Splunk + AWS:

Two Great Things Even Better Together!

Al Liebl - Splunk

October 2018 | Version 1.0

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> .conf18

Your Tour Guide

AL LIEBL

Senior Sales Engineer

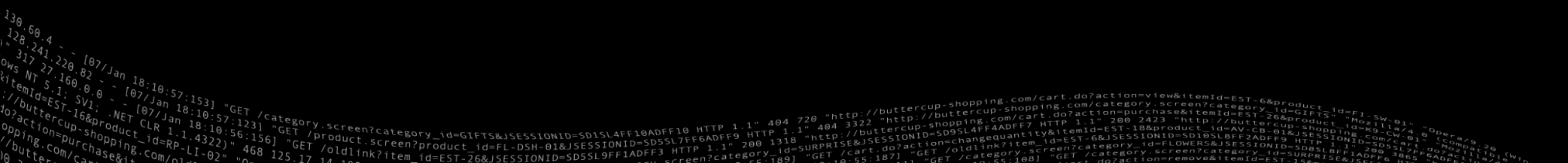
aliebl@splunk.com

splunk> .conf18

Background

Not our First Rodeo

- AWS Incident Response (IR) manages multiple simultaneous investigations each week
  - Investigations can range from a few hundred GB in size to tens of TB or more
- AWS IR usually spins up a new Splunk environment for most new investigations
  - Also have a permanent (non-ephemeral) Splunk environment for on-going work.
- AWS has internal tools to handle all data collection
- Philosophy = move fast and break things but learn from it



Something Happened. We Have a Ton of Data. Now What?

Let the Games Begin!

Deploying Splunk for Data at Scale

splunk> .conf18

Getting the Party Started!

Day 1

- We know this will be a large investigation – we just don't know how large
  - Relevant data requested from internal systems – will take some time to be delivered
- Using the AWS Add-on via Heavy Forwarder for ingest
  - All data to be delivered via S3
- Discussed server options
  - Spin up new HW or build out existing cluster?
- Decided to keep it simple using known environments/tools
  - Build out as we need to
  - Use simplest ingest method

Monday May 14, 2018 - Recap

We don't Know What we don't Know

- We're in a holding pattern until we understand the actual data volume
- We're being conservative while fact finding
- Sounded reasonable at the time…

Blocking and Tackling

Day 2

- Data is being delivered from internal systems
  - Entire data set will be delivered in ~130 files
  - Smallest files ~15 GB in size
  - Largest files currently 1+ TB and still growing
  - Still unsure of total data set size
- Begin onboarding data to the existing cluster
- Prepared to add indexer nodes as needed

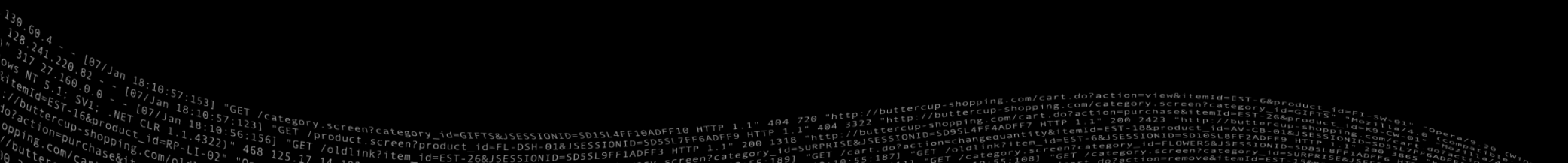
Current cluster specs:

