Worst Practices…

and How to Fix Them

Jeff Champagne | Principal Architect, Splunk

October 2018

Who's This Dude?

Jeff Champagne

Principal Architect

jchampagne@splunk.com

- Started with Splunk in the fall of 2014
- Member of the Splunk Architecture Council
- Former Splunk customer in the Financial Services Industry
- Lived previous lives as a Systems Administrator, Engineer, and Architect
- Loves Skiing, traveling, photography, and a good Sazerac

splunk> .conf18

10s, 10s, 10s Across the Board!

Rate My Session Please

Am I in the right place?

You'll find this session helpful if you…

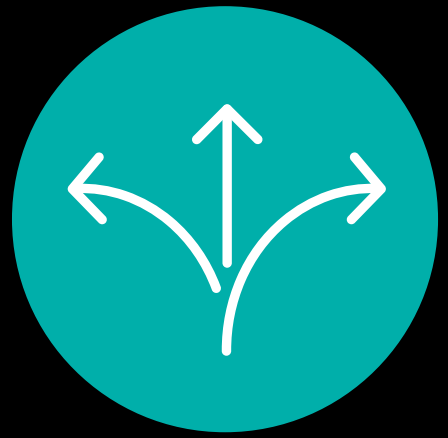
Target Audience: Splunk Admin or Knowledge Manager

- You should be familiar with general Splunk architectures
  - N00bs, you'll learn a lot...but some topics won't be explained in-depth

- Questions you may have…
  - What is the best way to collect my syslog data?
  - Why are my searches running slowly?
  - How can I speed up indexing?
  - Are there limitations to clustering?
  - What are the best practices for HA/DR?

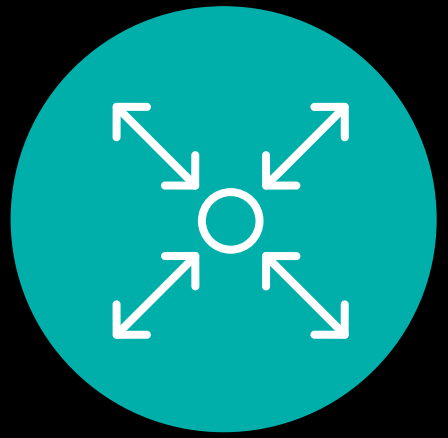
What will I learn?

Agenda



Data Collection

Data Management



Data Resiliency

Lossless Syslog/UDP

Busting The Myth…

"I want to collect 100% of my UDP syslog data"

Lossless data transmission over UDP does not exist

- UDP lacks error control AND flow control
  - Delivery cannot be guaranteed
  - Packets may be lost
    - They never arrived due to network issues
    - They were dropped due to a busy destination
  - Retransmits can result in duplicates
- You can engineer for redundancy
  - Loss can still happen
  - Avoid over-engineering

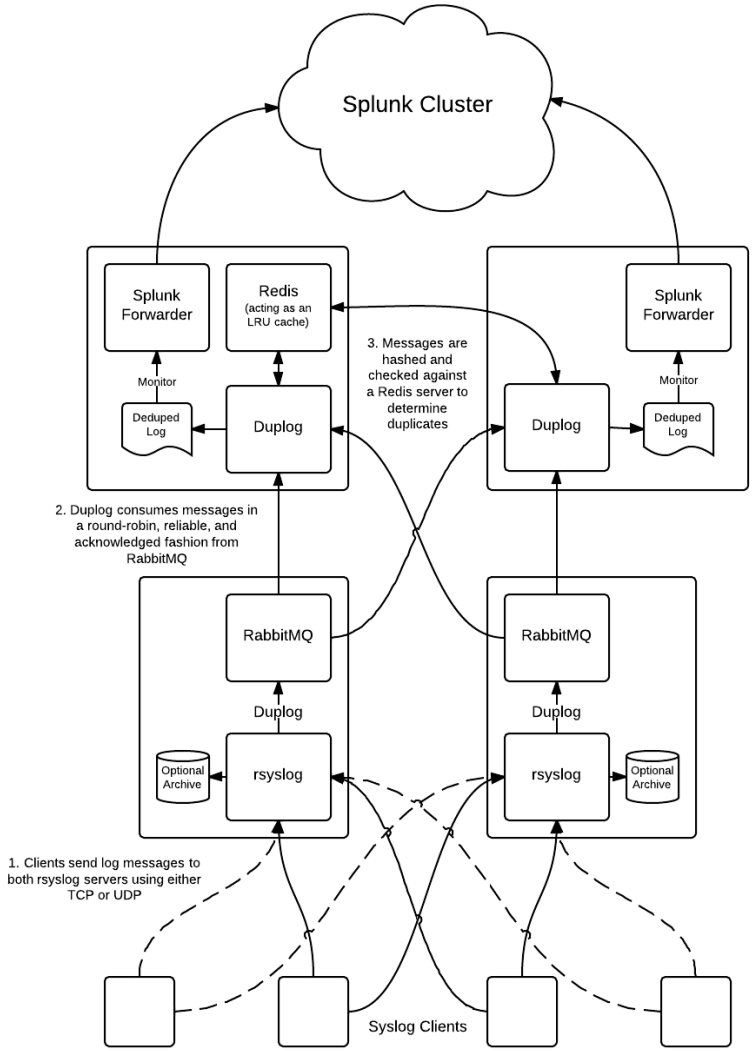
Worst Practice

Over-Engineering

Don’t engineer a solution for syslog that is more complex than Splunk itself!

