How to Meta-Sumo

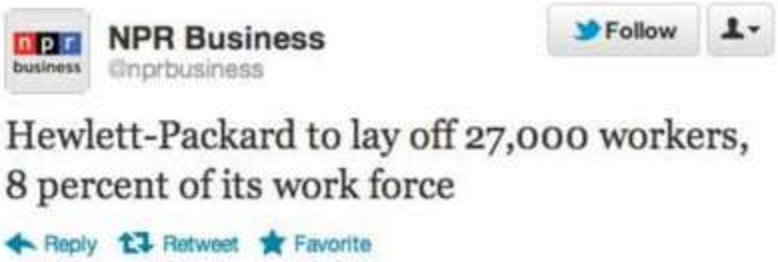
Using Logs for Agile Monitoring of Production Services

Christian Beedgen
christian@sumologic.com
@raychaser

Who Am I?

- Co-Founder & CTO, Sumo Logic since 2010
  - Cloud-based log management and analytics
  - Applications, operations, security
- Server guy, Chief Architect, ArcSight, 2001 – 2009
  - Major SIEM player in the enterprise space
  - Log management for security and compliance

Yo GlueCon!!@!!211



Yo GlueCon!!@!!211

Logs & Logging

What are Logs?

- The murmurings and whispers of your infrastructure
  - Devices & Services (ex. Security, Email, Authentication)
  - Applications (your code, or 3rd party applications)
- Written to disk by applications
  - Used during development by developers
  - And then often ignored in production
- Yes, Logs are Big Data
  - There's a lot of Logs (→ Volume)
  - A large Variety of formats, plus it's real-time (→ Velocity)
- Free-form text, usually one message per line
  - Scrolls by in your terminal
  - Makes you feel like you're in the matrix

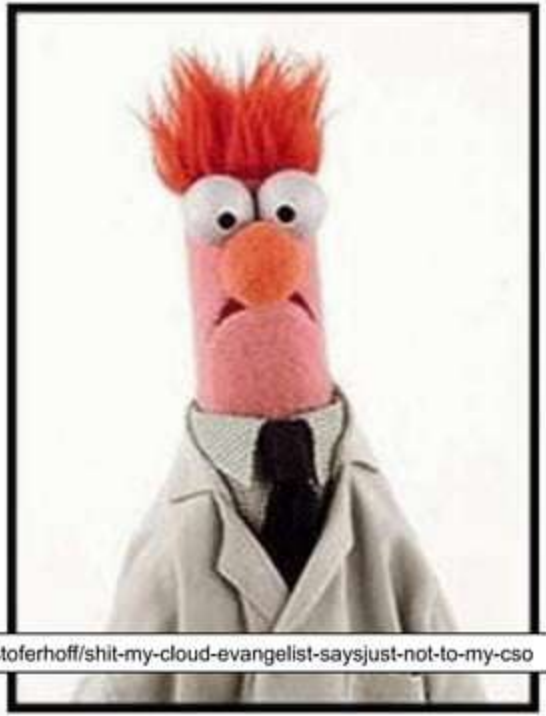
Application Logging

- Just do it*
  - Use what's common in your language (Log4J, ...)
  - If in doubt, log it - you might need it later, and disk is cheap
- The obvious stuff
  - Always log a precise timestamp
  - Add a log level for filtering
- Building a distributed system?
  - Always add the process/service/module name
  - Add the host ID if you can
- Log the context
  - When processing a request, remember who it is from
  - Within the scope of the request, always log the context
- Bring it all together in a single place
  - Open Source options
  - Commercial options

Just Do It*

- Actually, please do think first
  - Do not log ***passwords***
  - Do not log ***credit card numbers***
  - Do not log ***anything that's PII***