6 c4.8xlarge for indexers

3 c4.8xlarge for search heads

1 c4.2xlarge for CM/LM

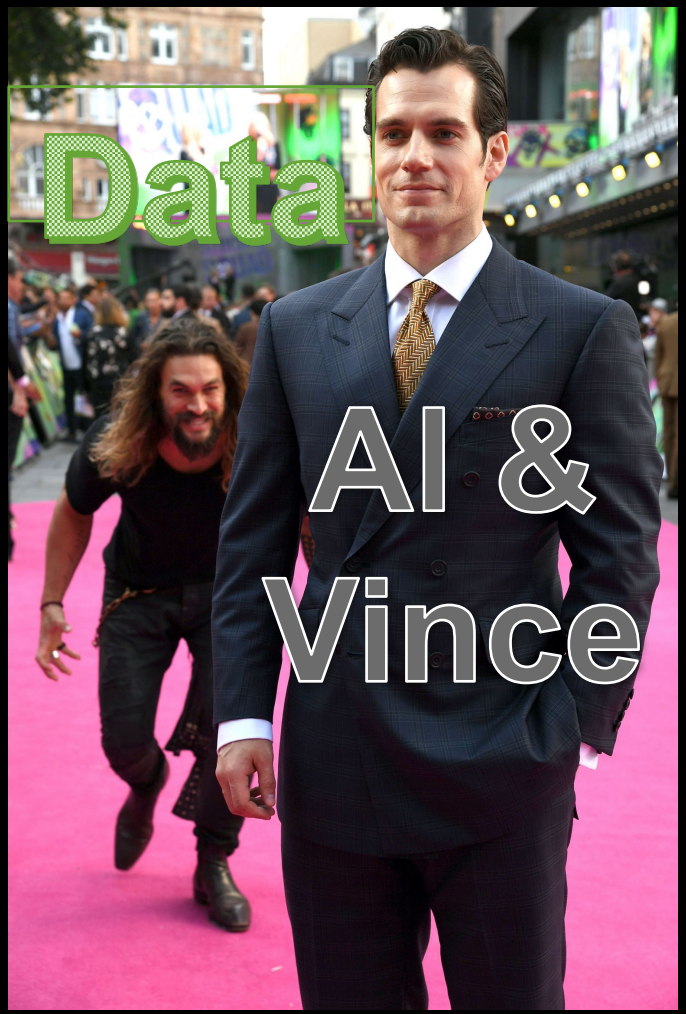
1 c4.2xlarge for HF



Tuesday May 15, 2018 - Recap

Something Doesn't Seem Right…

- Becoming concerned about file size
- Largest files are now 3+TB by EOD
- Suspect final data set will be more than 100 TB
- Ingest is proceeding but slower than desired
- We increase the number of pipelines/inputs on the HF
  - Great idea but… let's talk about those big files



The Default isn't Always the Best

Day 3

- After 24 hours we only have 1.4 TB ingested
- Performance is slow because we used Generic S3 AWS input method
- All requested data has been delivered and totals 115 TB
  - At current rates it will only take 82 days to load the data
- Discuss options for tuning ingest and the ramifications of using the existing environment
  - Need to dramatically modify storage and number of nodes
  - Probably easier to build out a new cluster

Performance with the AWS TA

Getting the Most Bang for your Instance

- When using the AWS TA:
  - When available use SQS based inputs for better scale and built in fault tolerance
    - SQS based approach is stateless and allows for multiple HF
  - Generic S3 works but will be slower and no fault tolerance
    - Generic S3 is stateful and only runs on a single HF
  - High volume data can leverage Lambda functions to HEC
  - Kinesis is great for high volume real-time ingest
- Since the TA needs a HF, increase your pipelines if you have the resources and need to scale

Wednesday May 16, 2018 - Recap

- Generic S3 = bad idea
- SQS based S3 = better idea
- Bigger cluster = more indexers
- More indexers = better search performance & faster ingest times

Things are About to get Real!

Day 4

- Decide to go BIG and build out a new cluster
  - But how many nodes?
  - Automation will be critical
- Need to get data loaded ASAP
  - But what's the best method?
- And the best part…we can use any AWS services/resources we want as long as we move fast!

How Big is Big?

Day 4

- Decide to build out the cluster to support sustained ingest @100TB per day
- Using c4.8xlarge (36 cores, 60 GB RAM) instances
  - 504 indexers – 168 per AZ
  - 3 search heads
  - 1 cluster master
  - 1 license master
  - 1 deployment server
  - 1 monitoring console
- Indexer storage on SSD EBS volumes - total of 2,000,000 provisioned IOPS
  - 7 TB per indexer
  - RF=3, SF=2

Terraform + Ansible != Terrible!

Day 4

- AWS IR has a collection of Terraform and Ansible scripts to automate Splunk deployments
  - We will use as the basis for building out our new cluster
- Terraform for base deployments
  - Used to deploy instances and storage
- Ansible for tricky Splunk modifications
  - Great for distributing .conf files and applying commands en masse
- Use version control with your scripts and you've leveled up!

What About the Data?!

Day 4

- Decide to ingest via Universal Forwarders pulling data from local disk
  - Gives us the most control over ingest performance
  - More forgiving if we need to make changes
- 12 UF using c4.8xlarge (36 cores) - Why so big?
  - Pipelines!
  - Docs indicate you can safely increase pipelines to 2
  - We're going to use 8 – Don't try this at home!
- Begin transferring data from S3 buckets to EBS volumes attached to UF

Thursday May 17, 2018 - Recap

Building for Speed!

