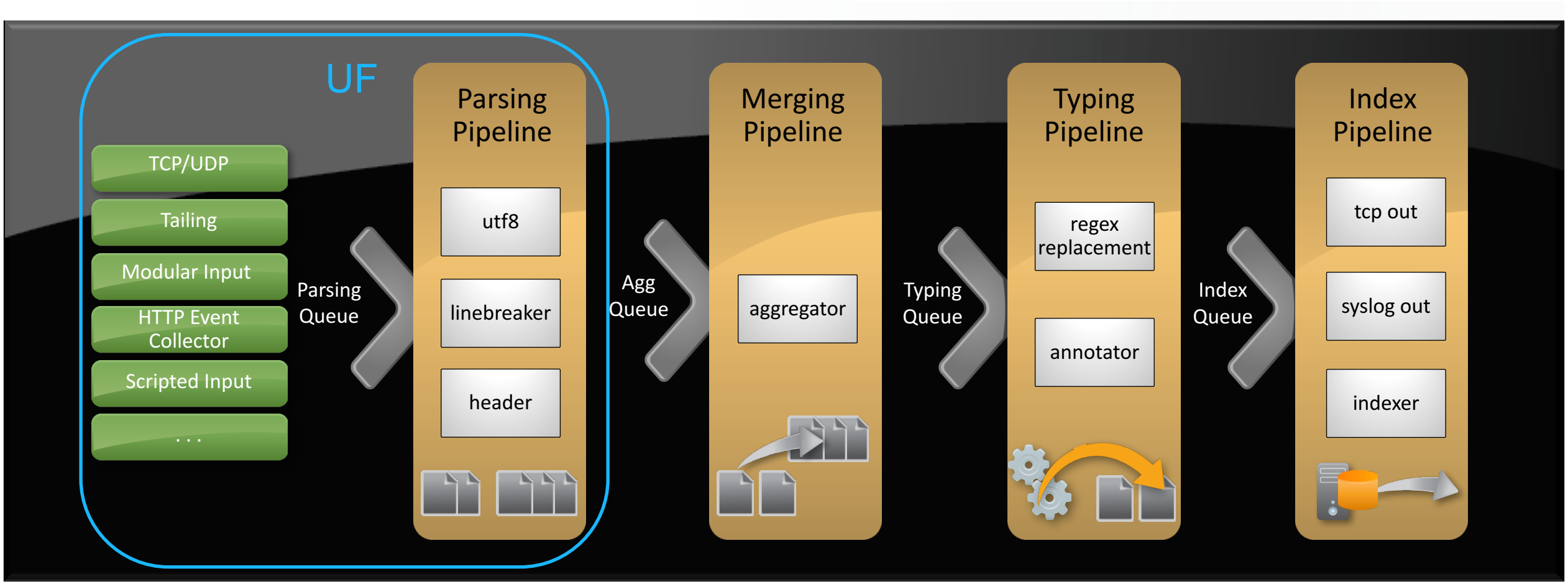
- Processor: Performs small but logical unit of work
- Contained within a Pipeline
- Examples: LineBreaker, Aggregator, TcpInput, Index

Pipelines

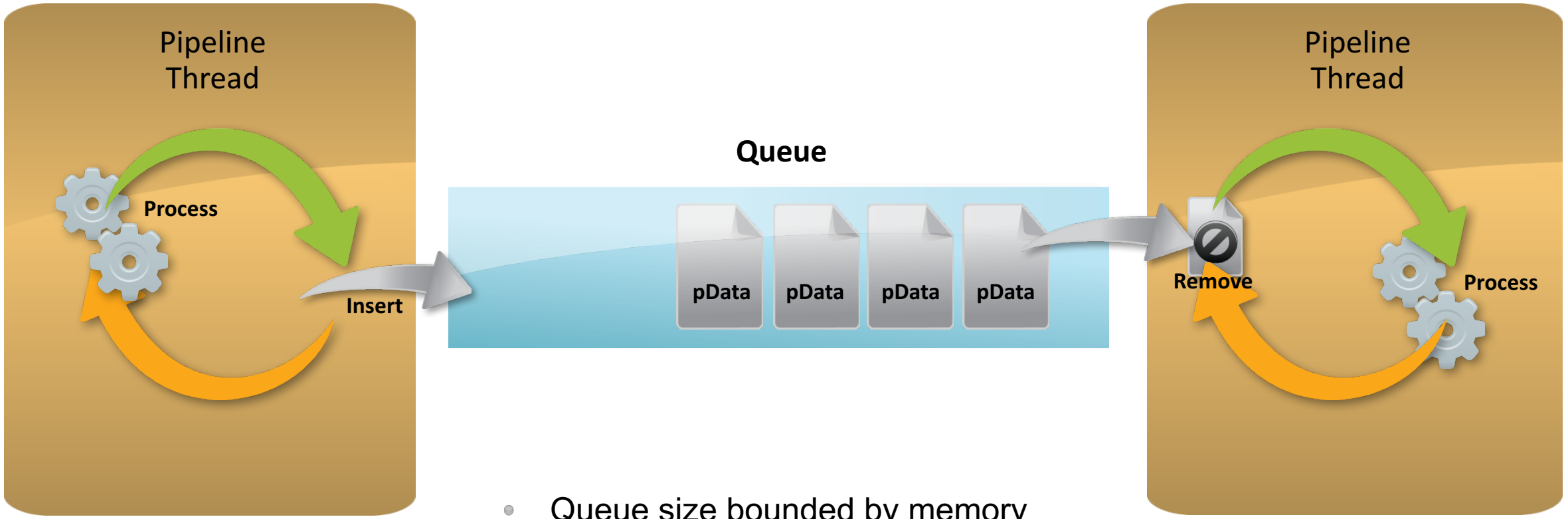


- Each pipeline runs in separate thread
- Naturally parallelized and modular
- Data flows linearly
- Configured in: $SPLUNK_HOME/etc/modules/
- Rendered to: $SPLUNK_HOME/var/run/splunk/composite.xml