- Loss of data is still possible
  - UDP does not guarantee delivery…make peace with it

- Design for redundancy while maintaining minimal complexity



Best Practice

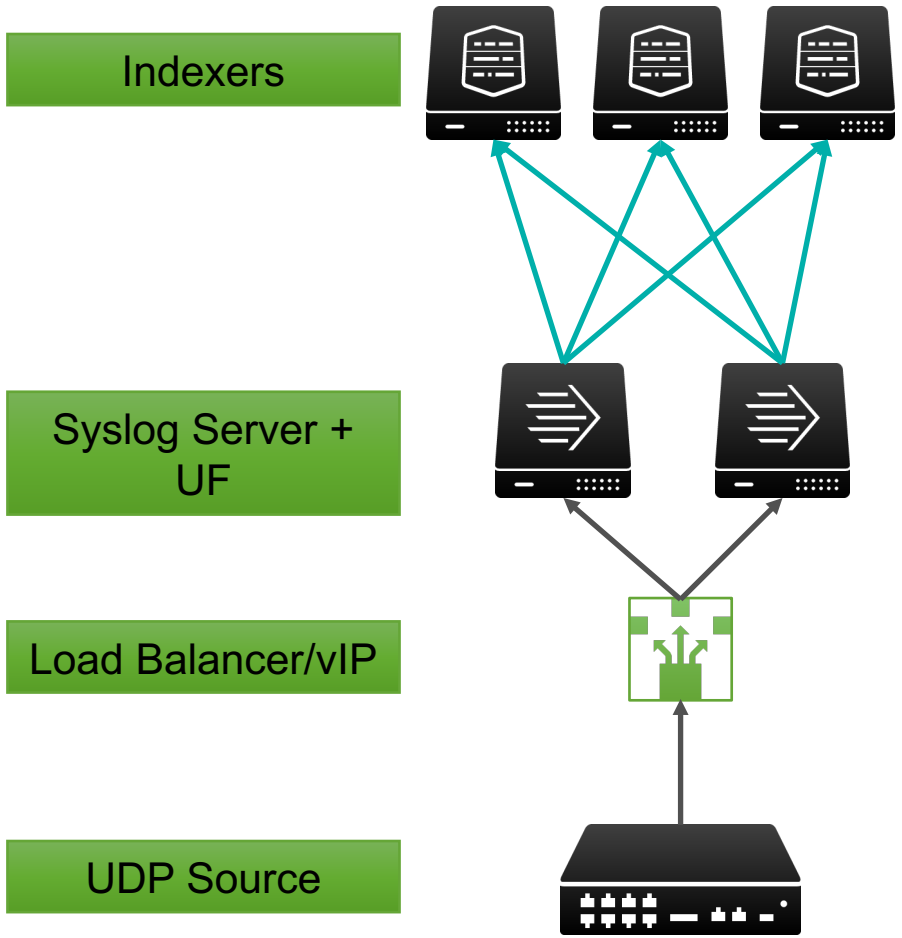
Simplified syslog collection

Goal: Minimize loss

- K.I.S.S. – Keep it Simple…Silly

Incorporate redundancy without making it overly complex

- Utilize a syslog server
  - Purpose built solution
  - Gives more flexibility
    - Host extraction, log rolling/retention
- Minimize # of network hops between source and syslog server



Direct TCP/UDP Data Collection

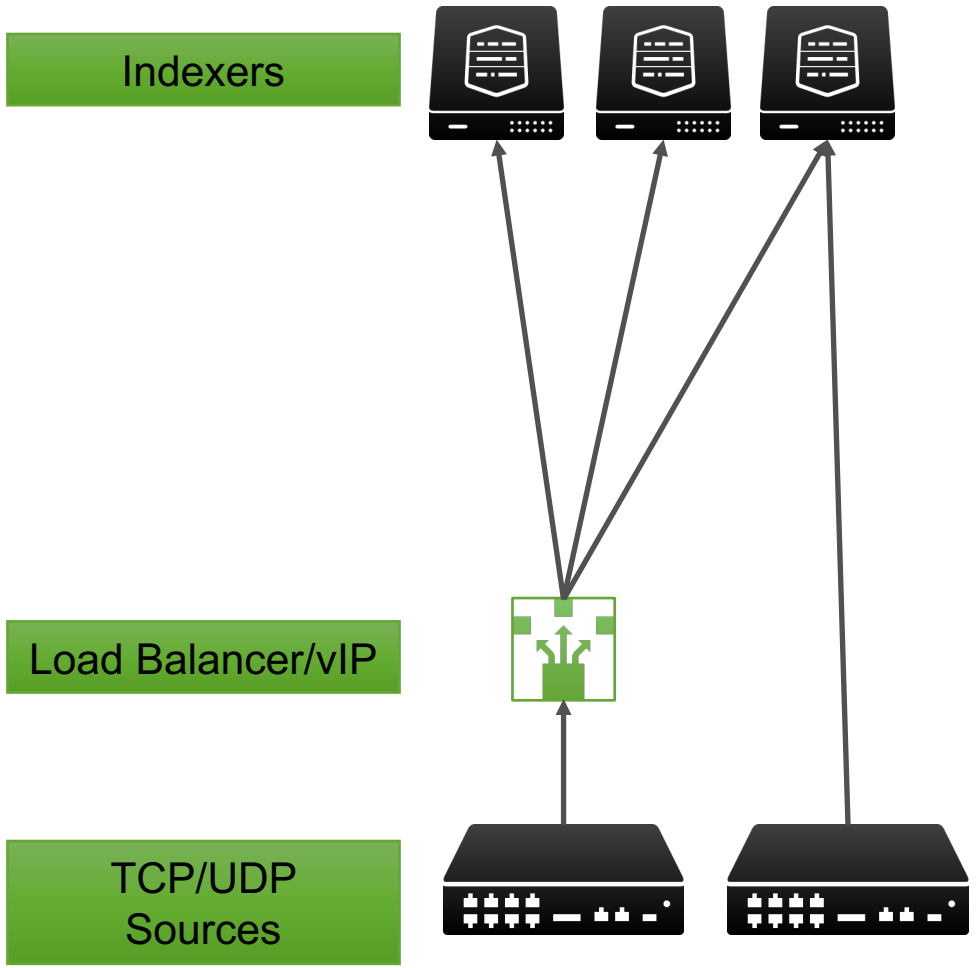
Worst Practice

Sending TCP/UDP straight to Indexers

- TCP/UDP stream sent to Indexers
  - Directly or via Load Balancer

Event distribution on Indexers is CRITICAL

- Distribute your search workload as much as possible across Indexers
- Load Balancers
  - Typically only DNS load balancing
    - Large streams can get stuck on an Indexer
  - Don't switch Indexers often enough