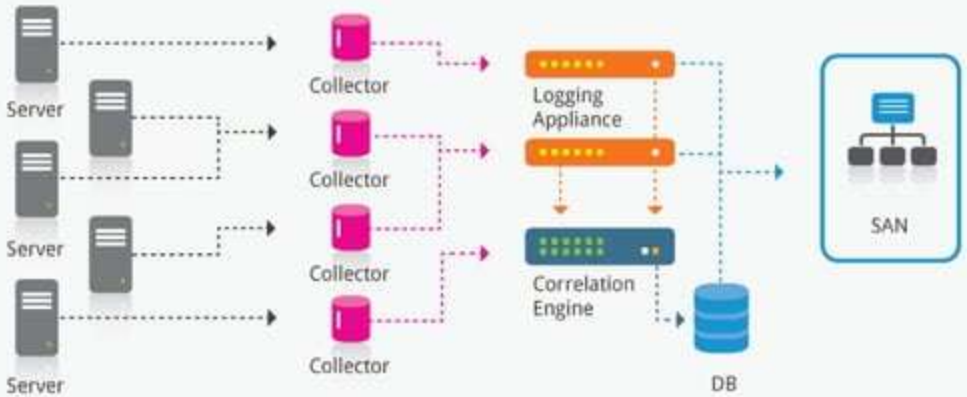
Or Else…



http://www.slideshare.net/christoferhoff/shit-my-cloud-evangelist-saysjust-not-to-my-cso