Complete Ingestion Pipeline Set

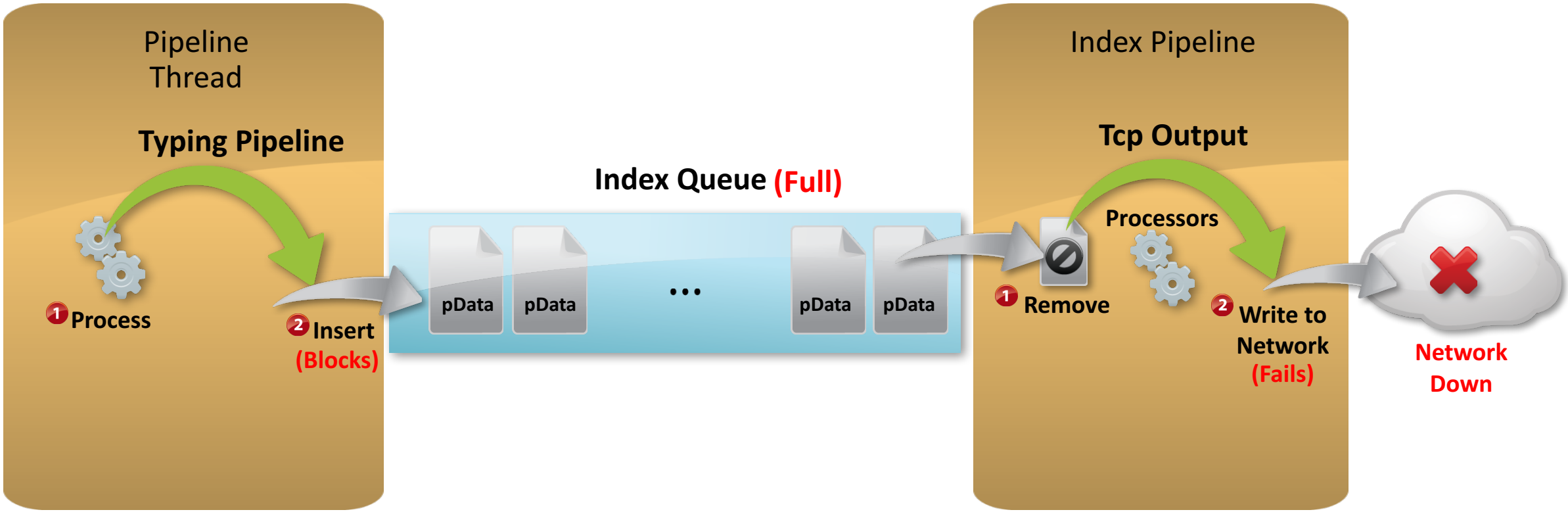


Queue

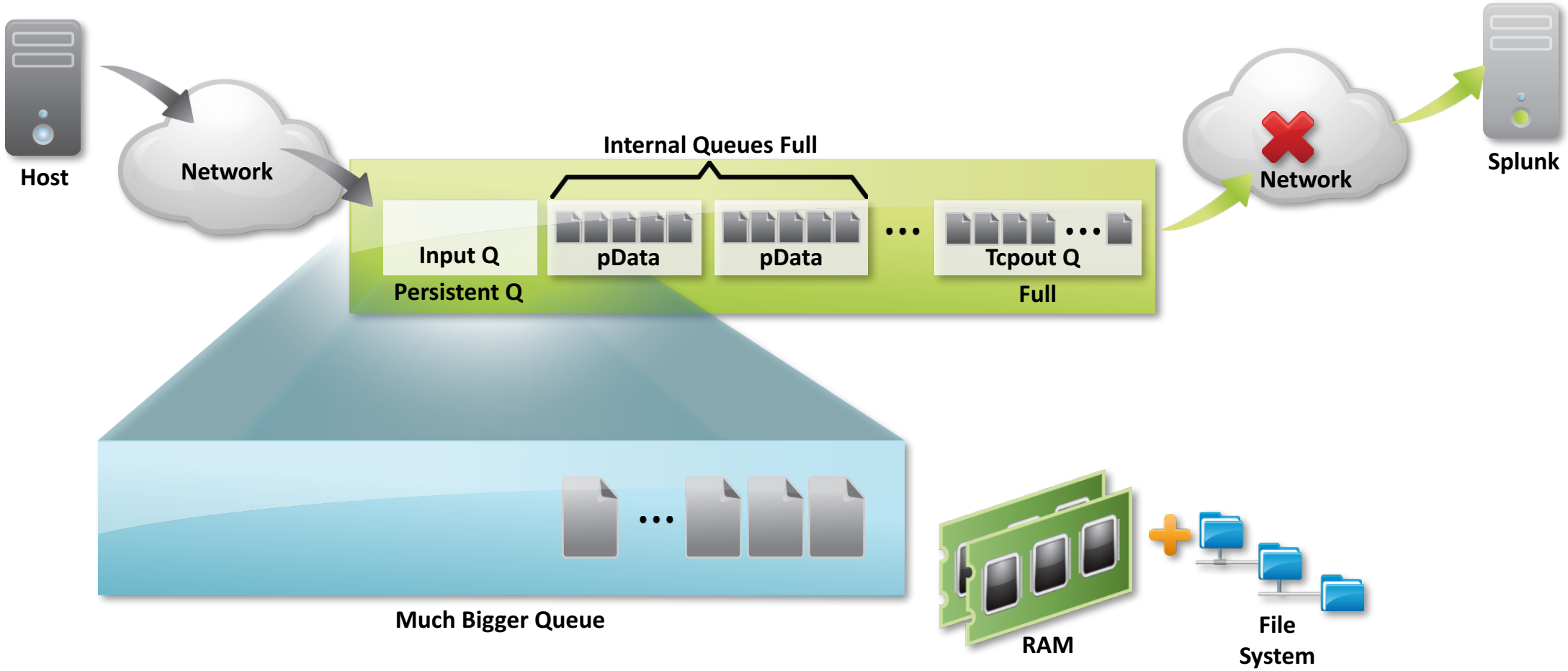


- Queue size bounded by memory
- Holds variable sized Pipeline Data

Queue



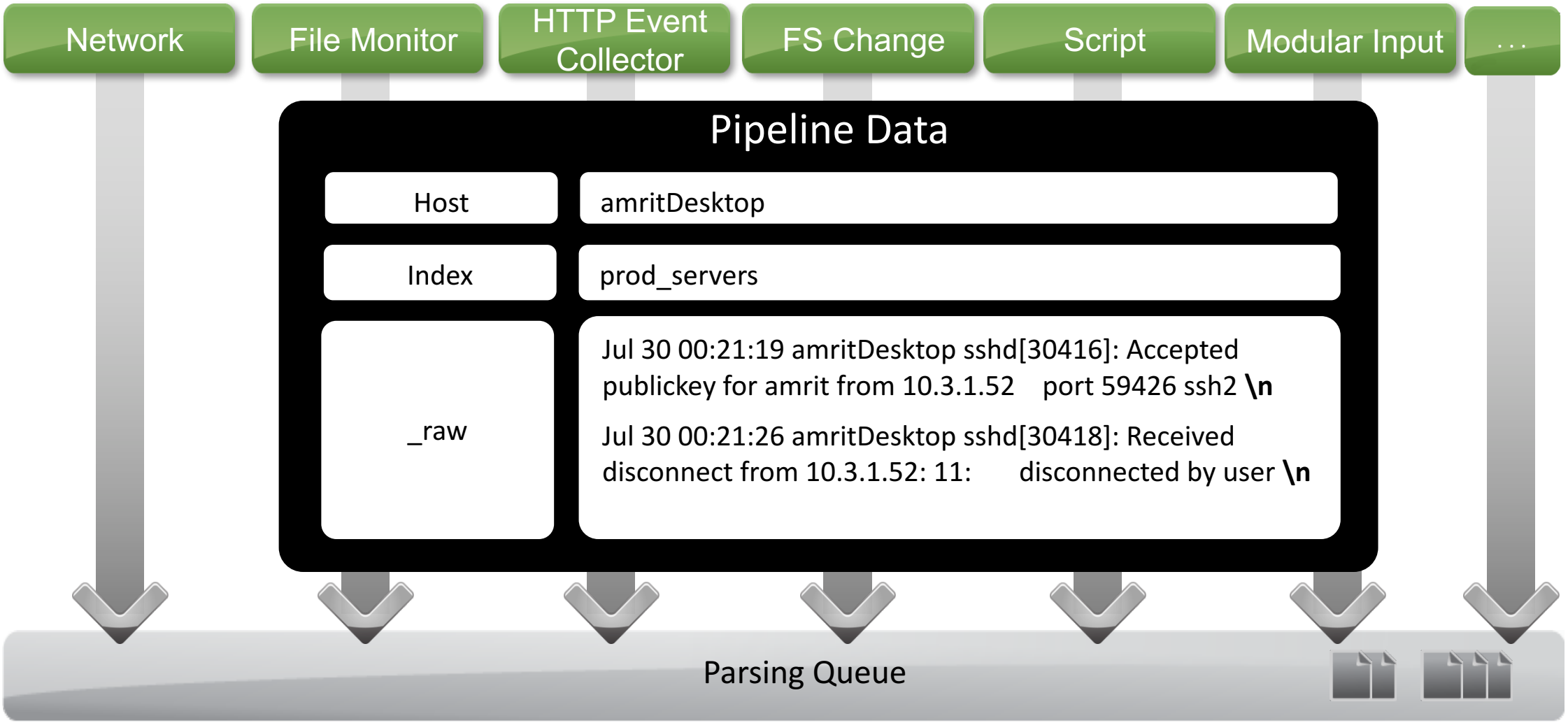
Persistent Queue



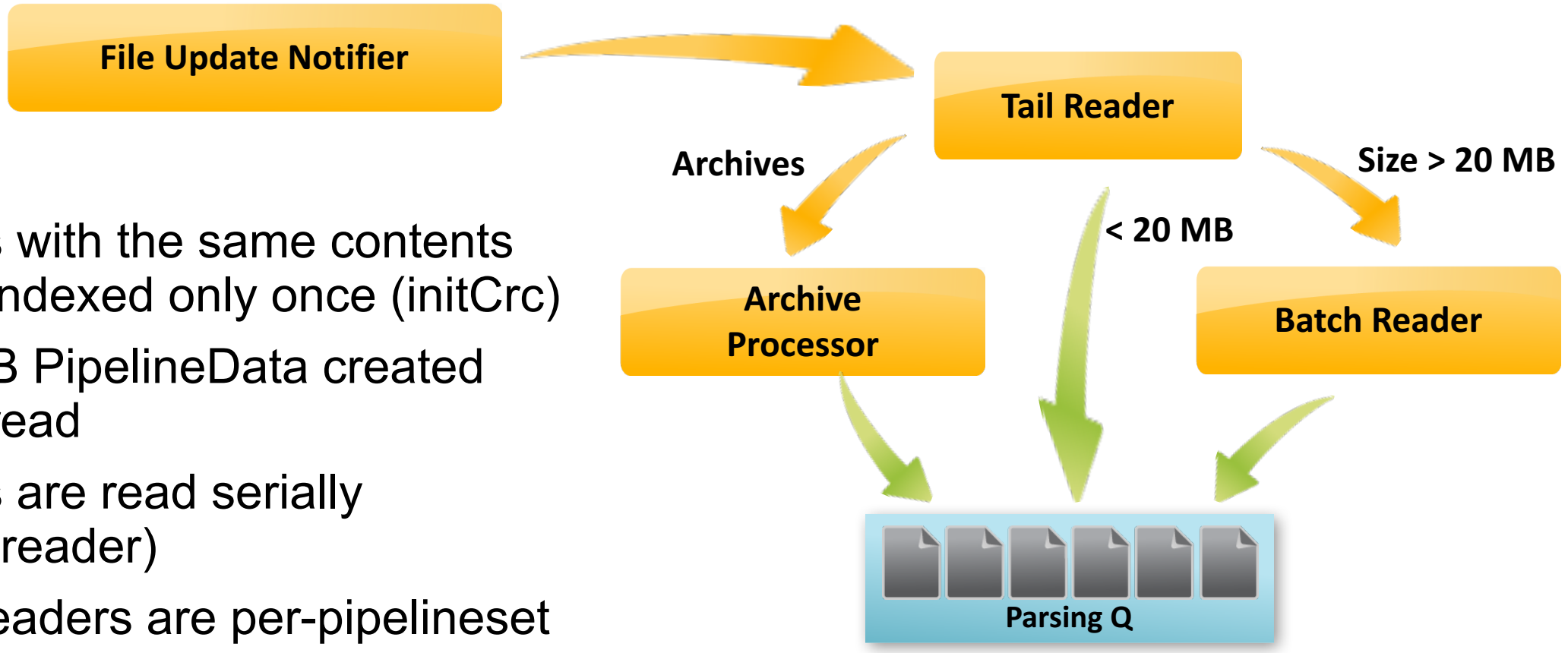
Input Processors

splunk> .conf2017

Input Pipelines

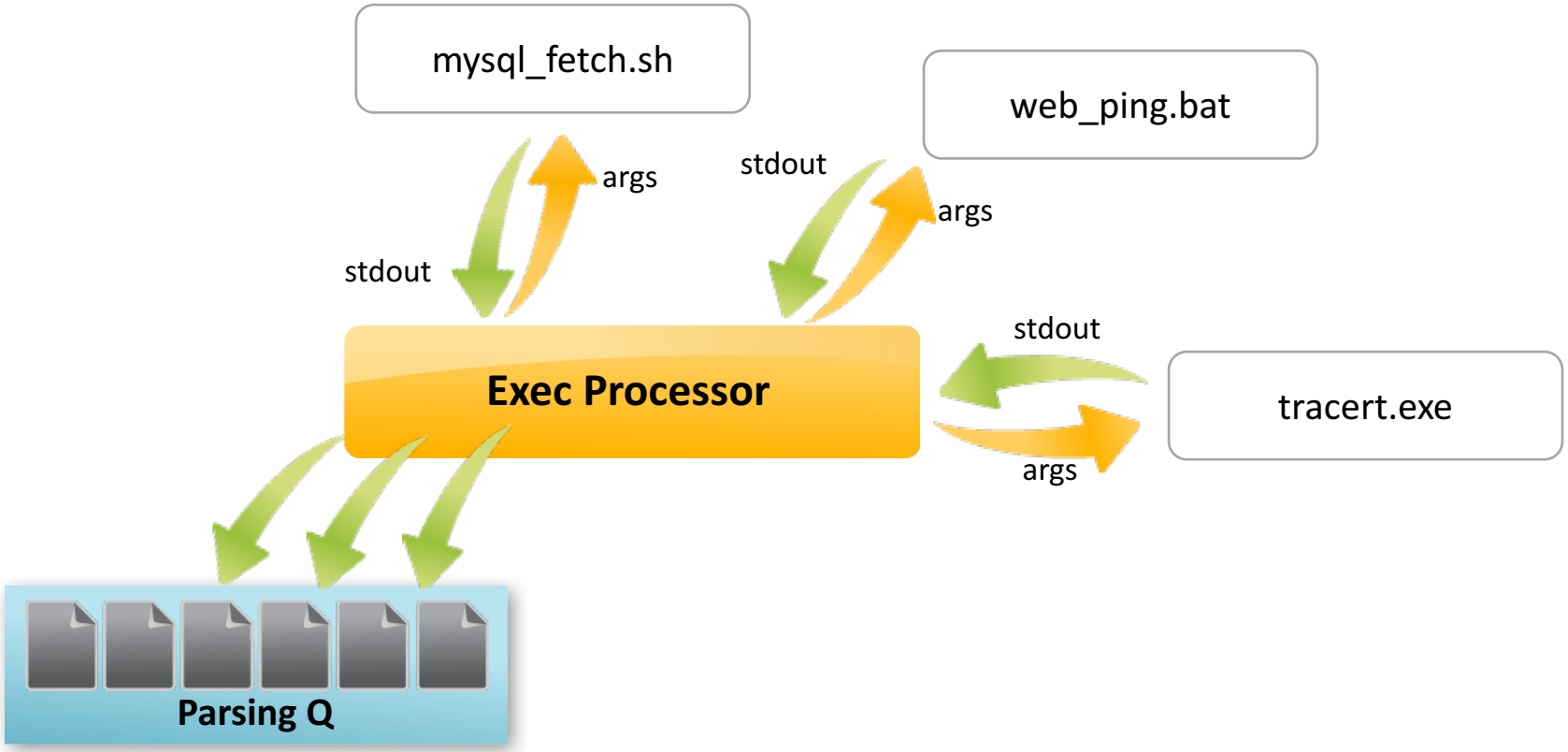


Tailing Processor (monitor input)



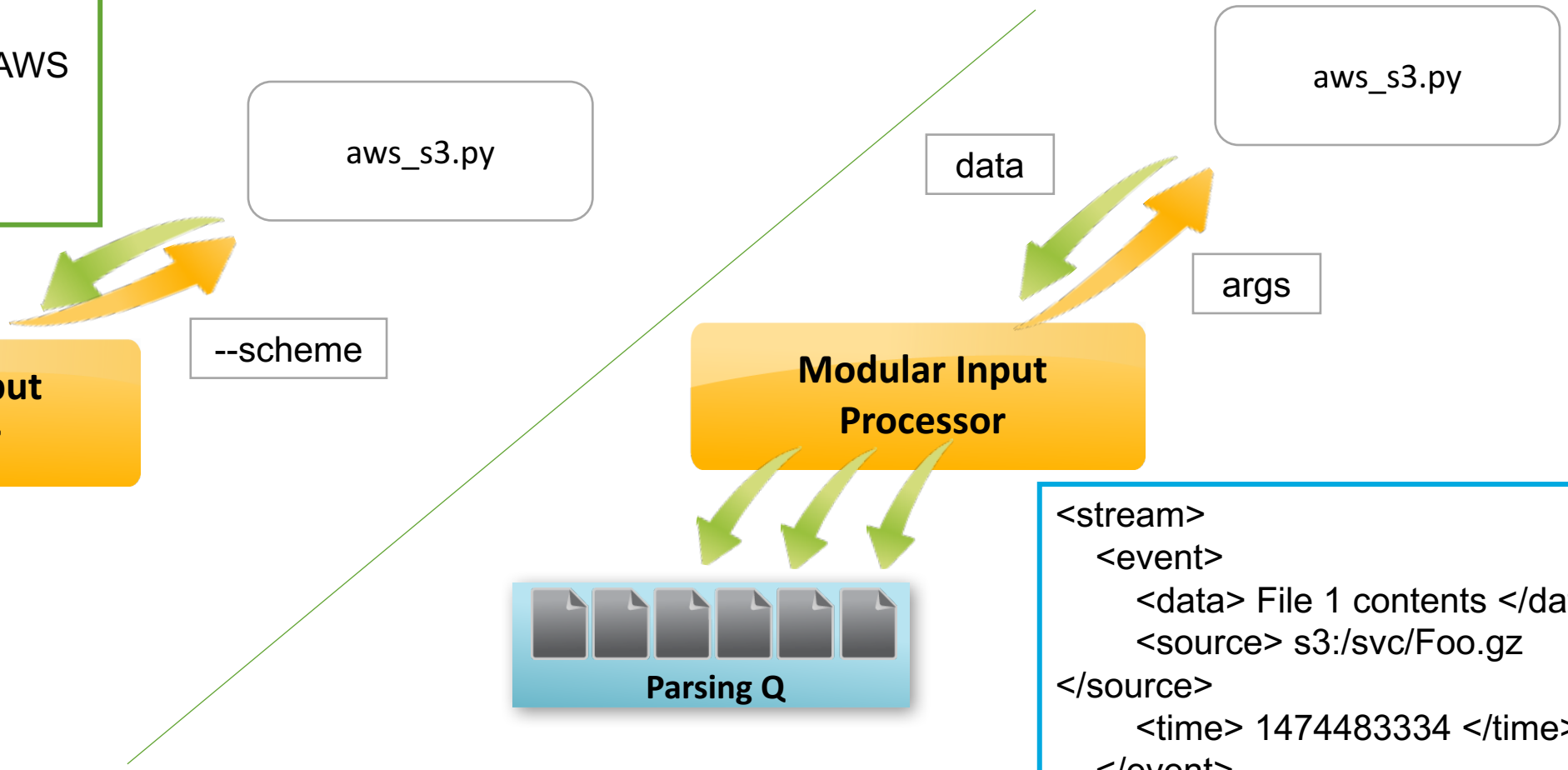
- Files with the same contents are indexed only once (initCrc)
- 64KB PipelineData created per read
- Files are read serially (per reader)
- All readers are per-pipelineset