Best Practice

Use Splunk Auto Load Balancing

This looks familiar…

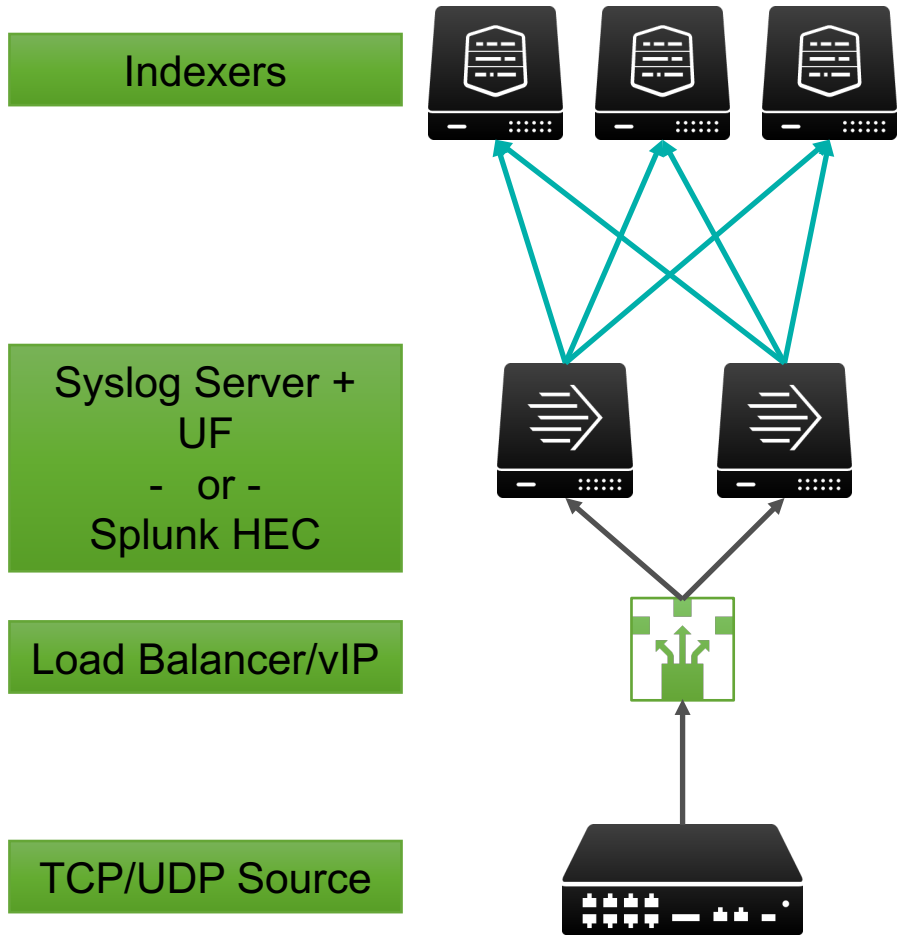
- It should, it's the same as the recommended UDP/Syslog configuration

Splunk AutoLB

- Handles distributing events across Indexers automatically
- [forceTimebasedAutoLB] or [event_breaker]
  - Can be used for large files or streams

▶ Utilize a syslog server

- For all the same reasons we discussed before



Forwarder Load Balancing

Load Balancing

A Primer…

What is it?

- Distributes events across Indexers

outputs.conf

```
autoLB = true
autoLBFrequency = 30
autoLBVolume = <bytes>
```

- Why is it important?
  - Distributed Processing
    - Distributes workload
    - Parallel processing
- When does it break?
  - Large files
  - Continuous data streams

How does it break?

- Forwarder keeps sending to the same Indexer until:

inputs.conf

```
[monitor://<path>]
time_before_close = 3
  * Secs to wait after EoF
[tcp://<remote server>:<port>]
rawTcpDoneTimeout = 10
```

  - Regardless of autoLB settings
- Why does that happen?
  - UF doesn't see event boundaries
  - We don't want to truncate events

Worst Practices

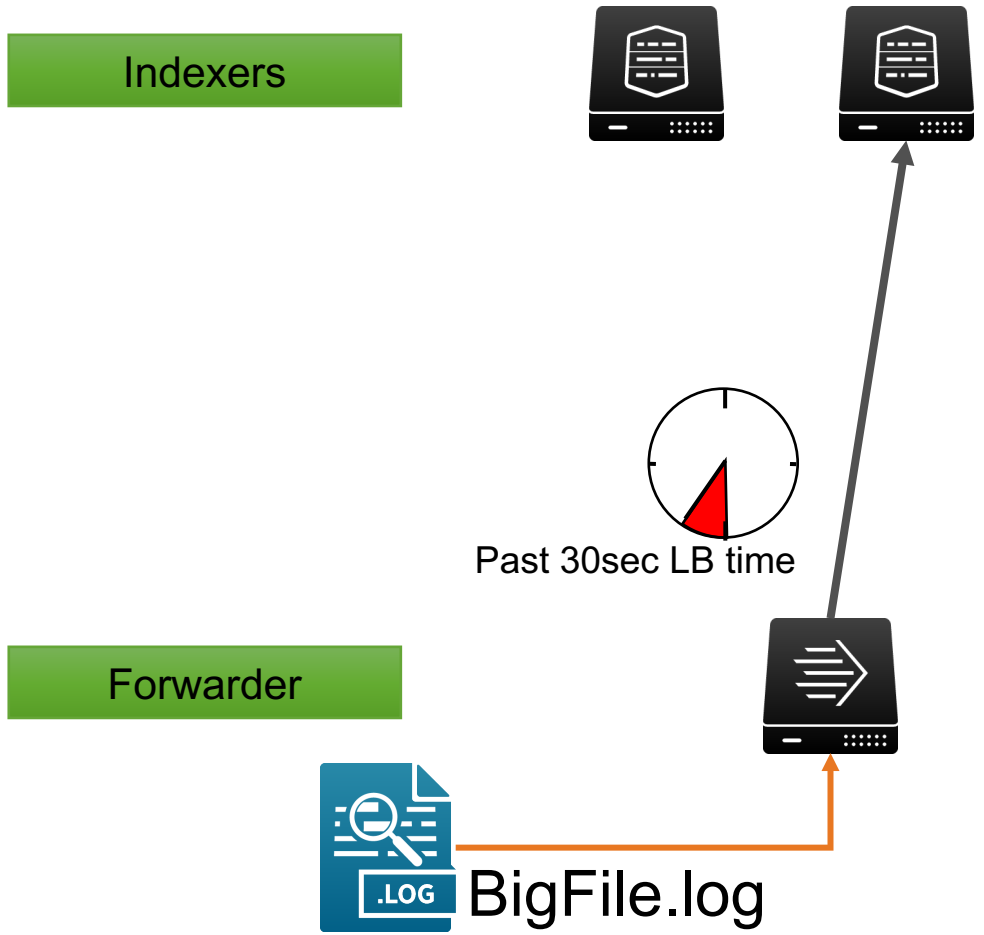
Sticky forwarders

Using the UF to monitor…

- Very large files
- Frequently updated files
- Continuous data streams

…Without modifying default autoLB behavior

- Forwarder can become “locked” onto an Indexer
  - We have settings that can help