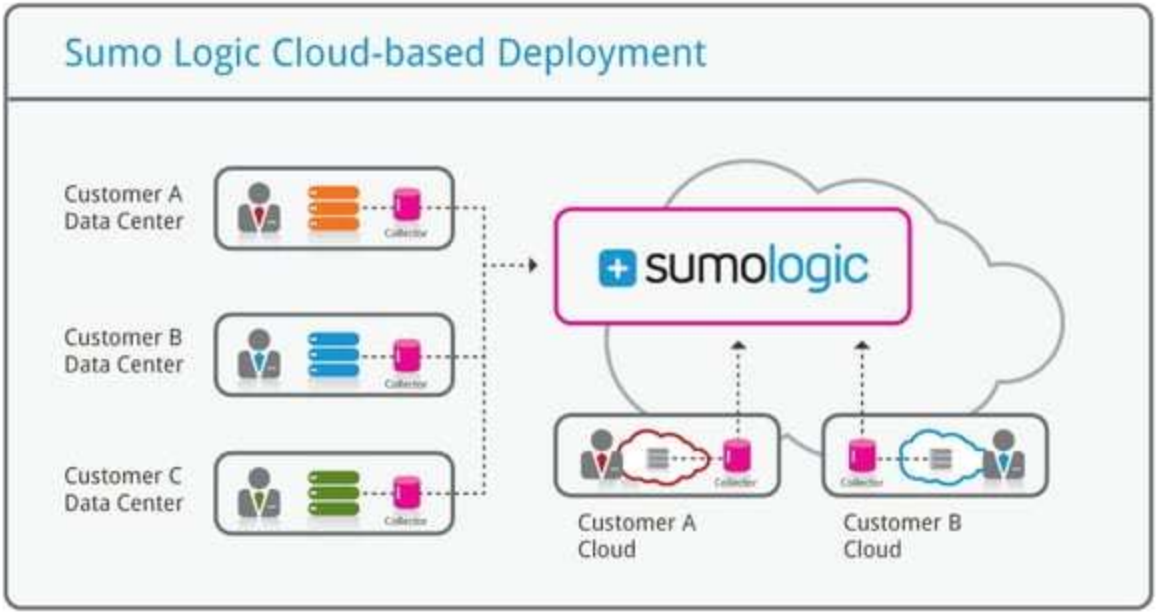
Log Management

Traditional On-premise Deployment



sumologic

Log Management



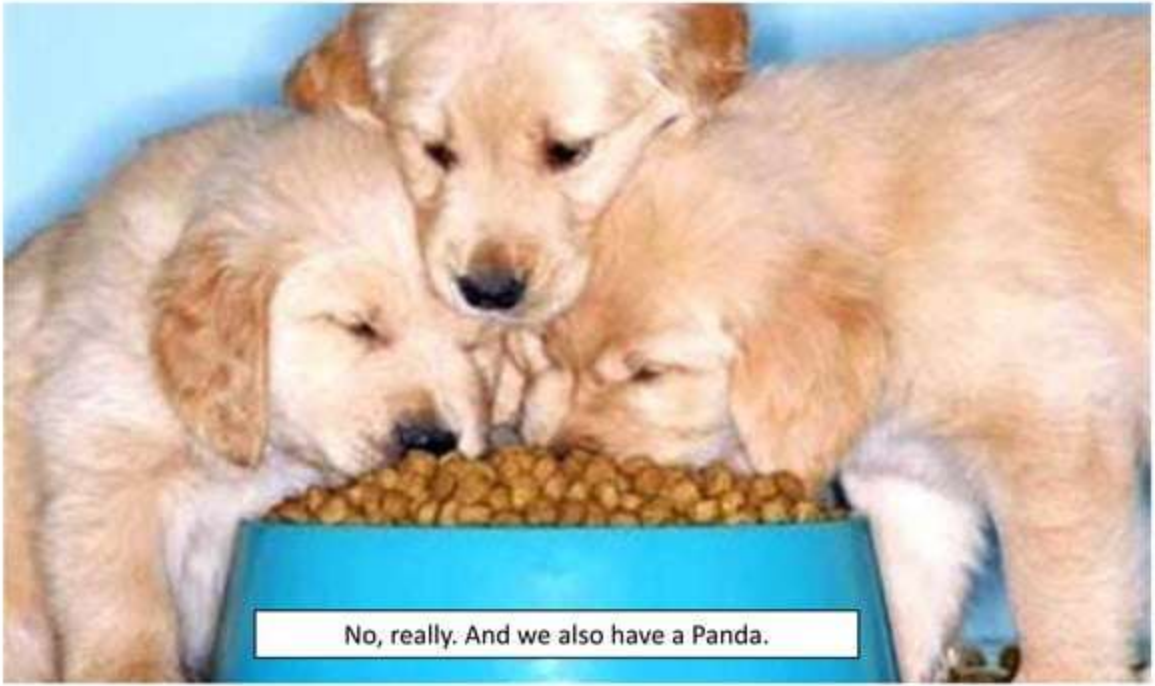
Sumo Logs

This is the Meta part

- Our system to collect and analyze Logs is actually a distributed, cloud-based infrastructure that itself generates a ton of logs
- We are running a second, smaller instance that receives all the logs from the **Prod** system – we call this our ***Shadow*** system
- A majority of the management and monitoring of the **Prod** system is done via the ***Shadow*** system, which we are madly in love with

My team

My team



Example

```
2012-05-22 18:47:26,807 -0700 INFO  [hostId=long-frontend-1] [module=RECEIVER]
[logger=scala.receiver.MessageBlocker] [thread=MTP-MessagePilePipeline-3]
[auth=Collector:prod-cass-raw-8:000000000000483D:0000000000000005:false]
[remote_ip=184.73.74.54] [web_session=MepMG8CS...] Pile for customer:
'0000000000000005', ID: '8000000064076378', block: '80000000004C9A11', msg
count: '1', size: '264', collector: '000000000000483D'
```

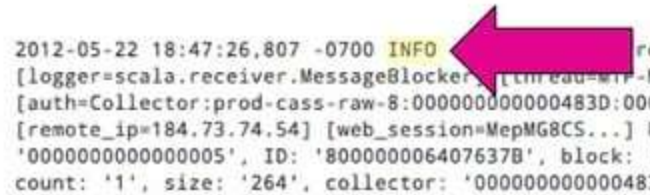
Example

```
2012-05-22 18:47:26,807 -0700                    long-frontend-1] [module=RECEIVER]
[logger=scala.receiver.MessageB      ker] [thread=MTP-MessagePilePipeline-3]
[auth=Collector:prod-cass-raw-8:000000000000483D:0000000000000005:false]
[remote_ip=184.73.74.54] [web_session=MepMG8CS...] Pile for customer:
'0000000000000005', ID: '8000000064076378', block: '80000000004C9A11', msg
count: '1', size: '264', collector: '000000000000483D'
```

- Timestamp with time zone!