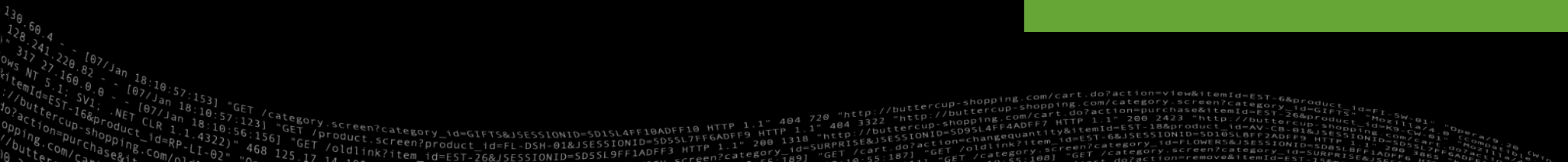
- Why are we using indexer clustering?
  - Clustering is NOT a performance enhancer
  - But having a Cluster Master manage indexer discovery is priceless
- Provisioning and automation will be critical

Server.conf on CM:

[indexer_discovery]
pass4SymmKey = <string>
polling_rate = <integer>
indexerWeightByDiskCapacity = <bool>

Outputs.conf on UF:

[indexer_discovery:<name>]
pass4SymmKey = <string>
master_uri = <uri>
[tcpout:<target_group>]
indexerDiscovery = <name>
[tcpout]
defaultGroup = <target_group>



splunk> .conf18

Attempt #2 or How to DOS yourself

Day 5

- Data still being copied from S3 to UF servers
- We iterate on the deployment scripts and the Cluster deploys
  - We now have 504 indexers hell bent for leather!
  - But we need an index and some props/transforms first
  - Good thing we're in a cluster right?
- Funny thing about updating a 504 node cluster – DIY DOS!
  - The Cluster Master is single threaded AND
  - All nodes ask for updates at the same time by default

Pro-tips for Clustering at Scale

Learn from our Pain!

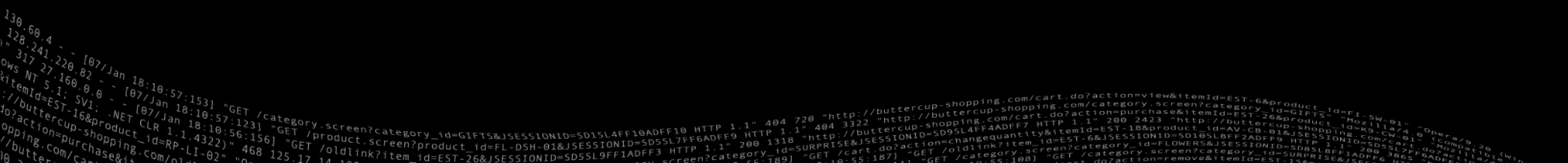
- On the Cluster Master
  - In sever.conf set max_peers_to_download_bundle = some reasonable number like 20
    - Normally used to deal with large bundle sizes but also applies at scale
  - Make sure your apps are clean
    - You shouldn't ignore validation errors even in apps you don't care about
- On the peer nodes
  - Note that indexes won't show up until you've added data
  - In server.conf you can set parallelIngestionPipelines
    - Docs state 2 is safe
    - We're using 8 for maximum speed

splunk> .conf18

Friday May 18, 2018 - Recap

TGIF?!

- First UF has all of it's data but we're fighting with the Cluster Master to replicate configuration bundles
- We decide to let the CM try to sort itself out while we sleep
- Clustering is great but has its' challenges
  - Downside – we're fighting to get simple configuration files replicated
  - Upside – did we mention you also get great data distribution with indexer discovery!
- Pipeline settings on UF and indexers should make things interesting



Getting it Right the First Time for the Second Time

Day 6

- In the morning the the indexers seem good so we start loading some data
- We notice events aren't parsing correctly so we stop ingest
- Props.conf is bad and we're still fighting with the CM
- Clock is ticking so we need a new plan
  - Decide to rebuild AGAIN
  - We pause the automation after the cluster has 3 nodes, add our .conf changes to the CM, and resume
  - Takes 1:45 to complete and everything looks good!

splunk> .conf18

Saturday May 19, 2018 - Recap

Say Rebuild Again! I Dare You!

- Automation + Brute Force = Problem Solved
  - We didn't realize cluster replication issues were due to bad app configurations so we improvised
- AWS + Automation = Awesome!
  - 500+ fully configured instances and 2,000,000 dedicated IOPS provisioned in less than 2 hours!
- Lesson learned in hindsight: Slow down with clustering if you want to go fast

Unleash the Data?