Best Practices

Un-stick your forwarders

- If you're running 6.5+ UFs…
  - Use UF event breaking
    - Applied per sourcetype
      - Default behavior is followed if not configured

props.conf

```
[<sourcetype>]
EVENT_BREAKER_ENABLE = true
EVENT_BREAKER = <regex>
```

- If you're running a pre-6.5 UF...
  - Use [forceTimebasedAutoLB]

Events may be truncated if an individual event exceeds size limit

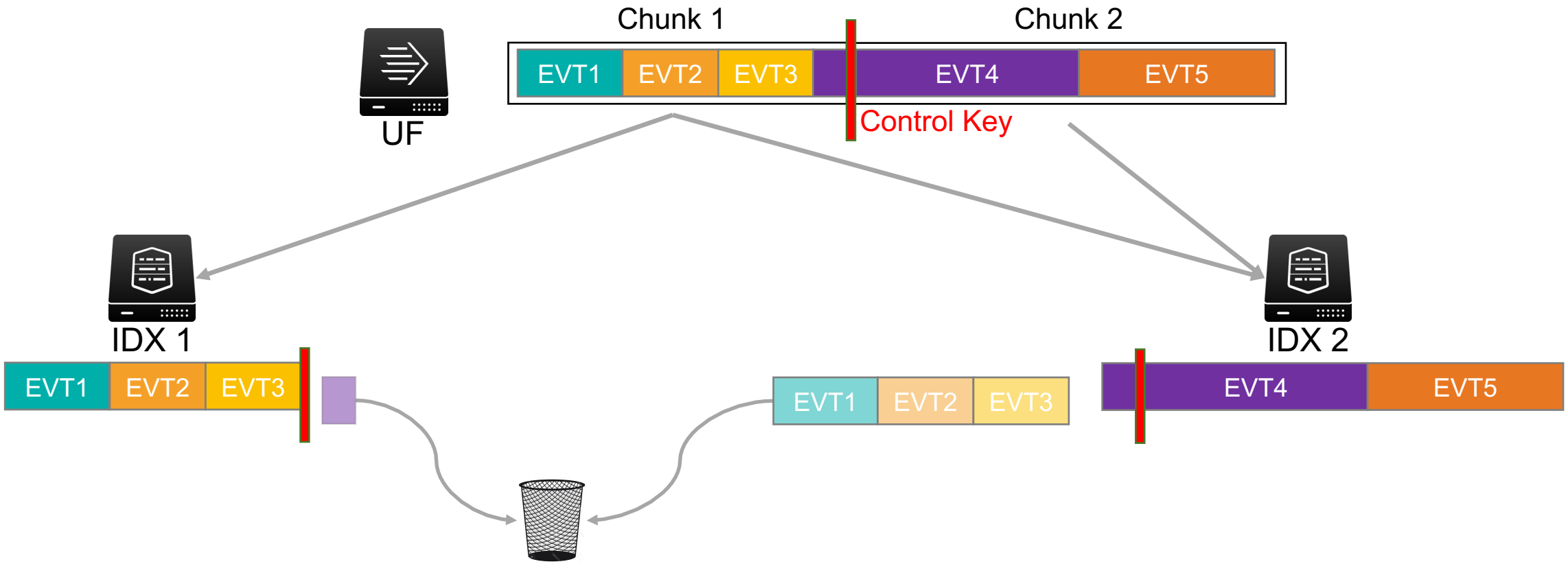
- Know the limits
  - File Inputs: 64KB
  - TCP/UDP Inputs: 8KB
  - Mod Inputs: 65.5KB (Linux Pipe Size)

outputs.conf

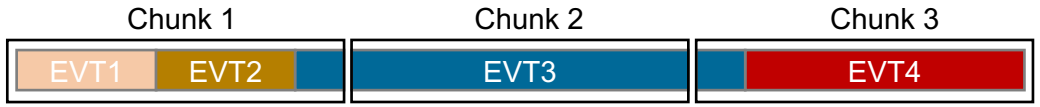
```
autoLB = true
autoLBFrequency = 30
forceTimeBasedautoLB =
true
```

forceTimebasedAutoLB

How does it work?



Problem Event:



splunk> .conf18

UF Event Breaking

A better way to get un-stuck

- Available in Splunk 6.5+
  - Only available on the Universal Forwarder (UF)

What does it do?

- Provides lightweight event breaking on the UF
  - AutoLB processor now sees event boundaries
    - Prevents locking onto an Indexer
    - [forceTimeBasedautoLB] not needed for trained Sourcetypes

How does it work?

- Props.conf on UF
- Event breaking happens for specified Sourcetypes
- Sourcetypes without an event breaker are not processed
  - Regular AutoLB rules apply

props.conf

```
[<sourcetype>]
EVENT_BREAKER_ENABLE = true
EVENT_BREAKER = <regex>
```

Intermediate Forwarders

Gone Wrong

splunk> .conf18

Intermediate forwarder

noun

:

A Splunk Forwarder, either Heavy or Universal, that sits between a Forwarder and an Indexer.