Example

```
2012-05-22 18:47:26,807 -0700 INFO                    rontend-1] [module=RECEIVER]
[logger=scala.receiver.MessageBlocker  [thread=MP-MessagePilePipeline-3]
[auth=Collector:prod-cass-raw-8:000000000000483D:0000000000000005:false]
[remote_ip=184.73.74.54] [web_session=MepMG8CS...] Pile for customer:
'0000000000000005', ID: '8000000064076378', block: '80000000004C9A11', msg
count: '1', size: '264', collector: '000000000000483D'
```



- Timestamp with time zone!
- Log level

Example

```
2012-05-22 18:47:26,[illegible] [hostId=long-frontend-1] [module=RECEIVER]
[logger=scala.receiver.MessageBlocker] [thread=MTP-MessagePilePipeline-3]
[auth=Collector:prod-cass-raw-8:000000000000483D:0000000000000005:false]
[remote_ip=184.73.74.54] [web_session=MepMG8CS...] Pile for customer:
'0000000000000005', ID: '8000000064076378', block: '80000000004C9A11', msg
count: '1', size: '264', collector: '000000000000483D'
```

- Timestamp with time zone!
- Log level
- Host ID & module name (process/service)

Example

```
2012-05-22 18:47:26,807 -0700 INFO  [host=along-frontend-1] [module=RECEIVER]
[logger=scala.receiver.MessageBlocker]                      ssagePilePipeline-3]
[auth=Collector:prod-cass-raw-8:0000000000483D:0000000000000005:false]
[remote_ip=184.73.74.54] [web_session=MepMG8CS...] Pile for customer:
'0000000000000005', ID: '8000000064076378', block: '80000000004C9A11', msg
count: '1', size: '264', collector: '000000000000483D'
```

- Timestamp with time zone!
- Log level
- Host ID & module name (process/service)
- Code location or class

Example

```
2012-05-22 18:47:26,807 -0700 INFO  [hostId=long-frontend-1] [module=RECEIVER]
[logger=scala.receiver.MessageBlocker] [thread=MTP-MessagePilePipeline-3]
[auth=Collector:prod-cass-raw-8:000000000000483D:0000000000000005:false]
[remote_ip=184.73.74.54] [web_session=MepMG8CS...] Pile for customer:
'0000000000000005', ID: '8000000006407637B', block: '80000000004C9A11', msg
count: '1', size: '264', collector: '000000000000483D'
```

- Timestamp with time zone!
- Log level
- Host ID & module name (process/service)
- Code location or class
- Authentication context

Example

```
2012-05-22 18:47:26,807 -0700 INFO  [hostId=long-frontend-1] [module=RECEIVER]
[logger=scala.receiver.MessageBlocker] [thread=MTP-MessagePilePipeline-3]
[auth=Collector:prod-cass-raw-8:000000000000483D:0000000000000005:false]
[remote_ip=184.73.74.54] [web_session=MepMG8CS...] Pile for customer:
'0000000000000005', ID: '8000000064076378', block: '80000000004C9A11', msg
count: '1', size: '264', collector: '000000000000483D'
```

- Timestamp with time zone!
- Log level
- Host ID & module name (process/service)
- Code location or class
- Authentication context
- Key-value pairs

What about structure?

- Should you rather use JSON, say?
  - Hey, it's really up to you – machines love it
  - But do humans read the logs as well?
- Modern tools extract structure if there is structure
  - Even if the structure doesn't conform to a protocol
  - So you don't strictly have to use a structured format

Sumo Pet Tricks

Sumo Pet Trick #1 – Basic Troubleshooting

- All errors in the application have an error instance ID

- UI always tries very hard to display the ID
  - User can easily copy/paste the error ID into a ticket