Scripted Input



Modular Input

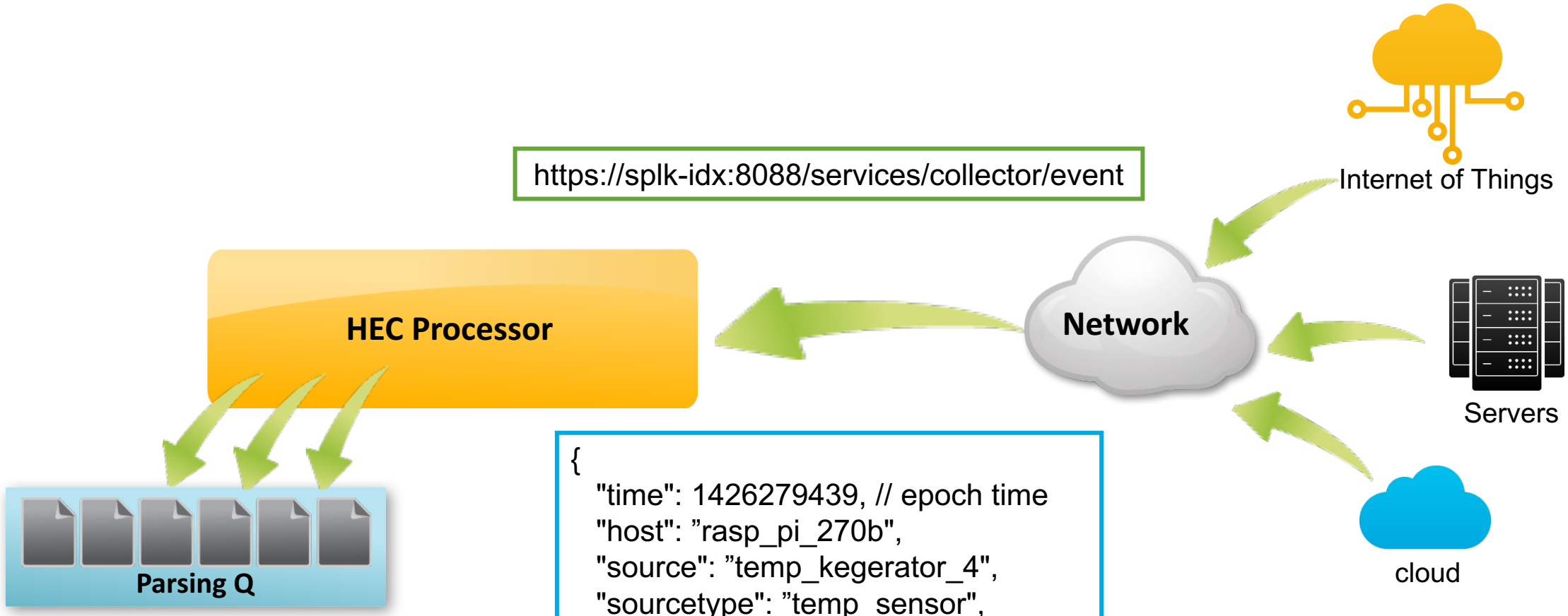
```
<scheme>
    <title>S3 Input</title>
    <description>
        Index log files from AWS
S3.
    ….
</scheme>
```



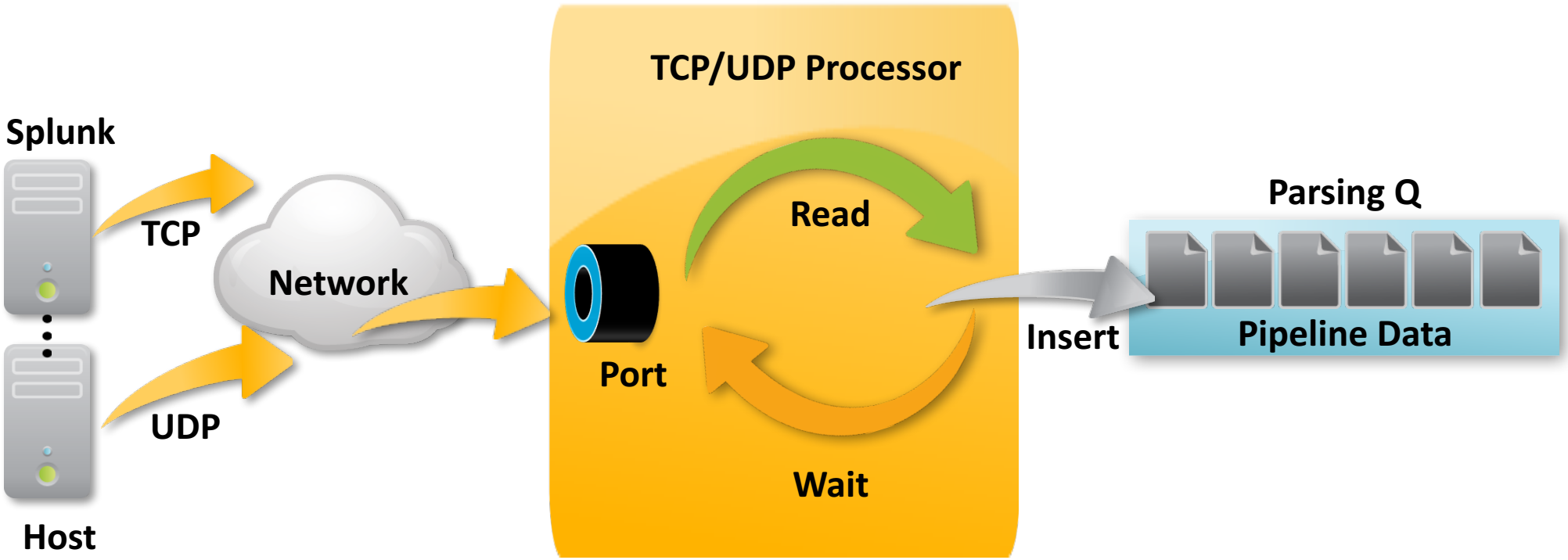

```
<stream>
    <event>
        <data> File 1 contents </data>
        <source> s3:/svc/Foo.gz
</source>
        <time> 1474483334 </time>
    </event>
    ….
</stream>
```

splunk> .conf2017

HTTP Event Collector



TCP/UDP Input

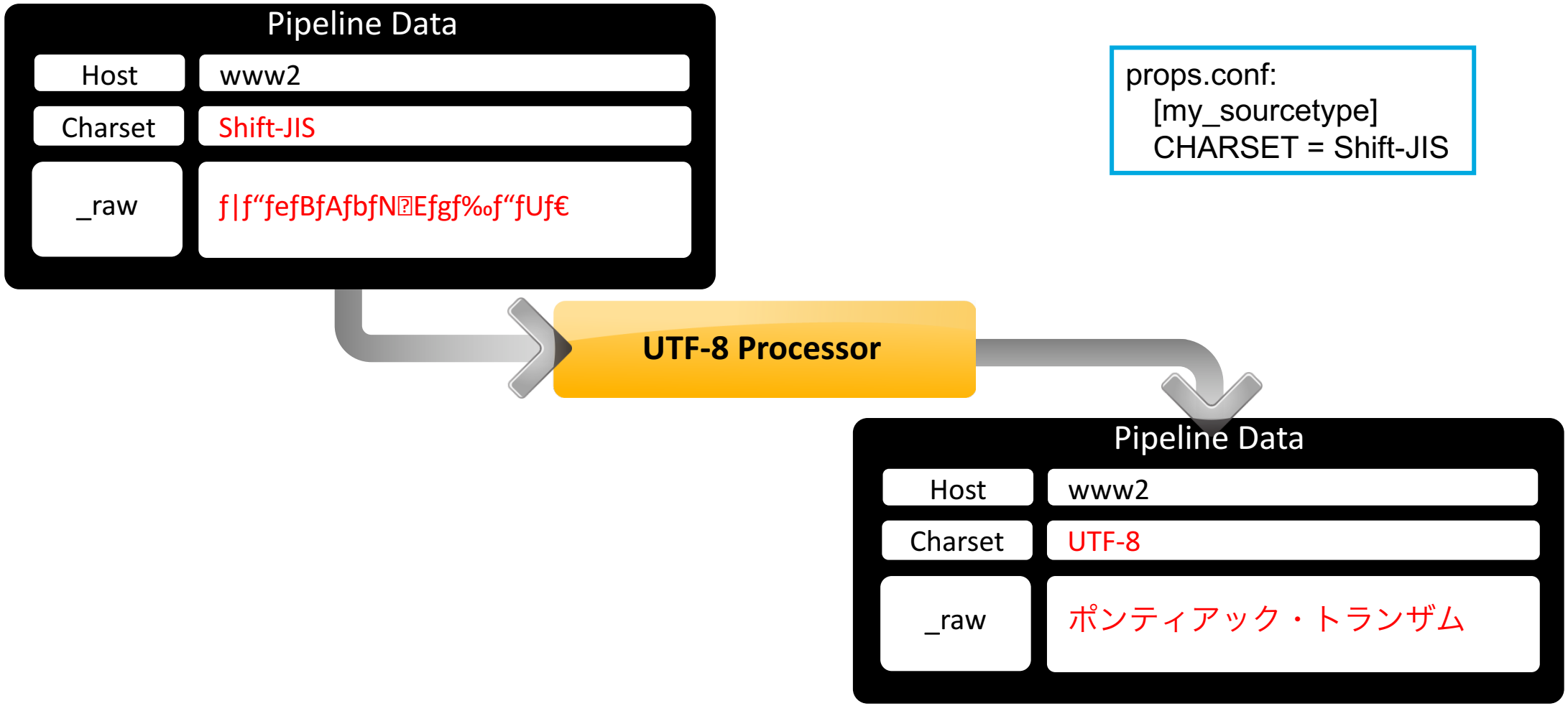


Mining Processors

We’re getting there

splunk> .conf2017

Parsing: UTF-8 Processor




```
props.conf:
   [my_sourcetype]
   CHARSET = Shift-JIS
```

Parsing: Line Breaker

Pipeline Data

_raw

Sep 12 06:11:58 abath-mba.no.cox.net storeagent[49597] <Critical>: Starting update scan ↲
Sep 12 06:11:58 abath-mba.no.cox.net storeagent[49597] <Critical>: UpdateController: Message tracing { ↲
"power_source" = ac; ↲
"start_date" = "2014-08-21 20:10:39 +0000"; ↲
} ↲
Sep 12 06:11:58 abath-mba.no.cox.net storeagent[49597] <Critical>: Asserted BackgroundTask power ↲

Line Breaker

Pipeline Data

_raw | Sep 12 06:11:58 abath-mba.no.cox.net storeagent[49597] <Critical>: Starting update scan

Pipeline Data

_raw | Sep 12 06:11:58 abath-mba.no.cox.net storeagent[49597] <Critical>: UpdateController: Message tracing {

Pipeline Data

_raw | "power_source" = ac;

Pipeline Data

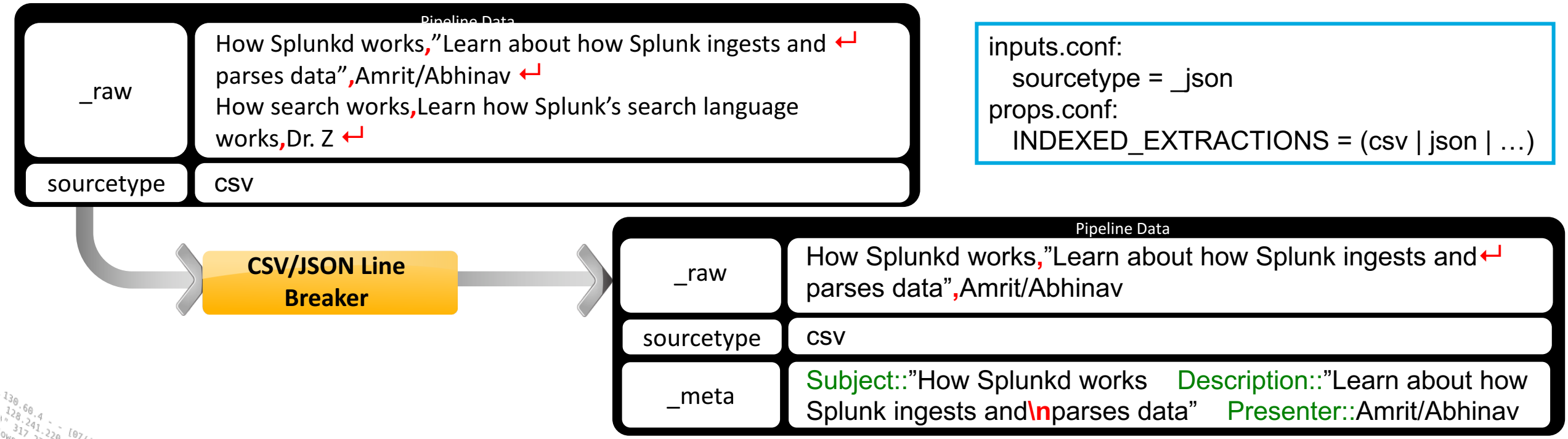
_raw | "start_date" = "2014-08-21 20:10:39 +0000";

splunk> .conf2017

OR Parsing: CSV/JSON Line Breaker (6.0+)