Day 7

- 12:05 AM and the cluster is ready to load
- We fire up the UFs, data starts flowing, and we go to bed
- Midday we have a look and see only 2.4 TB has been ingested.
  - Not good.
  - At this rate it will take about 24 days
- What is going on?!

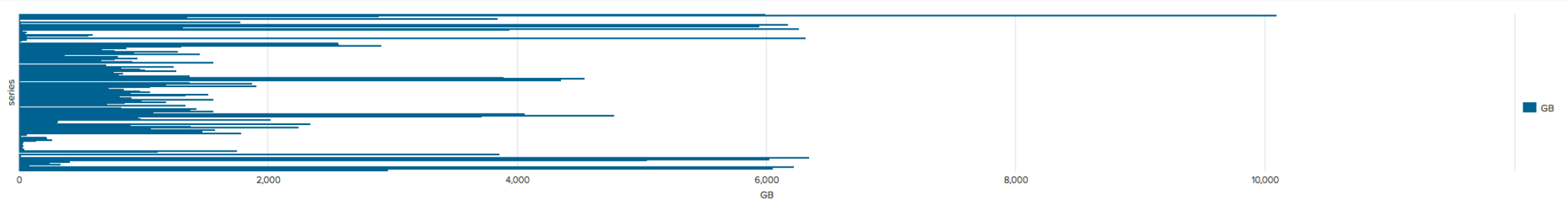
I See What You Did There…

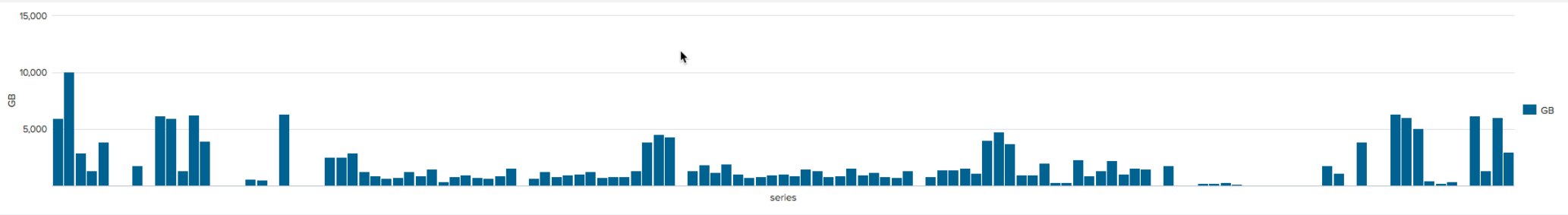
Day 7

- We dig a bit deeper and realize that:
  - We only started one UF before going to bed
  - The UF only had 2 pipelines defined
  - So we saw 2.4 TB of ingest on a single UF with only 2 pipelines in 12 hours!
- So we set pipelines to 8 for each UF and let them rip… Guess what happens next

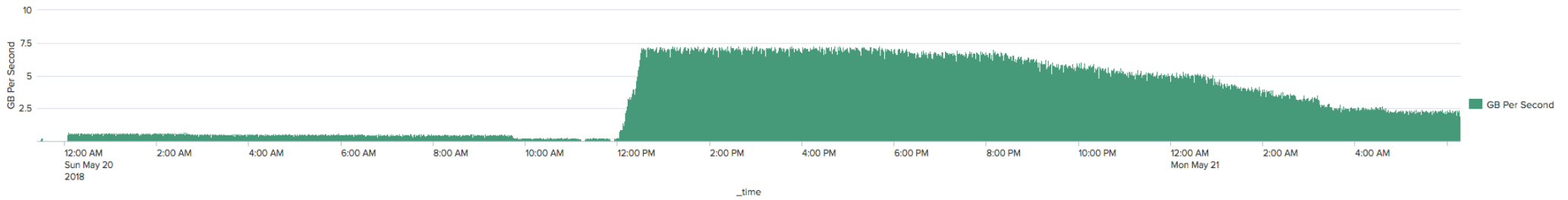
[redacted]

Edit | Export | ...