Sumo Pet Trick #1 – Basic Troubleshooting

- Find AP2V3-HZICT-BP6UO in the logs!

```
2012-05-08 11:00:41,974 -0700 ERROR [hostId=prod-frontend-5] [module=SERVICE] [logger=service.util.exception.ExceptionHandlingInterceptor] [thread=btpool0
[auth=User:                    :                    :                    :false] [remote_ip=            ] [web_session=1htzfzle...] Error ID 'AP2V3-HZICT-BP6UO' while
'ReflectiveMethodInvocation: public abstract com.sumologic.service.endpoint.user.v1.model.AddUserResult
com.sumologic.service.endpoint.user.v1.api.UserManagementService.addUser(java.lang.String,com.sumologic.service.endpoint.user.v1.model.User
[com.sumologic.service.endpoint.user.v1.impl.UserManagementServiceImpl]'
java.lang.RuntimeException: java.util.IllegalFormatConversionException: c != java.lang.String
        at com.sumologic.service.endpoint.util.v1.model.ServiceUtils$class.safe(ServiceUtils.scala:37)
        at com.sumologic.service.endpoint.user.v1.impl.UserManagementServiceDelegate.
                                                    UserManagementServiceDelegate.scala:29)
        at com.sumologic.service.endpoint.user.v1.impl.UserManagementServiceDelegat
                                                    llyGetUserObject(UserManagementServiceDelegate.
                                                                                    468)
        at com.sumologic.service.endpoint.user.v1.impl.UserManagementServiceDelegate
                                                    $userExistsInCurrentOrg$1.apply(UserManagement
                                                                                    eDelegate.sc
        at com.sumologic.service.endpoint.user.v1.impl.UserManagementServiceDelegate
                                                    fun$userExistsInCurrentOrg$1.apply(UserManagement
                                                                                    eDelegate.sc
        at com.sumologic.scala.config.graphORM.Transaction$$anonfun$transaction$1.ap
                                                    unsaction.scala:680)
        at com.sumologic.util.scala.Metrics$$anonfun$withGlobalTimer$1.apply(Metrics
                                                    :143)
        at com.yammer.metrics.Timer.time(Timer.scala:20)
        at com.sumologic.util.scala.Metrics$class.withGlobalTimer(Metrics.scala:136)
        at com.sumologic.scala.config.graphORM.Transaction$.withGlobalTimer(Transact
                                                    ala:29)
        at com.sumologic.util.scala.Metrics$$anonfun$withOrganizationTimer$1.apply(M
                                                    .scala:112)
        at com.yammer.metrics.Timer.time(Timer.scala:20)
        at com.sumologic.util.scala.Metrics$class.withOrganizationTimer(Metrics.scala:111)
        at com.sumologic.scala.config.graphORM.Transaction$.withOrganizationTimer(Transaction.scala:29)
        at com.sumologic.util.scala.Metrics$class.time(Metrics.scala:29)
        at com.sumologic.scala.config.graphORM.Transaction$.time(Transaction.scala:29)
        at com.sumologic.scala.config.graphORM.Transaction$.transaction(Transaction.scala:657)
        at com.sumologic.service.endpoint.user.v1.impl.UserManagementServiceDelegate.userExistsInCurrentOrg(UserManagementServiceDelegate.scala:150)
        at com.sumologic.service.endpoint.user.v1.impl.UserManagementServiceDelegate.addUser(UserManagementServiceDelegate.scala:205)
        at com.sumologic.service.endpoint.user.v1.impl.UserManagementServiceImpl.addUser(UserManagementServiceImpl.java:49)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
        at java.lang.reflect.Method.invoke(Method.java:597)
```

Sumo Pet Trick #2 – Everything breaks

- Distributed systems → lots of different components
  - There will be fail
  - The morning coffee routine…
- There is beauty in numbers
  - Pull out all the logs that contain the term "error" (or "fail", or "exception", or all of the above)
  - Then count by process/service/module, or host ID
- Even the most basic aggregation will create actionable insight

Sumo Pet Trick #2 – Everything breaks

```
error | parse "hostId=*]" as hostId | count as error_count by hostId | sort error_count
```

#	hostid	error_count
1	long-config-1	1,633
2	long-ftsearch-2	298
3	long-ftsearch-3	236
4	long-ftsearch-1	201
5	long-cass-raw-3	64
6	long-cass-raw-1	57
7	long-cass-raw-2	47
8	long-frontend-2	27
9	long-frontend-1	22
10	long-cass-meta-3	18
11	long-cass-meta-1	18
12	long-cass-meta-2	17
13	long-bill-1	1

#	module	error_count
1	CONFIG	1,645
2	SEARCH	425
3	STREAM	315
4	RAW	170
5	META	54
6	SERVICE	49
7	BILL	1

```
error | parse "module=*]" as module | count as error_count by module | sort by error_count
```

Sumo Pet Trick #2 – Everything breaks

```
error | parse "hostId=*]" as hostId | count as error_count by hostId | sort error_count
```

#	hostid	error_count
1	long-config-1	1,633
2	long-ftsearch-2	298
3	long-ftsearch-3	236
4	long-ftsearch-1	201
5	long-cass-raw-3	64
6	long-cass-raw-1	57
7	long-cass-raw-2	47
8	long-frontend-2	27
9	long-frontend-1	22
10	long-cass-meta-3	18
11	long-cass-meta-1	18
12	long-cass-meta-2	17
13	long-bill-1	1