PRESENTATIONS.CSV

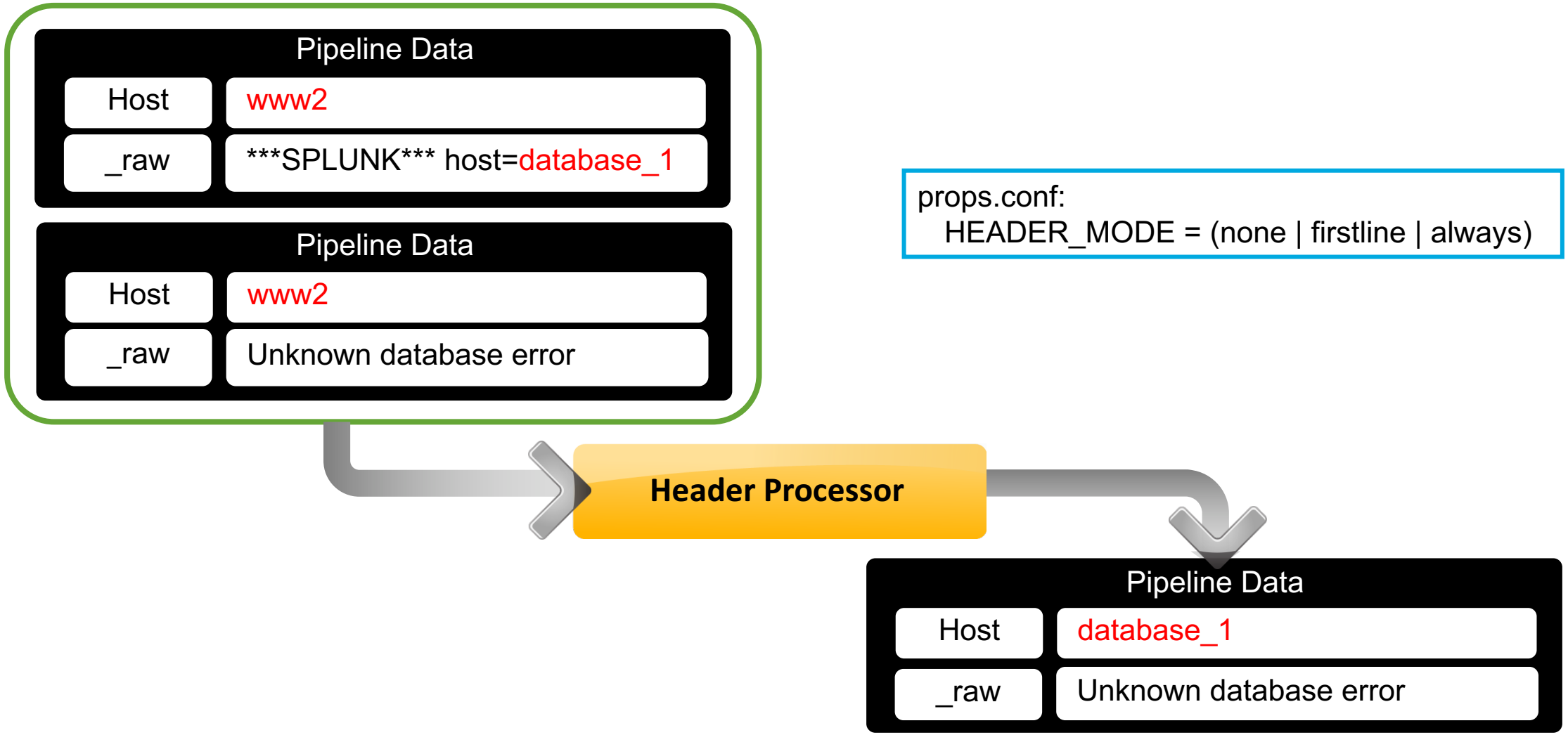
Subject ,	Description ,	Presenter ↵
How Splunkd works ,	”Learn about how Splunk ingests and ↵ parses data” ,	Amrit/Abhinav ↵
How search works ,	Learn how Splunk’s search language works ,	Dr. Z ↵




```
inputs.conf:
   sourcetype = _json
props.conf:
   INDEXED_EXTRACTIONS = (csv | json | …)
```

splunk> .conf2017

Parsing: Header Processor




```
props.conf:
   HEADER_MODE = (none | firstline | always)
```

Merging: Line Merge (aka Aggregator)

Pipeline Data

_raw	Sep 12 06:11:58 abath-mba13.no.cox.net storeagent[49597] <Critical>: Starting update scan

Pipeline Data

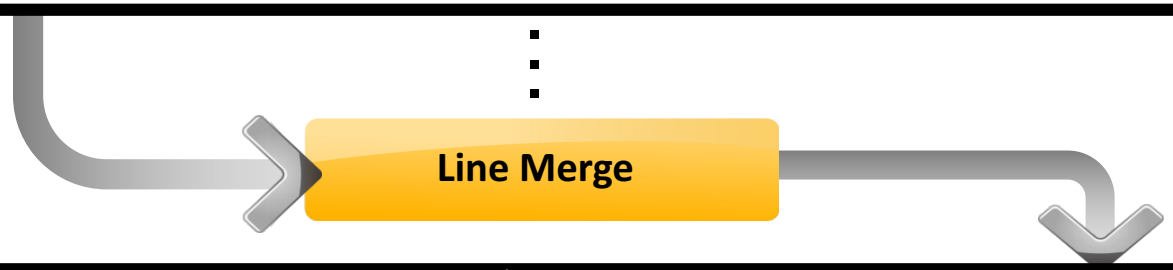
_raw	Sep 12 06:11:58 abath-mba13.no.cox.net storeagent[49597] <Critical>: UpdateController: Message tracing {

Pipeline Data

_raw	"power_source" = ac;

Pipeline Data

_raw	"start_date" = "2014-08-21 20:10:39 +0000";



Pipeline Data

_raw	Sep 12 06:11:58 abath-mba13.no.cox.net storeagent[49597] <Critical>: Starting update scan

Pipeline Data

_raw	Sep 12 06:11:58 abath-mba13.no.cox.net storeagent[49597] <Critical>: UpdateController: Message tracing {
	"power_source" = ac;
	"start_date" = "2014-08-21 20:10:39 +0000";
	}

Pipeline Data

_raw	Sep 12 06:11:58 abath-mba13.no.cox.net storeagent[49597] <Critical>: Asserted BackgroundTask power

Typing: Regex Replacement

Pipeline Data	
Host	logbox
Sourcetype	syslog
_raw	Sep 12 06:11:58 abath-mba13.no.cox.net storeagent[49597] <Critical>: Starting update scan

Regex Replacement

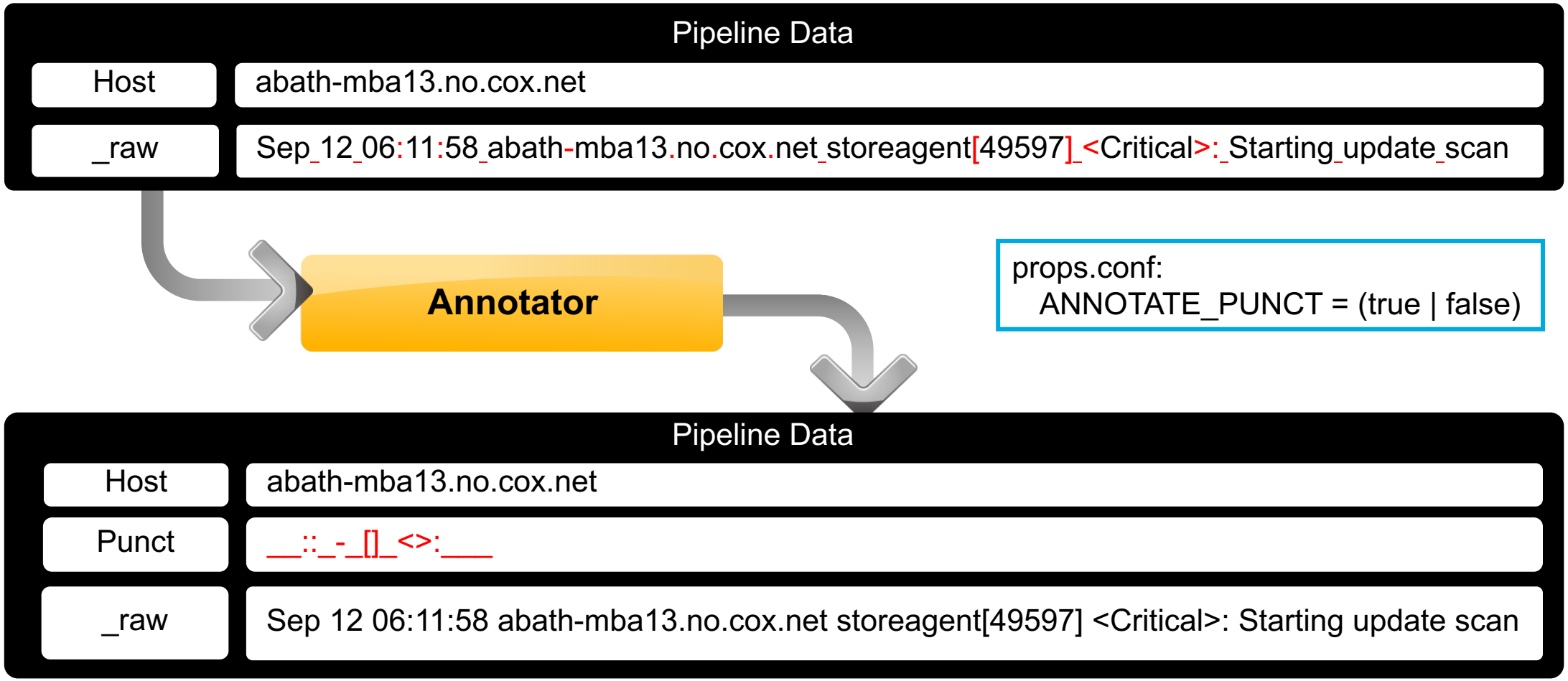
```
props.conf:
   [syslog]
   TRANSFORMS = syslog-host

transforms.conf:
   [syslog-host]
   DEST_KEY = MetaData:Host
   FORMAT = host::$1
   REGEX = . . . ( . . . ) . . .
```

Pipeline Data	
Host	abath-mba13.no.cox.net
Sourcetype	syslog
_raw	Sep 12 06:11:58 abath-mba13.no.cox.net storeagent[49597] <Critical>: Starting update scan

splunk> .conf2017

Typing: Annotator



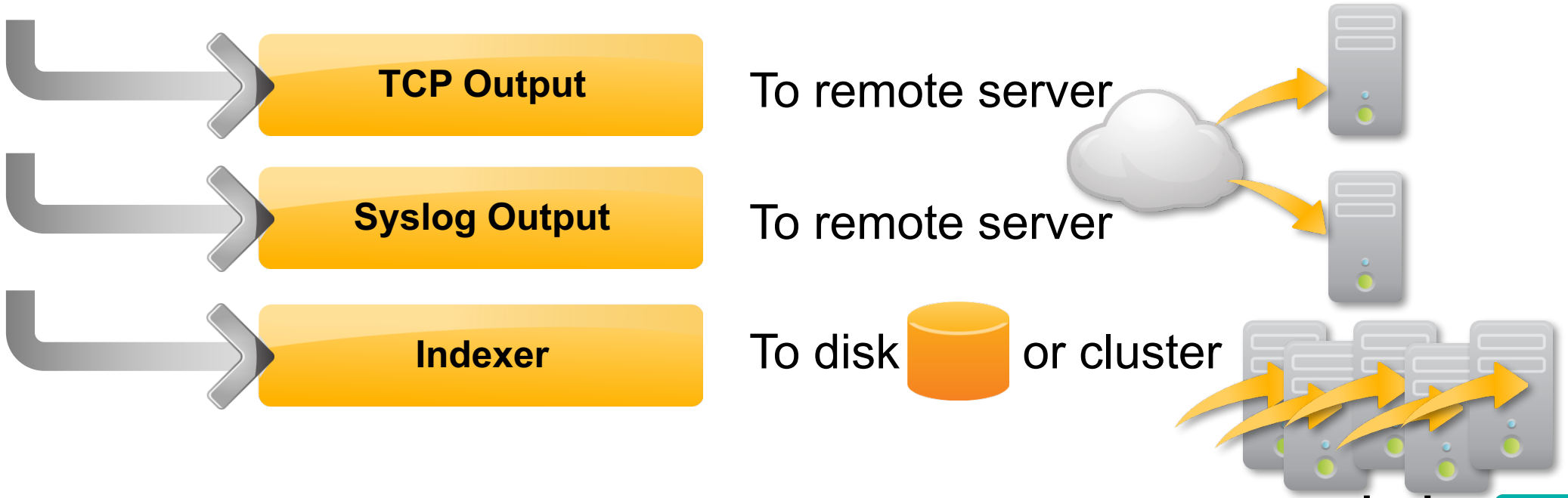
Index & Output Processors

We're still here…

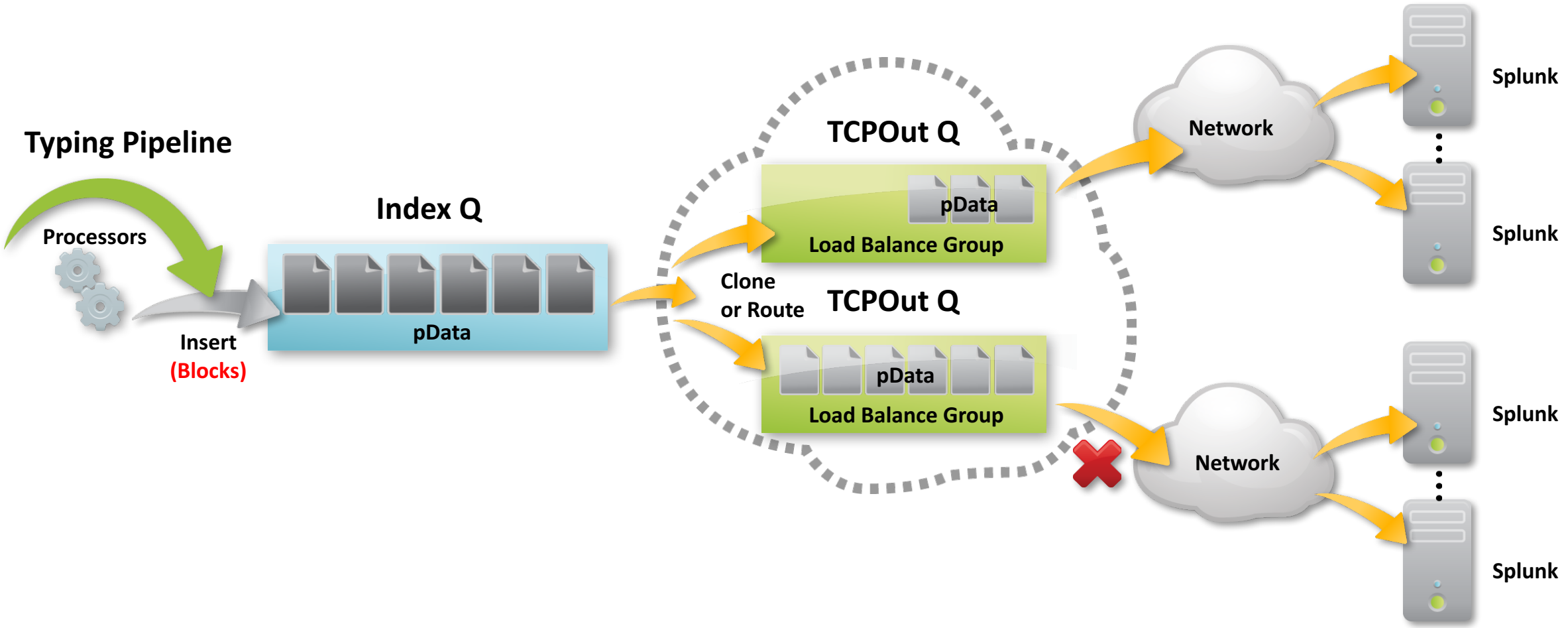
splunk> .conf2017

Indexer Pipeline: Fwding, Indexing

Pipeline Data	
Host	abath-mba13.no.cox.net
Index	main
_raw	Sep 12 06:11:58 abath-mba13.no.cox.net storeagent[49597] <Critical>: Starting update scan