Worst Practice

Using Heavy Forwarders vs. Universal Forwarders

Only use Heavy Forwarders (HWF) if there is a specific need

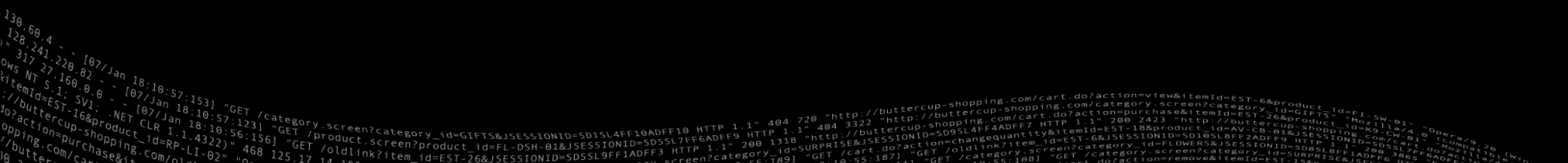
- You need Python
- Required by an App/Feature
  - HEC, DBX, Checkpoint, etc…
- Advanced Routing/Transformation
  - Routing individual events
  - Masking/SED
- Need a UI on the Forwarder

What's Wrong with my HWFs?

- Additional administrative burden
  - More conf files needed on HWFs
  - Increases difficulty in troubleshooting
- Cooked Data vs. Seared

Cooked: ~20% larger over the network

- UFs can usually do the same thing
  - Intermediate Forwarding
  - Routing based on data stream



splunk> .conf18

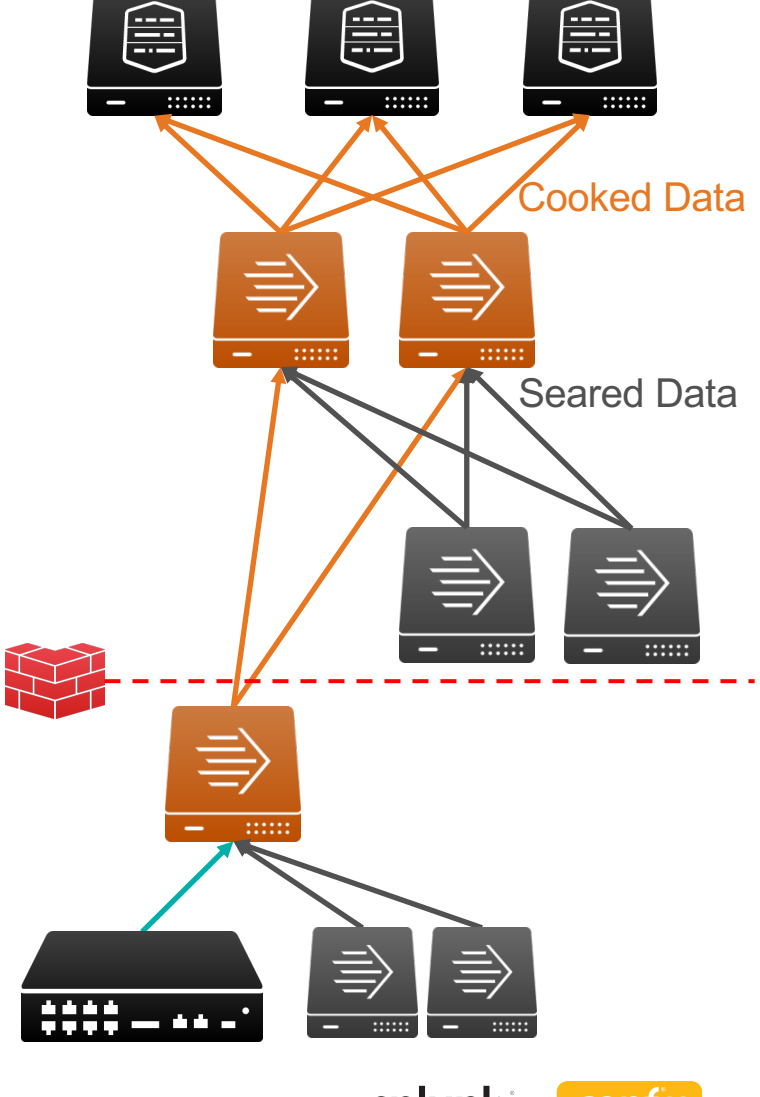
Worst Practice

Creating data funnels

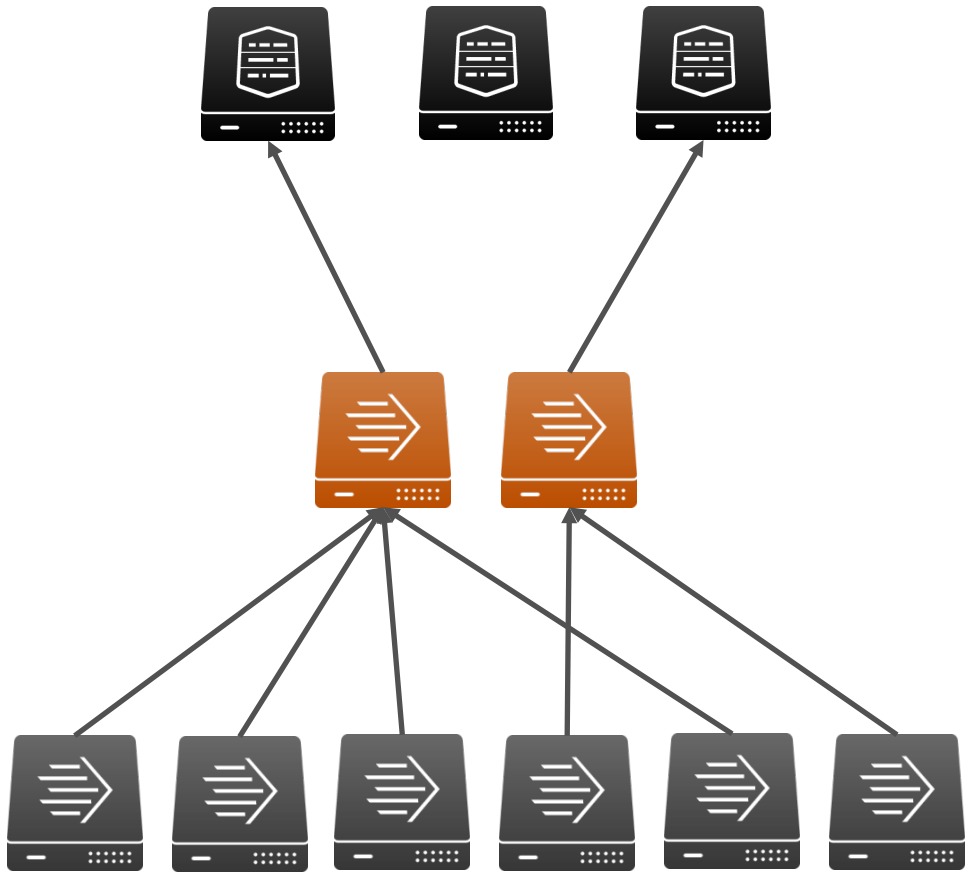
- Intermediate Forwarders
  - No data is being sent directly to indexers
  - HWFs are used when UFs will do