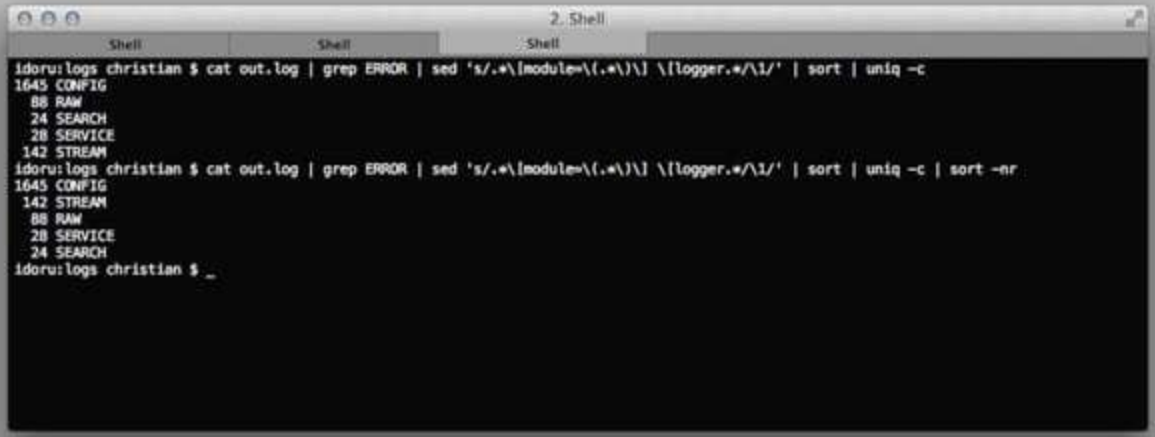
Schedule this!

#	module	error_count
1	CONFIG	1,645
2	SEARCH	425
3	STREAM	315
4	RAW	170
5	META	54
6	SERVICE	49
7	BILL	1

```
error | parse "module=*]" as module | count as error_count by module | sort by error_count
```

Sumo Pet Trick #2 – Everything breaks

- Old school can rock this too!



Sumo Pet Trick #3 – API Choke Points

- What is my API doing?
  - Which APIs are being called?
  - Average execution times per API
  - Who's calling?

```
5:42,279 -0700 INFO  [hostId=long-frontend-2] [module=SERVICE] [logger=service.util.interceptor.Loggi
-14] [auth=User:peter@demo.com:000000000000195C:0000000000000005:false] [remote_ip=98.248.40.103] [we
flectiveMethodInvocation: public abstract com.sumologic.service.endpoint.search.v2.model.GetMessagesR
ervice.endpoint.search.v2.api.SearchQueryService.getMessages(java.lang.String,java.lang.String,java.l
ass [com.sumologic.service.endpoint.search.v2.impl.SearchQueryServiceImpl]', arguments:
37AE93195A4,raw,1211,15,true', elapsed: '73'
```

Sumo Pet Trick #3 – API Choke Points

- Number of calls by API

#	api	_count
1	search.v1.api.SearchService.getStatus	36,799
2	collector.v1.api.CollectorService.getHistogramForCollector	1,110
3	search.v2.api.SearchQueryService.getSearchQueryStatus	1,089
4	search.v1.api.SearchService.startQuery	582
5	auth.v1.api.AuthenticationService.logOut	581
6	search.v1.api.SearchService.getMessages	300
7	search.v1.api.SearchService.cancelQuery	300
8	collector.v1.api.CollectorService.getCollectorsWithBlades	216
9	account.v1.api.UserPreferencesService.updateInitialTab	149
10	search.v2.api.SearchQueryService.createSearchQuery	112
11	account.v1.api.UserPreferencesService.getUserPreferences	102
12	auth.v1.api.AuthenticationService.getCurrentUserInfo	95
13	account.v1.api.AccountService.getSubscriptionDetails	93
14	search.v2.api.SearchQueryService.deleteSearchQuery	91