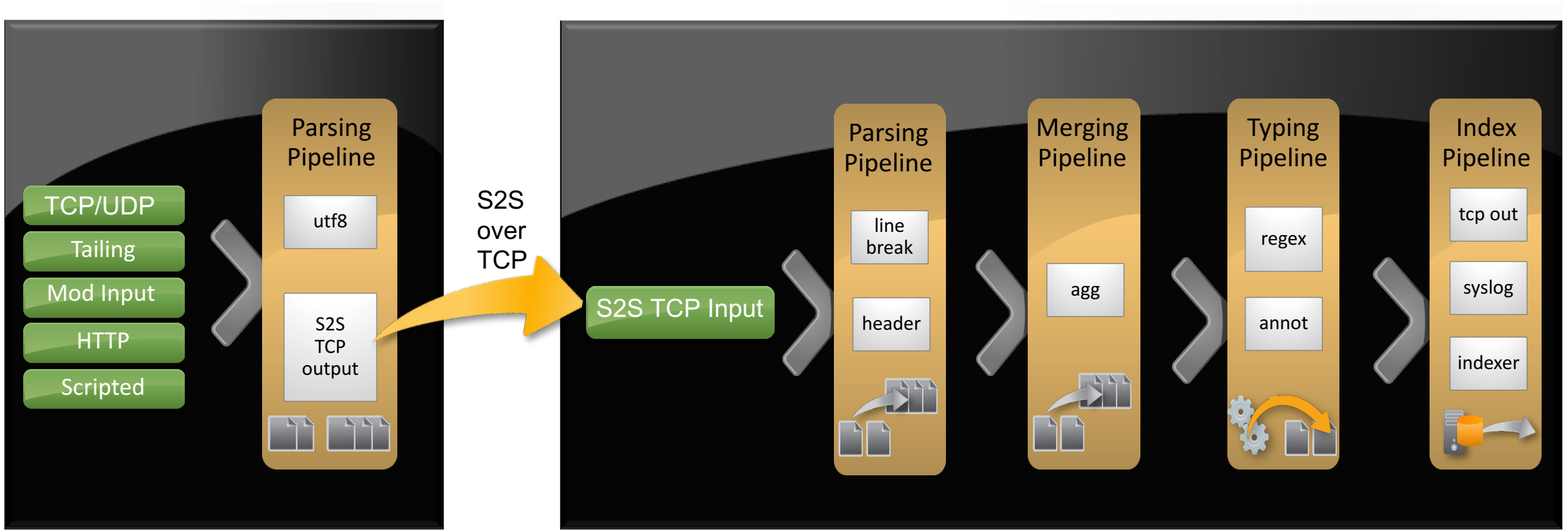
TCP Output Qs



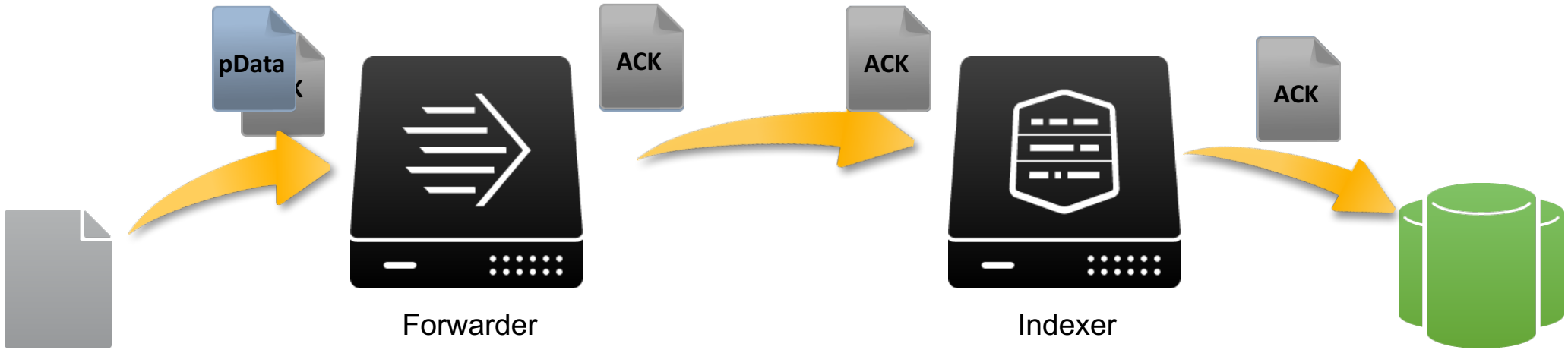
Forwarding and Receiving

Hi

Forwarding



Forwarding with ACK



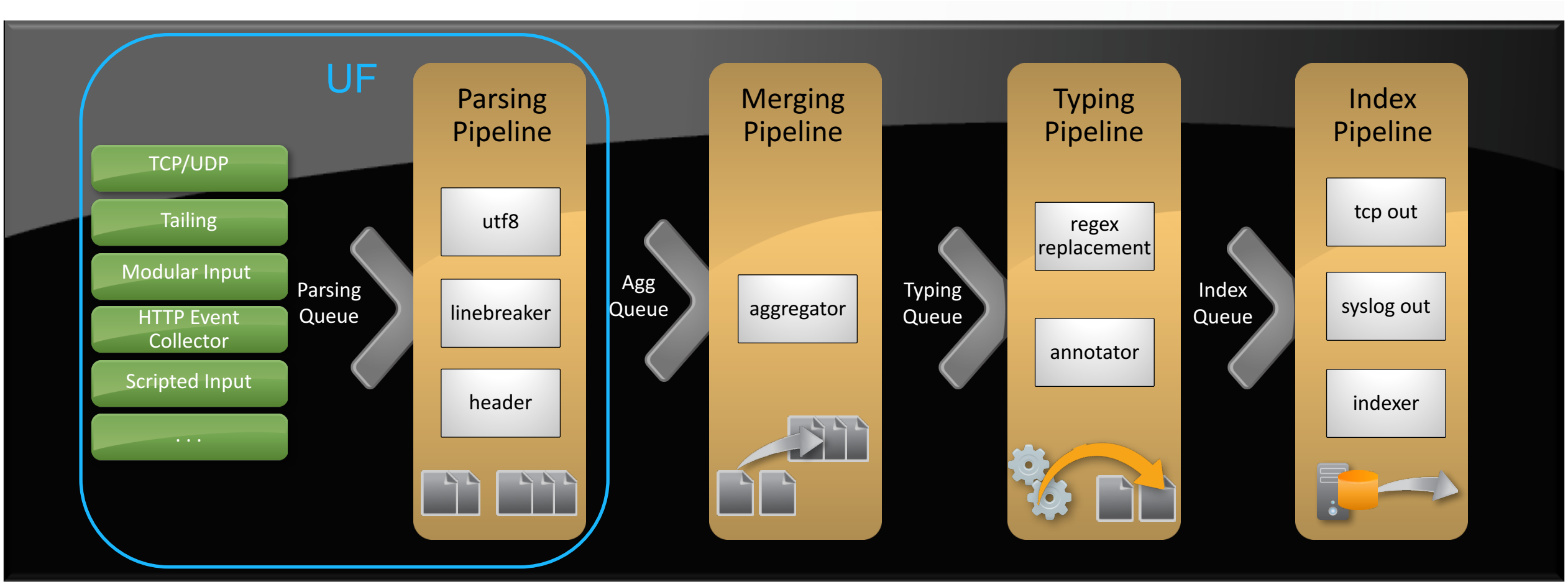

```
outputs.conf:
    [tcpout:foobar]
    useACK = true
```

Multiple Pipeline Sets

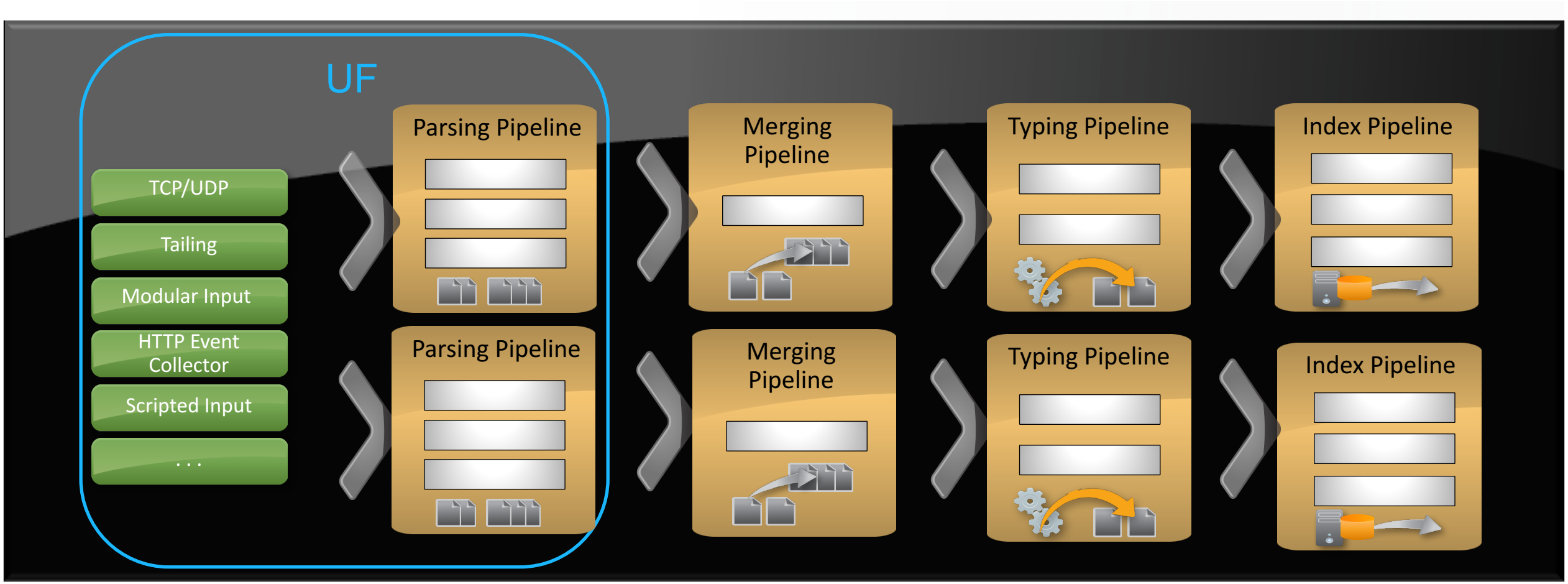
More faster, sometimes

splunk> .conf2017

Complete Ingestion Pipeline Set (Recap)



With Multiple Pipeline Sets (6.3+)



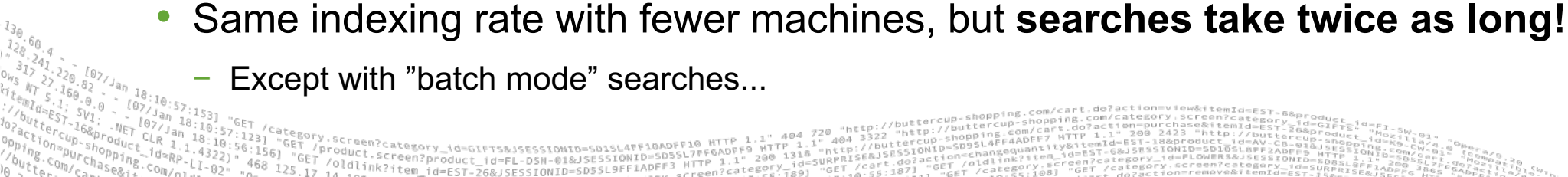
Pipeline Sets: CPU Utilization

- Rule of Thumb:
  - 4-6 cores per set
  - 1 core per search (unless batch_search_max_pipeline > 1…)
- Example: Indexer host with 2 Pipeline Sets:
  - 2 x 6 = 12 cores for splunkd daemon
  - 10 searches running
  - Other cores: “splunk-optimize”, etc, periodically.
  - Total: Roughly 10 + 12 = 22 cores in use.
- Parallel pipeline sets for underutilized hosts:

```
server.conf:
  [general]
  parallelIngestionPipelines = 2
```