Avoid data funnels

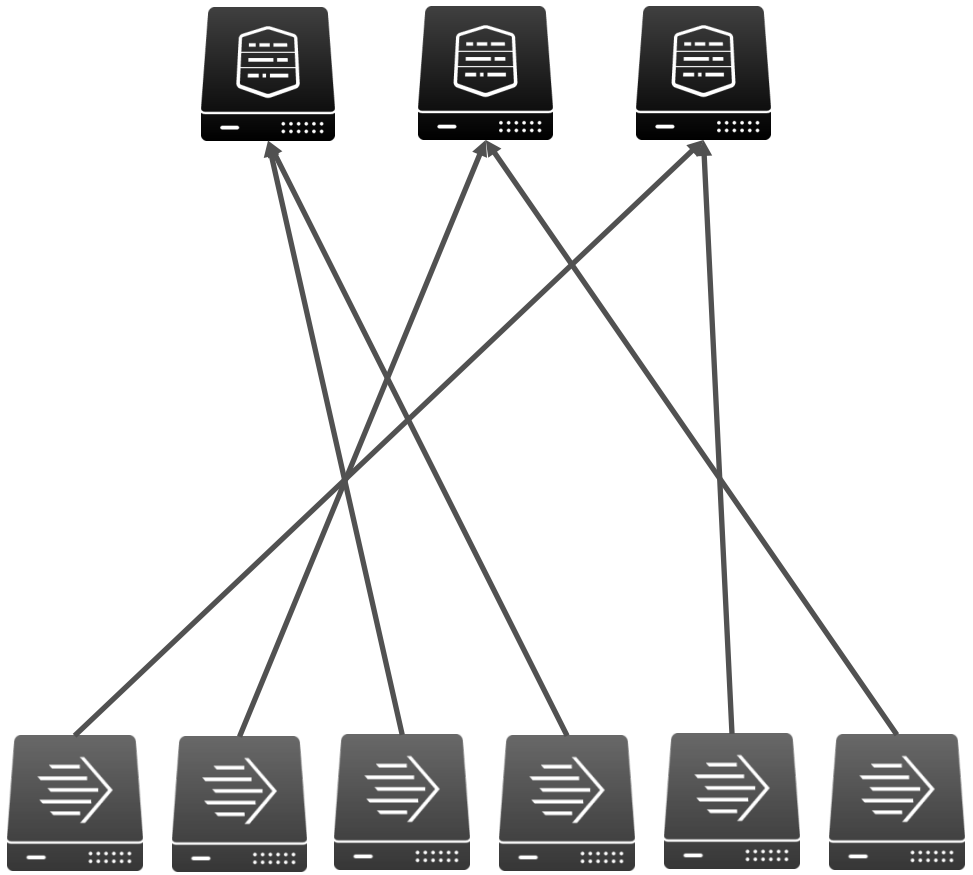
- Forwarders sending data to a handful of intermediate forwarders
- Causes indexer starvation
  - Indexers aren't receiving events for periods of time
  - Results in data imbalance and poor search performance