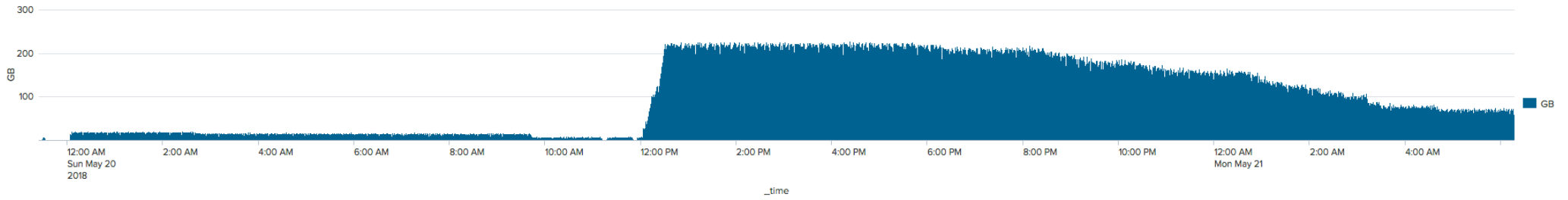




GB Per Second Ingest Rate



Ingest Rate Over Time in GB's



.conf18

Sunday May 20, 2018 - Recap

From Garden Hose to Firehose

- Adding pipelines to both the UF and indexers resulted in massive throughput
  - Nearly 7.5 gigs per second!
- While the data was loading we could see even data distribution across indexers
  - Just check license volume per indexer
- Preliminary searches are flying because map reduce is happy!
  - Good data distribution + many search peers = happy users

How Important is _time?

Day 8

- By 1 PM we have successfully loaded all 115 TB of data!
- Searches are very fast!
- Then we realize something is wrong with the data!
  - It's no big deal, it's only _time
  - There is an extra "%" in the regex for time extraction so extraction fails!
  - Events have been given timestamps based on their file time
  - How well will that work when there are more than 10 files nearly 5 TB in size?!

Monday May 21, 2018 - Recap

Improvise, Adapt, and Overcome?

- Splunk is a time series index
  - Need to be SURE that sourcetypes are properly defined
  - Always validate props.conf before replicating
  - Pro-tip: Default sourcetype definitions are meant to be tuned
    - Indexing performance can be improved more than 2X with minor modifications:
      - Set LINE_BREAKER, TRUNCATE, TIME_PREFIX, MAX_TIMESTAMP_LOOKAHEAD, SHOULD_LINEMERGE=FALSE
      - Avoid BREAK_ONLY_BEFORE

To Rebuild or not to Rebuild, That is the Question

Day 9

- We consider rebuilding the cluster yet again
  - Very difficult (but not impossible) to search without using _time
  - Need to determine if _time is needed for desired analysis
- Good news!
  - We only need to worry about _time if we find certain matches in the data
  - So now all searches are done for all time
  - Good thing we have a performant cluster!
- End result = Success!
  - Nothing responsive in this data set

Tuesday May 22, 2018 - Recap

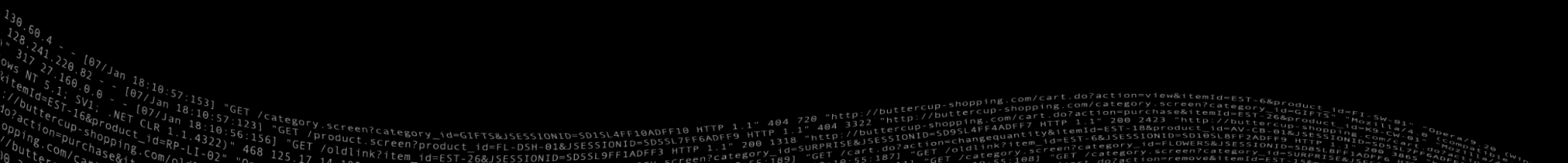
TL;DR: Self inflicted Wound NOT Fatal

- Successfully completed the investigation but it wasn't pretty
- We could have saved time with some Just In Time validation
  - Pro-tip: Create a Docker environment for sourcetype validation!
- Entire investigation took 9 days
- Most work was done via web conference after normal working hours

Lessons Learned

Or How to Avoid Working All Weekend

- Automation is Critical
  - Requires an up front investment but pays off in spades when you need to move fast
  - Document your scripts to make reuse simple
  - Standardize your tools for maximum potential
- Clustering is invaluable
  - Simplified management (e.g. indexer discovery) makes it easy to scale
  - Keep your CM/Apps clean and simple to minimize problems – eliminate cruft!
  - Eliminates concerns regarding data distribution
- JIT validation means breaking less things
  - Validate .conf files before deploying
    - EVEN SIMPLE ONES!
  - Consider Docker for simplified testing
  - Proper version control will make iterating easier



Thank You

Don't forget to rate this session in the .conf18 mobile app