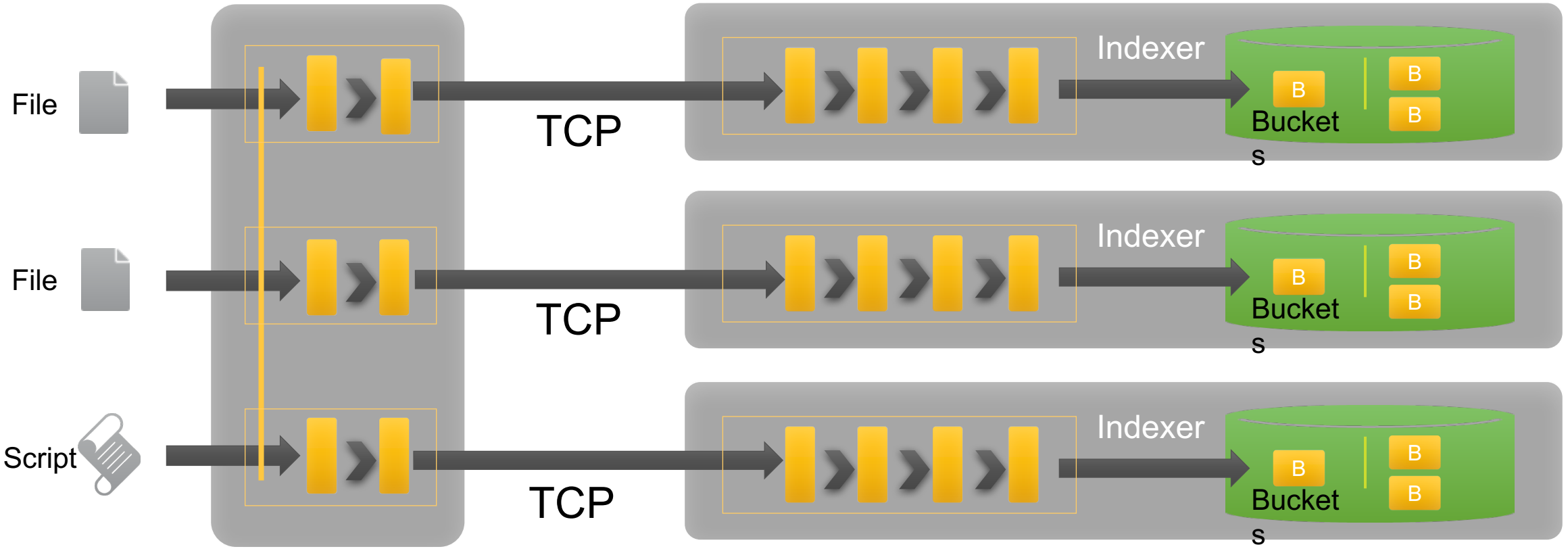
Pipeline Sets: Not an easy decision!

- Can increase forwarding or indexing rate:
  - Total rate = *base_rate* x *num_pipeline_sets*
  - **Good idea** on edge Forwarders *(see next slide…)*
  - **Bad idea** on already-overwhelmed indexers!
- **Does not reduce search time!**
  - Search time = *base_search_time* / *number_machines*
- Assume: indexing *N* GB/day
  - On 4 indexers, pipeline sets = 1.
    - Dense search, 24 hours: Each indexer searches (*N* / 4) GB.
  - Increase to pipeline sets = 2, downsize to 2 indexers.
    - Dense search, 24 hours: Each indexer searches (*N* / 2) GB.
    - Same indexing rate with fewer machines, but **searches take twice as long!**
      - Except with ”batch mode” searches...



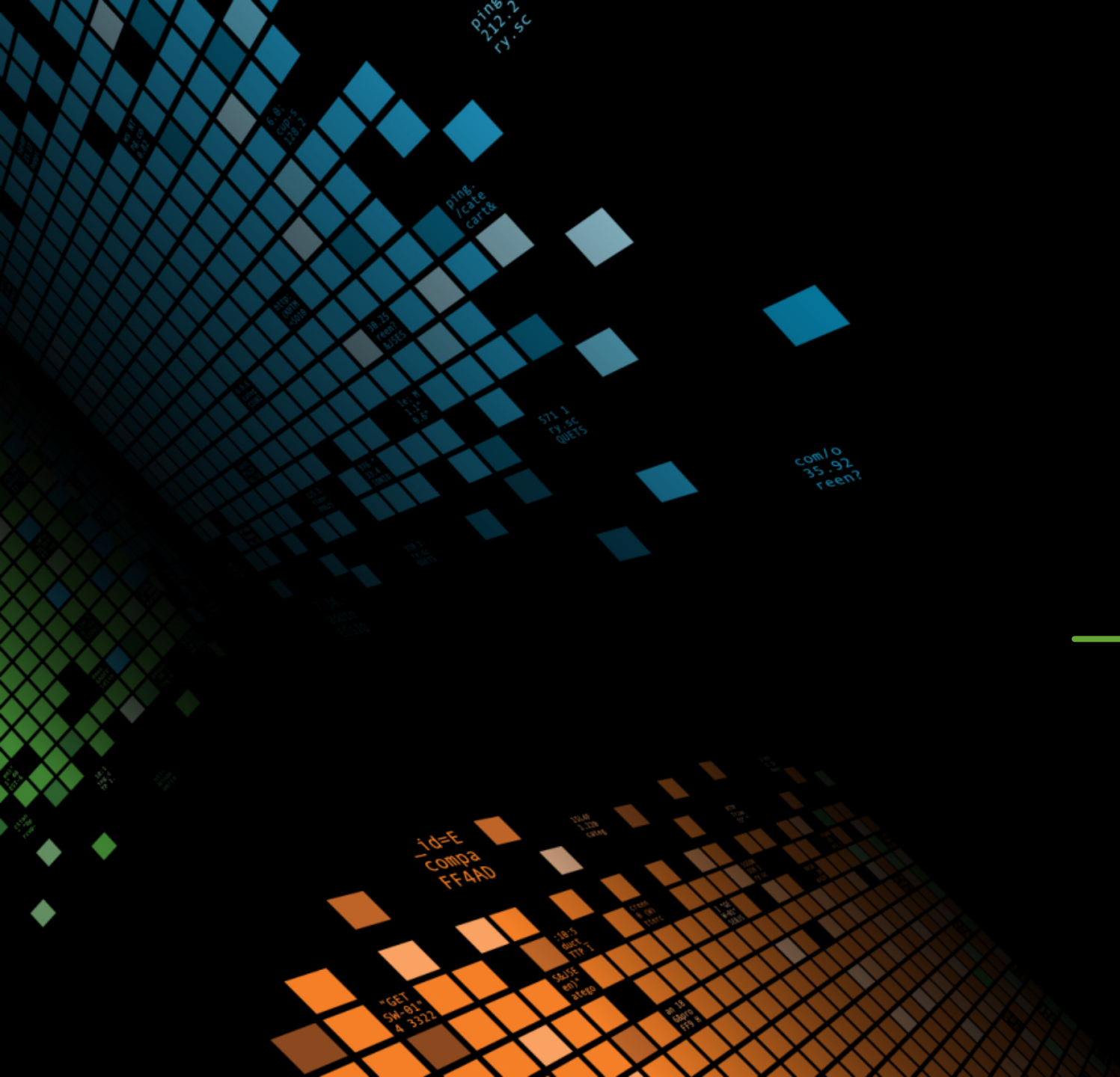
splunk> .conf2017

Multiple Ingestion Pipeline Sets over Network



Forwarder: 3 Pipeline Sets

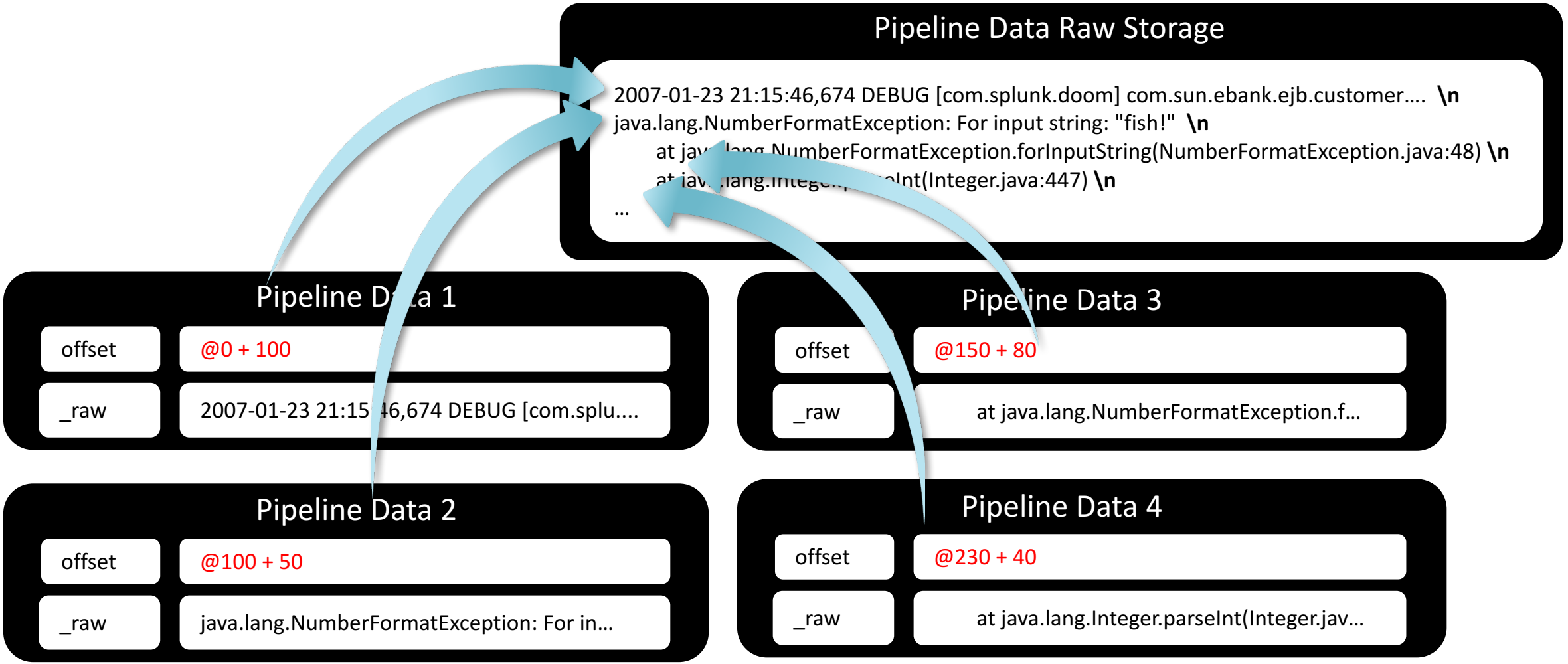
3 Indexers: 1 Pipeline Set each



Nerdier Stuff

8-^)