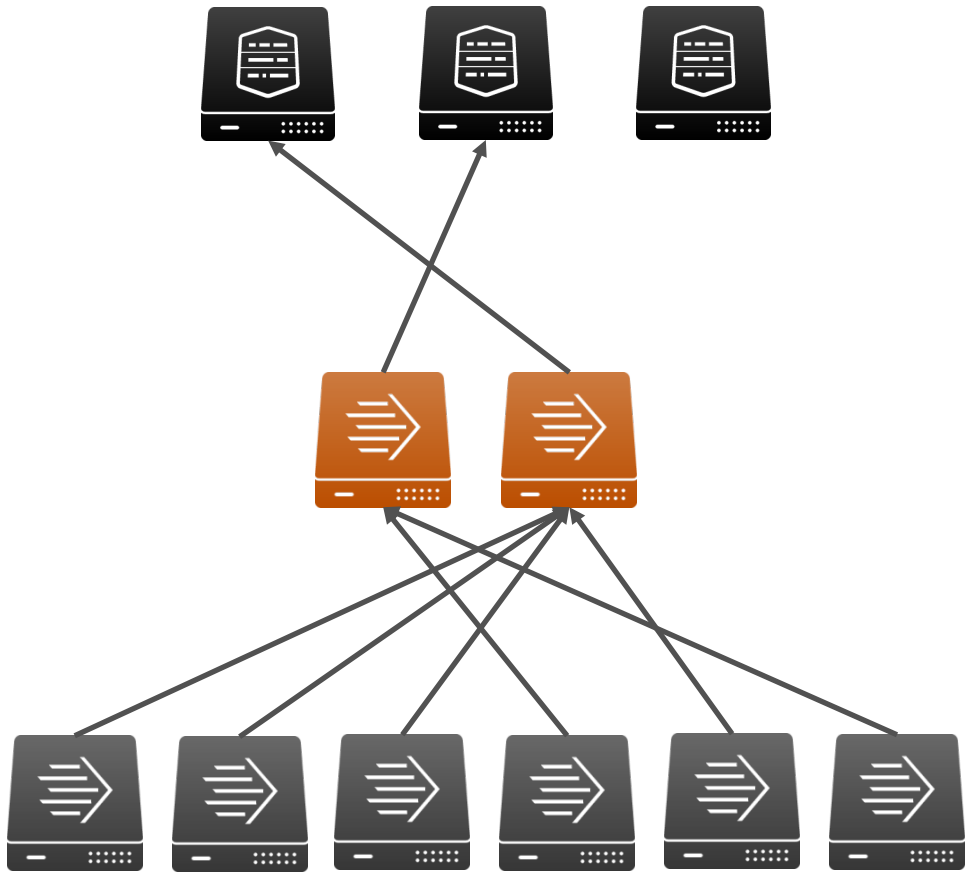
The Funnel Effect



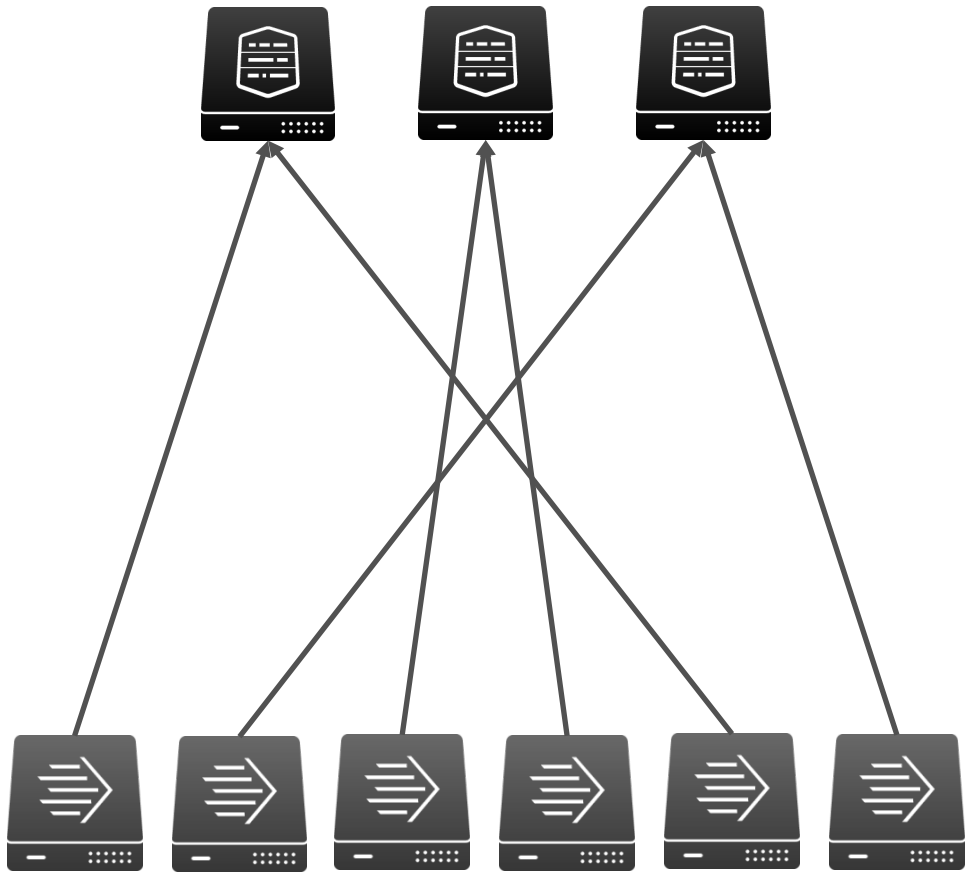
-vs-



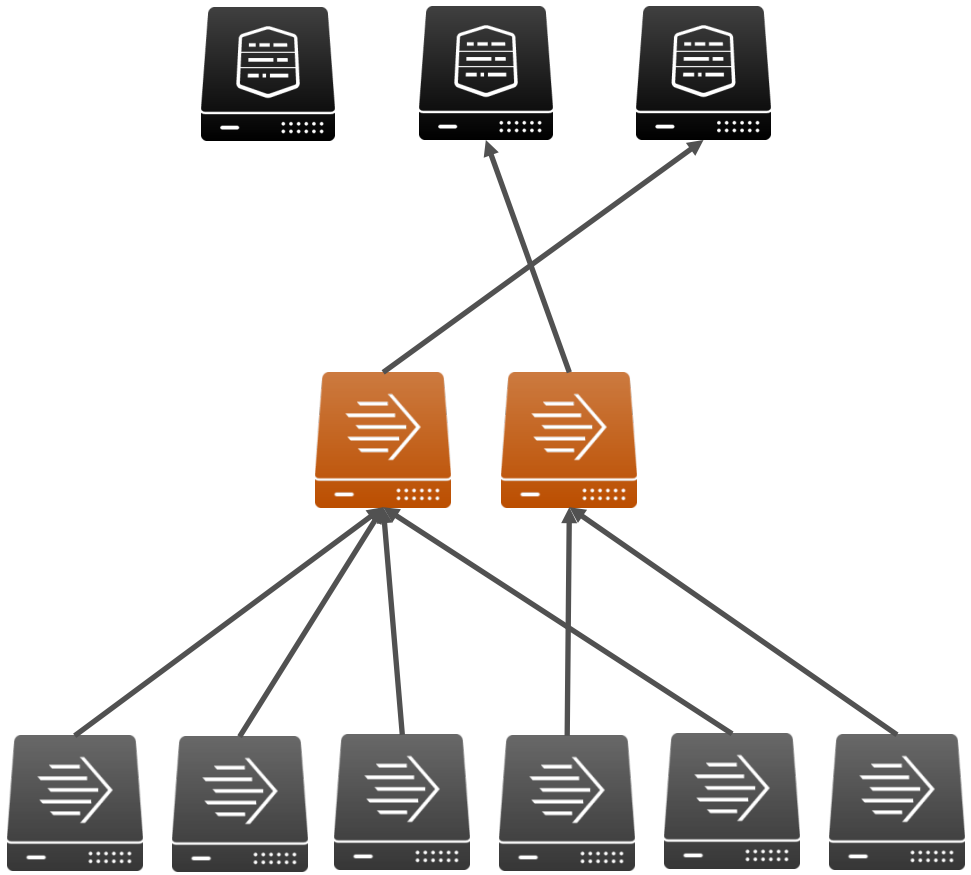
The Funnel Effect



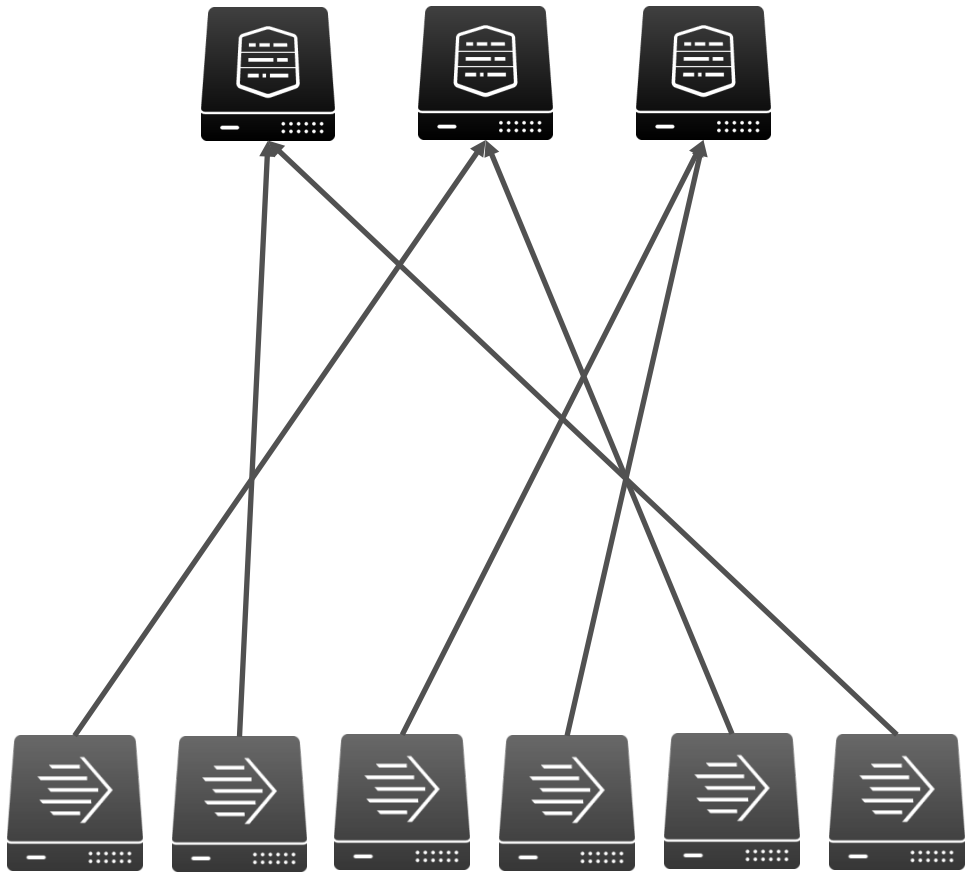
-vs-



The Funnel Effect