Sumo Pet Trick #3 – API Choke Points

- Average execution time by API

#	api	avg_ms
1	collector.v1.api.CollectorService.getHistogramForCustomer	1,405
2	collector.v1.api.CollectorService.getHistogramForCollector	1,199
3	search.v1.api.SavedSearchService.getSearches	590
4	account.v1.api.AccountService.getSubscriptionAndBillingUsages	547
5	user.v1.api.UserManagementService.listUsers	319
6	account.v1.api.AccountService.getSubscriptionDetails	247
7	collector.v1.api.CollectorService.collectorDownloadOfferings	173
8	search.v2.api.SearchQueryService.createSearchQuery	170
9	search.v1.api.SearchService.getMessages	166
10	search.v1.api.SearchService.startQuery	136
11	collector.v1.api.CollectorService.getCollectorsWithBlades	110
12	auth.v1.api.AuthenticationService.getCurrentUserInfo	91
13	search.v2.api.SearchQueryService.parseSearchQuery	91
14	collector.v1.api.CollectorService.createWildcardLocalFileTailBlade	78

Sumo Pet Trick #3 – API Choke Points

- Number of API calls by user

#	user	api	_count
1	test@demo.com	search.v1.api.SearchService.getStatus	37,956
2	mandy.liu@demo.com	collector.v1.api.CollectorService.getHistogramForCollector	1,044
3	rishi@sumologic.com	search.v2.api.SearchQueryService.getSearchQueryStatus	694
4	test@demo.com	search.v1.api.SearchService.startQuery	594
5	test@demo.com	auth.v1.api.AuthenticationService.logOut	593
6	test@demo.com	search.v1.api.SearchService.cancelQuery	305
7	test@demo.com	search.v1.api.SearchService.getMessages	305
8	james+example@sumologic.com	search.v2.api.SearchQueryService.getSearchQueryStatus	283
9	mandy.liu@demo.com	collector.v1.api.CollectorService.getCollectorsWithBlades	184
10	mandy.liu@demo.com	account.v1.api.UserPreferencesService.updateInitialTab	91
11	peter@demo.com	search.v2.api.SearchQueryService.getSearchQueryStatus	87
12	mandy.liu@demo.com	account.v1.api.UserPreferencesService.getUserPreferences	85
13	rishi@sumologic.com	search.v2.api.SearchQueryService.createSearchQuery	84
14	mandy.liu@demo.com	auth.v1.api.AuthenticationService.getCurrentUserInfo	81

Sumo Pet Trick #4 – Which searches are running

- Log a session ID on start and stop
  - When there's long-running operations
  - More than one node participates in generating the result
- For each session in the time range, did you get both the start and the stop message?

```
"module=SERVICE" SearchServiceImpl ("Started query" or "canceling query")
| parse "auth=User:*:" as user
| parse regex "for session.*?'(?<session>.*?)'"
| parse "query: '*'" as query nodrop
| count, first(query) by session, user
| where _count != 2
```

Sumo Pet Trick #4 – Which searches are running

#	session	user	_count	_first
1	7DF97979DEA03645	test@demo.com	1	etime AND !connect AND !disconnect !QA_EXTENDED_SEARCH_LOG
2	0277A1F1570F4E51	panda@demo.com	1	93025E32B0315B6A _sourcecategory=stream "Raw finished processing request" | parse "messageCount: *" as msg | avg(msg), sum(msg), max(msg)
3	503337586741B2B7	test@demo.com	1	(_sourceHost=prod-ftsearch-4 RawResolverDelegate) AND Received !QA_EXTENDED_SEARCH_LOG | parse "sessionId *," as sessionid|parse "size: *" as size|sum(size) by sessionid
4	111095253288E486	peter@demo.com	1	5F20D09072682F69 concierge executing query | parse "Executing query for stream session:
5	025AFAF5ABD9AB40	peter@demo.com	1	93025E32B0315B6A concierge executing query | parse "Executing query for stream session:
6	6FD7842042552D44	test@demo.com	1	"Summarize operator" !QA_EXTENDED_SEARCH_LOG | parse "(maxDepth=", totalNodes=", maxClusterId=*)" as maxDepth,totalNodes,maxClusterId, "searchExpression=*" as searchExpression | toLong(maxClusterId) | sort by maxClusterId
7	3FAB788A0D505064	test@demo.com	1	error !"Error while removing index" !INFO !"session has already been cancelled" !CassandraMessagesRetriever !"Could not remove pending" !QA_EXTENDED_SEARCH_LOG
8	1A9EB9314A849B6D	rishi@sumologic.com	1	
9	279FFAE94BC4B105	peter@demo.com	1	6156D23ED7ED8BCD SearchServiceImpl creating query | parse "session:
10	0378709A8E1791CD	rishi@sumologic.com	1	_sourceHost=pix | extract "(?%PIX-\d-\d{6})" | count by msg_code
11	3C500CF0915B6EB2	test@demo.com	1	error or fail * !QA_EXTENDED_SEARCH_LOG | summarize
12	676A1FF7FF22AB93	peter@demo.com	1	(2E2B0A1F5D40347D or 0B2CD7B49451BA47 or 96A207EA7D7E606D or 93025E32B0315B6A or 4AAA86552FC4334C)
13	6D85CC5E0D5EE63F	test@demo.com	1	"logger=org.quartz.core.ErrorLogger" "failed to execute task" !QA_EXTENDED_SEARCH_LOG | parse regex "*\nCaused by: java(?.*?)\n" nodrop | parse regex "*\nCaused by: java(?.*?) sessionId: .*?\n" nodrop | parse regex "*\nCaused by: java.util.concurrent.ExecutionException: java.lang.RuntimeException: Unable to run query .*\. (?.*?) \(\1\)\n" nodrop | count by cause | sort _count