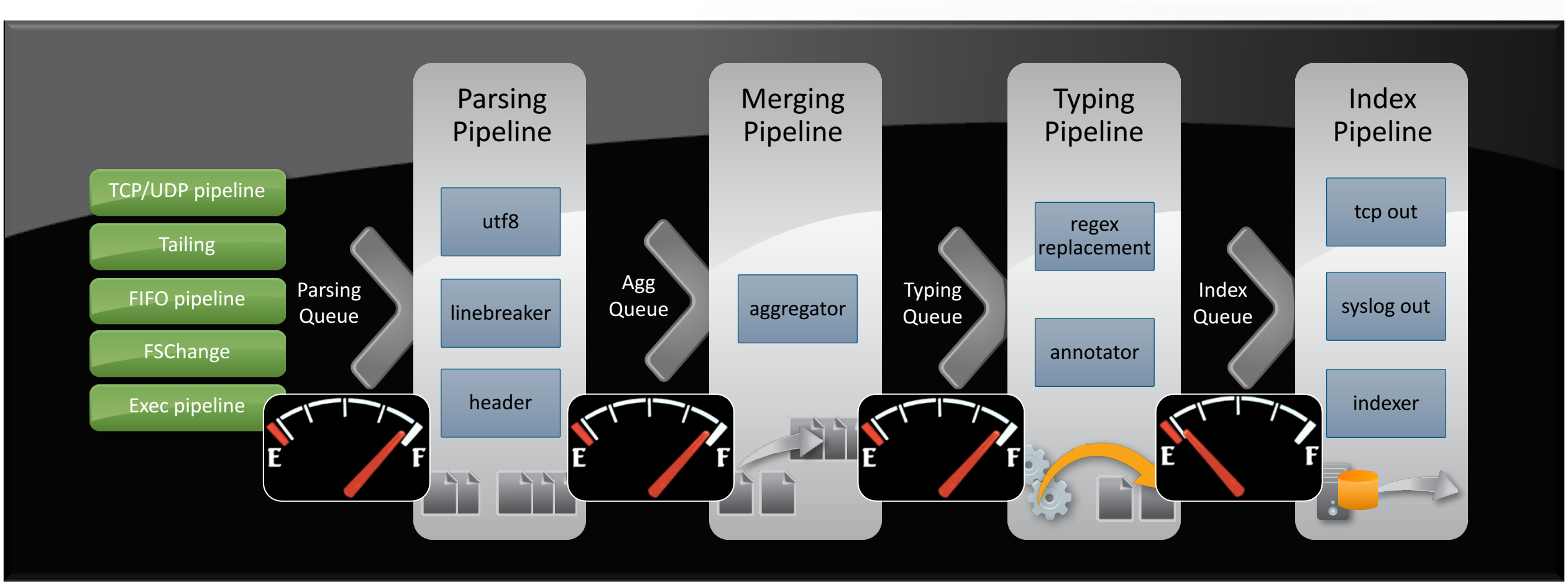
Resource Management



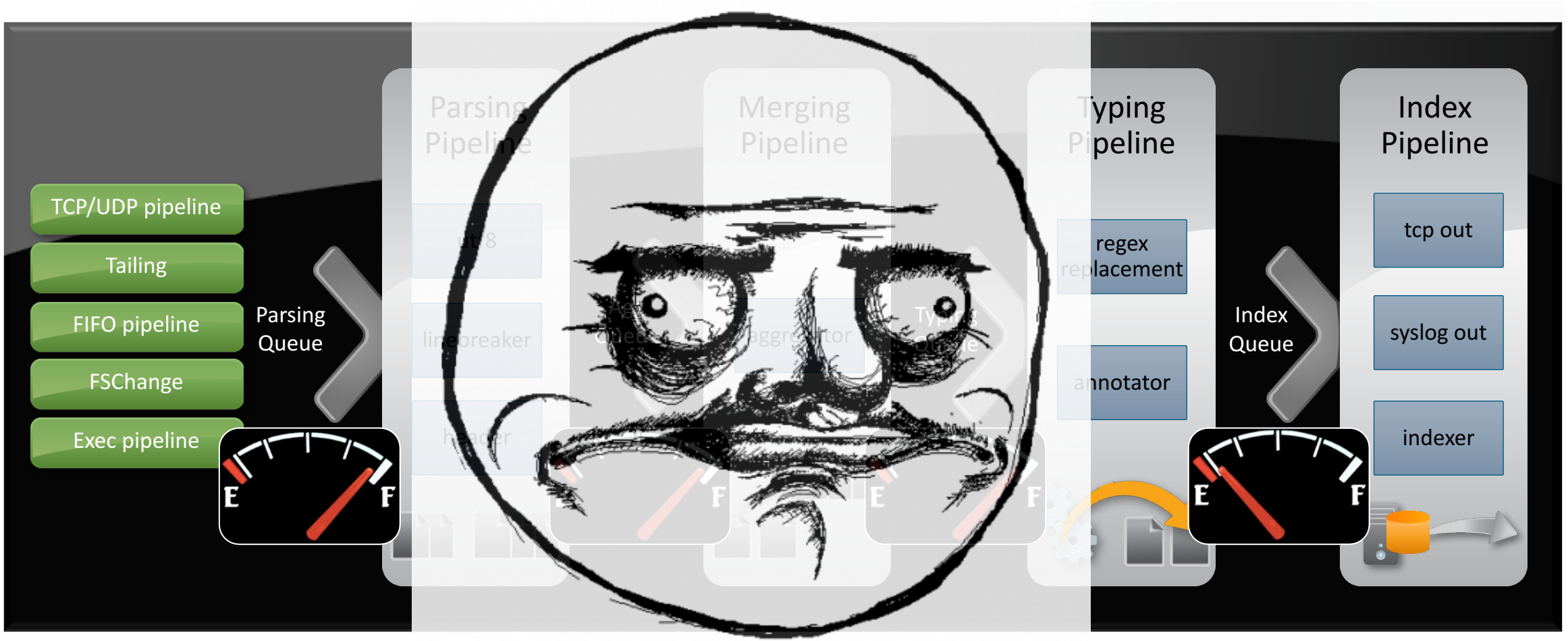
splunk> .conf2017

Debugging! Monitoring Console! (aka DMC!)

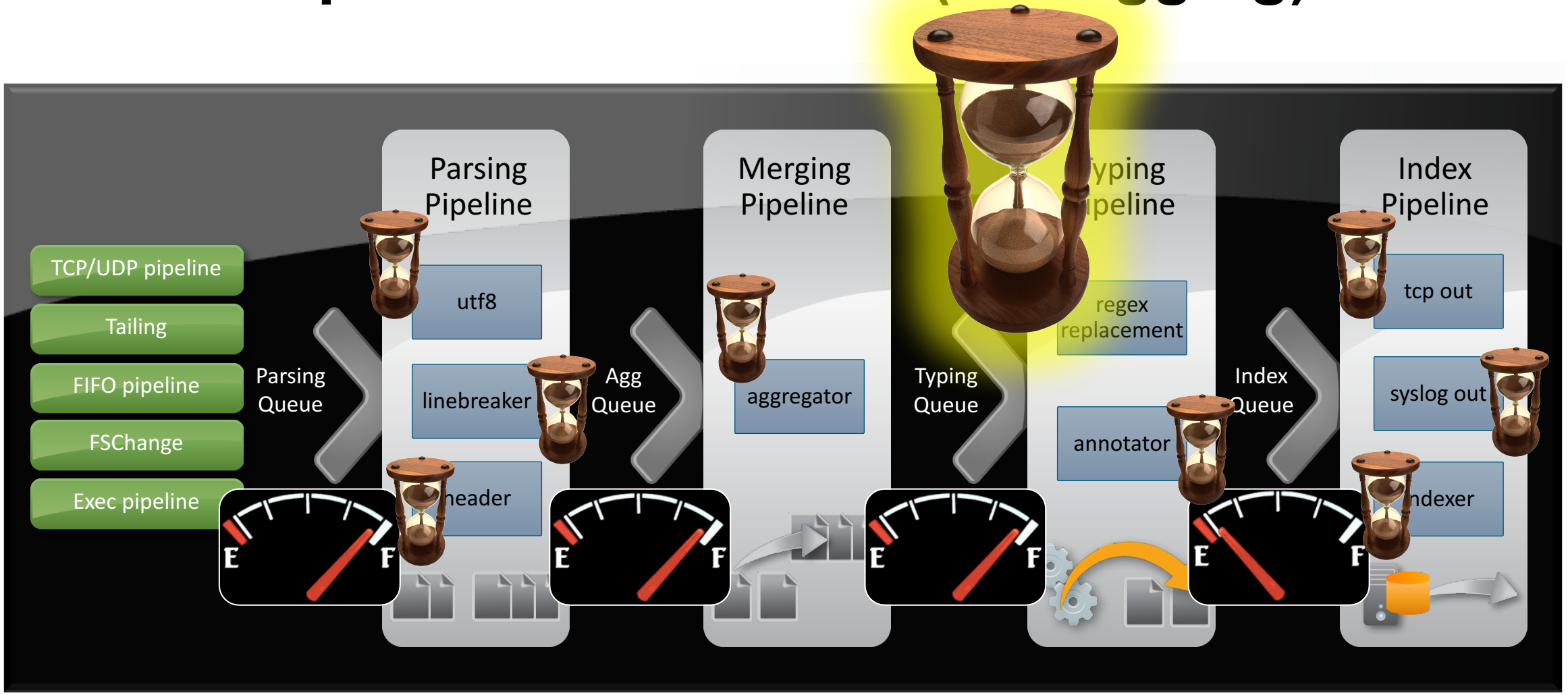
Pipelines/Processors (Debugging)



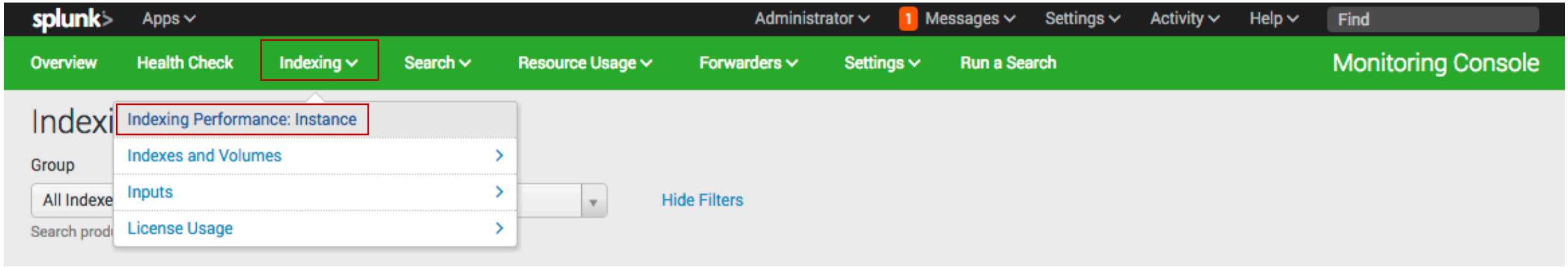
Pipelines/Processors (Debugging)



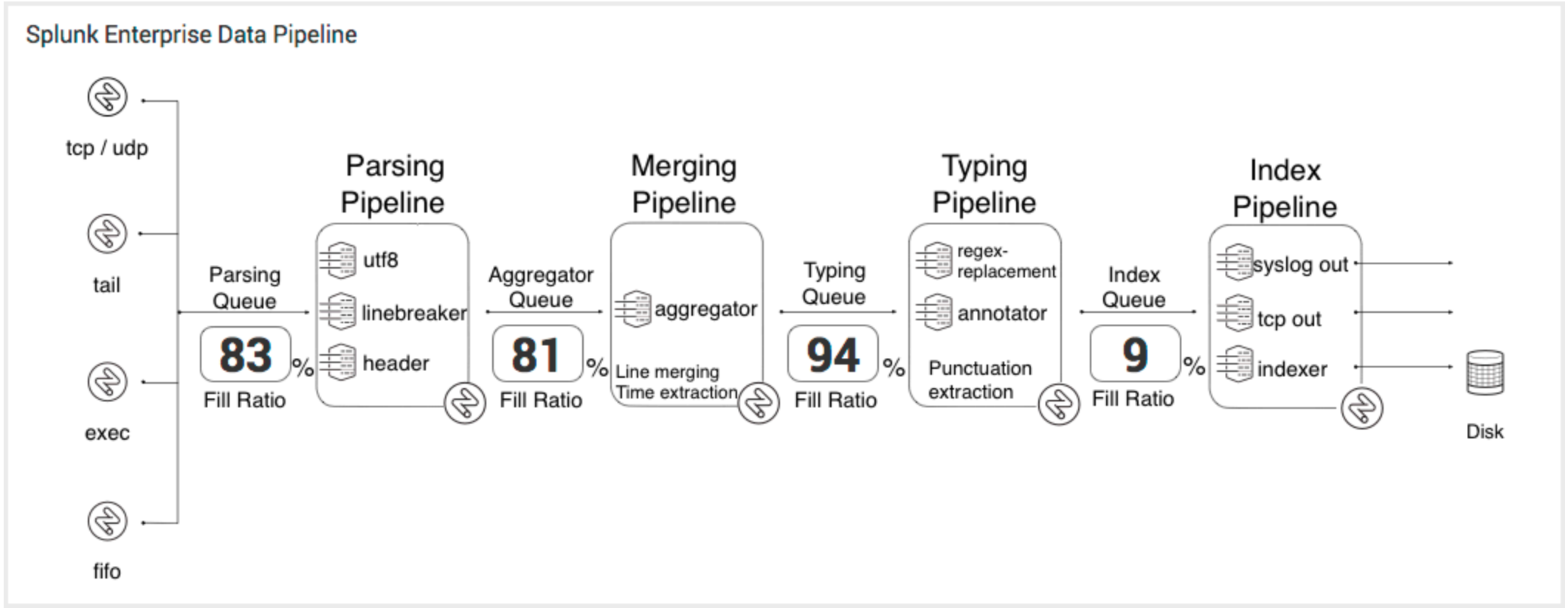
Pipelines/Processors (Debugging)



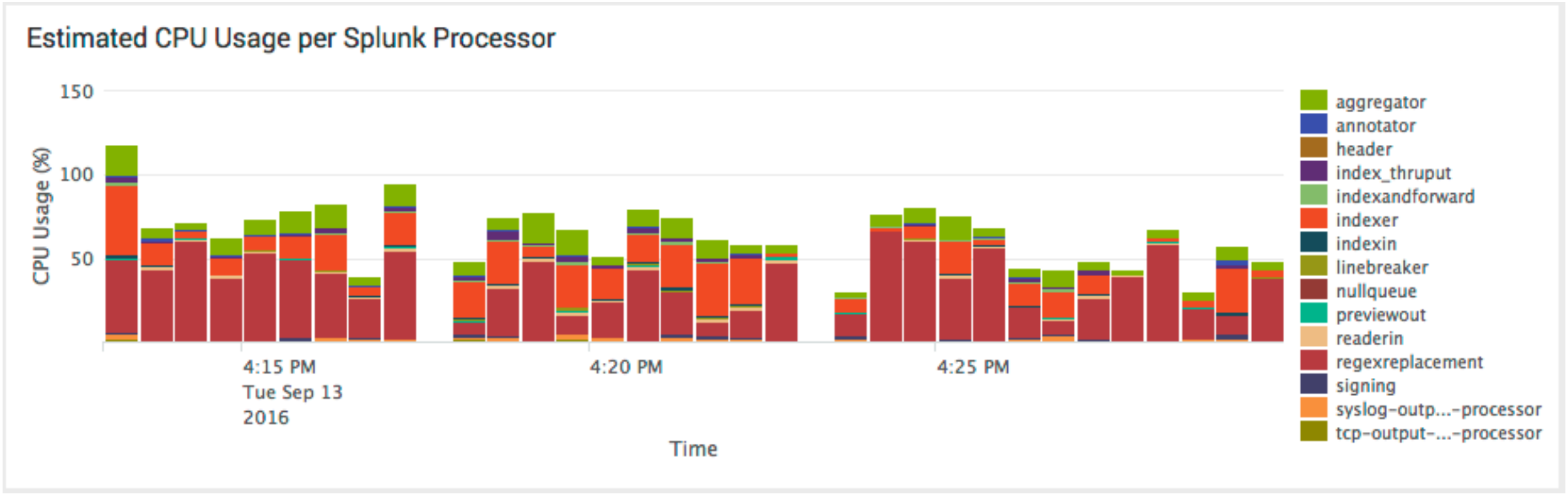
DMC/Monitoring Console – Indexing Performance



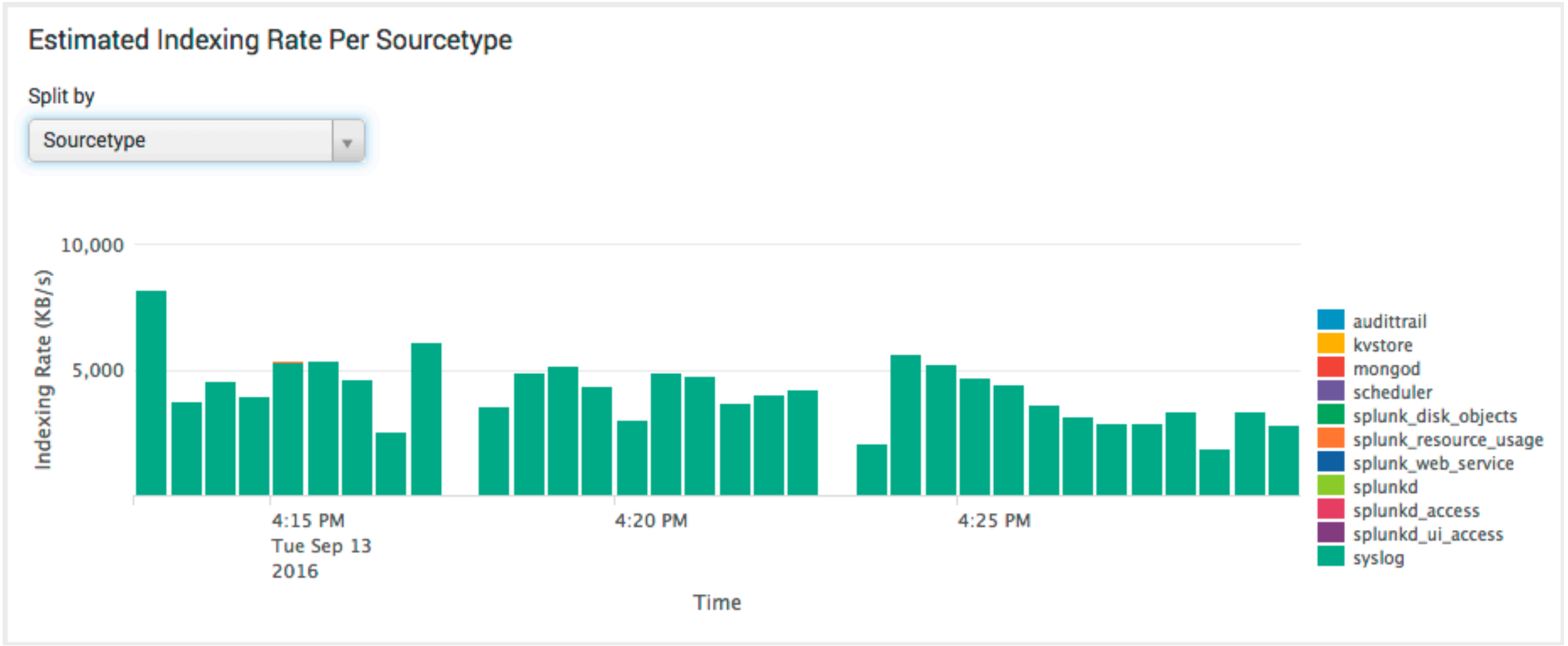
DMC/MC – Queues/Pipelines



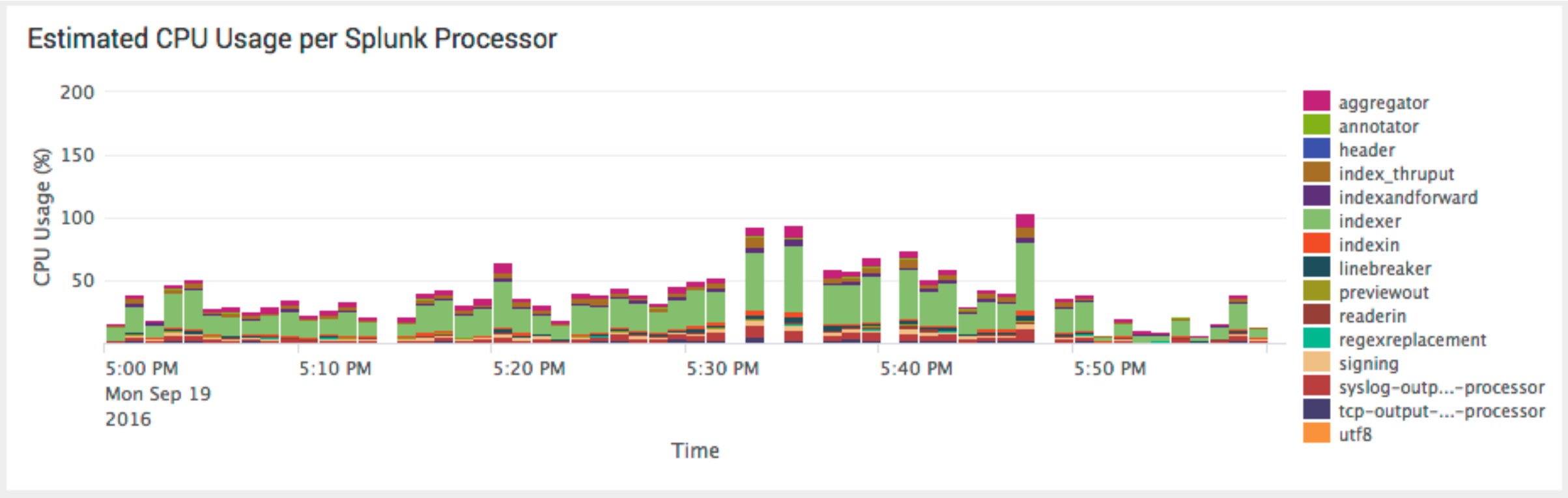
DMC/MC – CPU per Processor (Unhealthy)



DMC/MC – Indexing Rate



DMC/MC - CPU per Processor (Healthy)

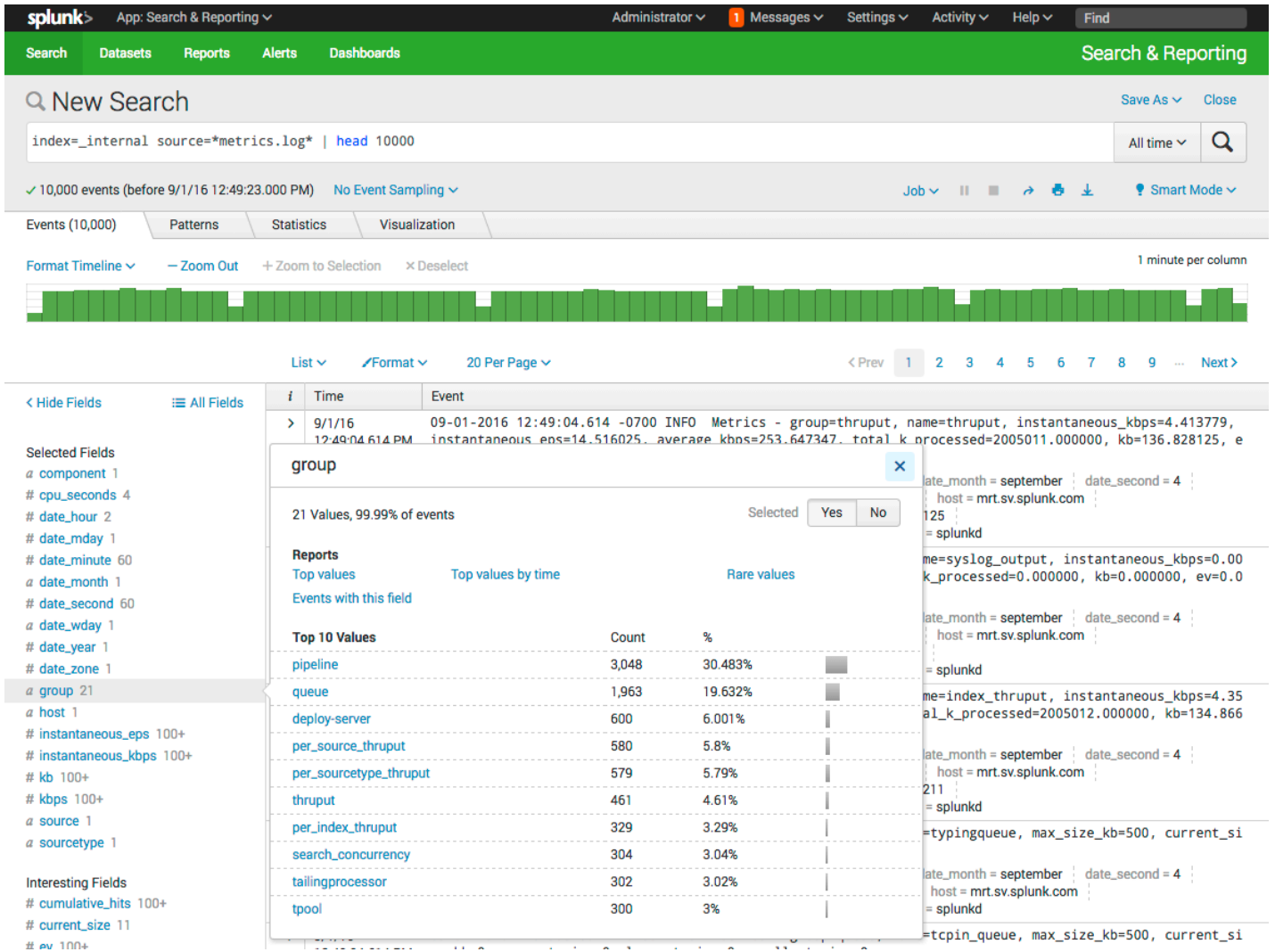


metrics.log

- Search:

  index=_internal
  source=*metrics.log*

- Groups
  - pipeline
  - queue
  - thruput
  - per_source_thruput
  - per_sourcetype_thruput
  - per_index_thruput
  - per_host_thruput
  - tcpin_connections
  - tcpout_connections
  - search_concurrency
  - clusterout_connections
  - deploy-server
  - tailingprocessor

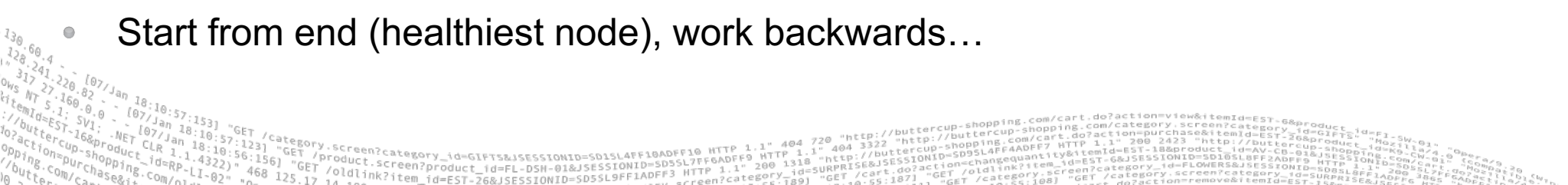


metrics.log: SPL searches

- Queue Sizes:
  - index=_internal group=queue
    | eval pc = (current_size_kb*100) / max_size_kb
    | timechart perc90(pc) by name
- CPU Usage by Splunk Processor:
  - index = _internal group=pipeline processor!=sendout
    | timechart perc90(cpu_seconds) by processor
- Throughput:
  - index=_internal source=*metrics.log* group=thruput
    | timechart per_second(kb)

metrics.log: Scenarios

- Indexing instance: Index Queue at 100%
  - Forwarding disabled:
    - Indexing rate: High? Lots of data. Low? Check cpu_seconds, iowait, etc. Zero? Disk full!
  - Forwarding enabled: See above, on downstream indexers.
  - Forwarding enabled, and downstream indexers are fine?
    - TCPOut rate: High? Lots of data. Low? See outputs.conf maxKBps, check bandwidth & latency…
      Zero? Check SSL, ports, splunkd.log…
    - Also indexing locally? See “Forwarding disabled” above…
- Universal Forwarder
  - Similar, but: Parsing Queue instead of Index Queue.
- Start from end (healthiest node), work backwards…



metrics.log: Universal Forwarder

- No indexing/searching capability
- Can't forward logs to indexers if forwarding is busted
- Fallback to raw file (grep!)

1) $ grep group=queue metrics.log | grep --color 'max_size.*current_size_kb[^,]*,'

   Metrics - group=queue, name=typingqueue, blocked=true, max_size_kb=500,

   current_size_kb=499, current_size=1821, largest_size=1821, smallest_size=0

2) $ grep group=tcpout_connections metrics.log

   … destIp=10.159.4.67, destPort=9997 … _tcp_KBps=27674.87 …

   _tcp_Kprocessed=802571

3) $ grep -E '(ERROR|WARN|FATAL|CRIT)' splunkd.log

4) $ splunk help list inputstatus

Older forwarders: http://blogs.splunk.com/2011/01/02/did-i-miss-christmas-2/

Recap

- Splunk instance consists of linear pipelines
- Splunk topology emulates pipelines
- Downstream slowdown results in upstream blockage
- metrics.log across the topology reveals the whole picture
  - Queue sizes
  - Indexing thruput
  - Forwarding thruput
  - CPU usage per PipelineData Processor
- This is how you should debug – the same way we do!

Questions?