Sumo Pet Trick #4 – Which searches are running

#	session	user	_count	_first
1	7DF97979DEA03645	test@demo.com	1	etime AND !connect AND !disconnect !QA_EXTENDED_SEARCH_LOG
2	0277A1F1570F4E51	panda@demo.com	1	93025E32B0315B6A _sourcecategory=stream "Raw finished processing request" | parse "messageCount: *" as msg | avg(msg), sum(msg), max(msg)
3	503337586741B2B7	test@[redacted]vm	1	(_sourceHost=prod-ftsearch-4 RawResolverDelegate) AND Received !QA_EXTENDED_SEARCH_LOG | parse "sessionId *," as sessionid|parse "size: *" as size|sum(size) by sessionid
4	111095253288E486	peter[redacted].com	1	5F20D09072682F69 concierge executing query | parse "Executing query for stream session:
5	025AFAF5ABD9AB40	peter[redacted].com	1	93025E32B0315B6A concierge executing query | parse "Executing query for stream session:
6	6FD7842042552D44	test@[redacted]com	1	"Summarize operator" !QA_EXTENDED_SEARCH_LOG | parse "(maxDepth=", totalNodes=", maxClusterId=")" as maxDepth,totalNodes,maxClusterId, "searchExpression=*" as searchExpression | toLong(maxClusterId) | sort by maxClusterId
7	3FAB788A0D505064	test@demo.com	1	error !"Error while removing index" !INFO !"session has already been cancelled" !CassandraMessagesRetriever !"Could not remove pending" !QA_EXTENDED_SEARCH_LOG
8	1A9EB9314A849B6D	rishi@sumologic.com	1	
9	279FFAE94BC4B105	peter@demo.com	1	6156D23ED7ED8BCD SearchServiceImpl creating query | parse "session:
10	0378709A8E1791CD	rishi@sumologic.com	1	_sourceHost=pix | extract "(?%PIX-\d-\d{6})" | count by msg_code
11	3C500CF0915B6EB2	test@demo.com	1	error or fail * !QA_EXTENDED_SEARCH_LOG | summarize
12	676A1FF7FF22AB93	peter@demo.com	1	(2E2B0A1F5D40347D or 0B2CD7B49451BA47 or 96A207EA7D7E606D or 93025E32B0315B6A or 4AAA86552FC4334C)
13	6D85CC5E0D5EE63F	test@demo.com	1	"logger=org.quartz.core.ErrorLogger" "failed to execute task" !QA_EXTENDED_SEARCH_LOG | parse regex "*\nCaused by: java(?.*?)\n" nodrop | parse regex "*\nCaused by: java(?.*?) sessionId: .*?\n" nodrop | parse regex "*\nCaused by: java.util.concurrent.ExecutionException: java.lang.RuntimeException: Unable to run query .*\. (?.*?) \(\1\)\n" nodrop | count by cause | sort _count

Thank you!

https://www.sumologic.com/free-trial/

Christian Beedgen
christian@sumologic.com
@raychaser


http://iam.cat/cat-love-beer