-vs-



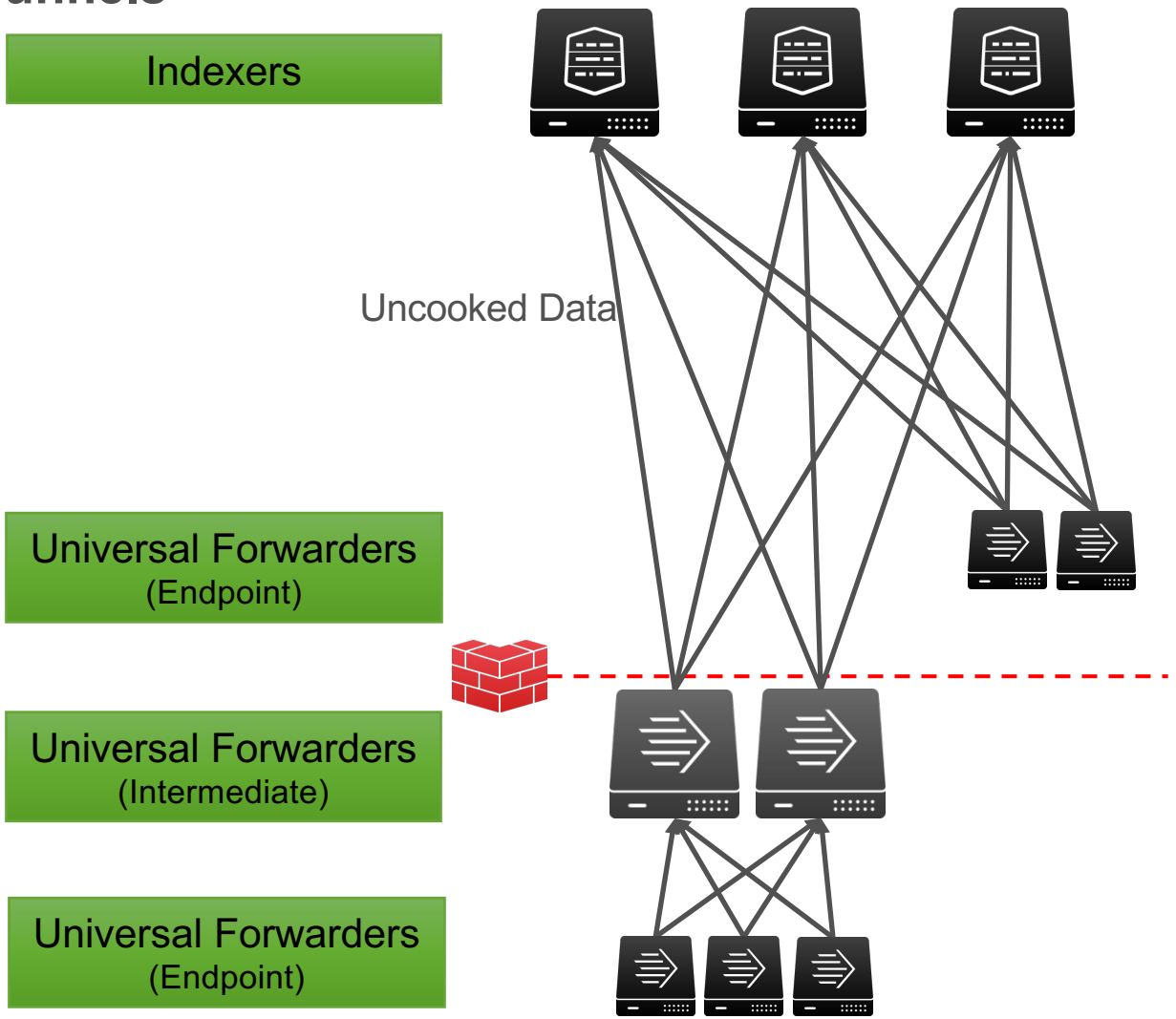
Best Practice

Reduce funnels

- Intermediate Forwarders
  - Limit their use
    - Most helpful when crossing network boundaries

Utilize forwarder parallelization

  - Avoid the “funnel effect”
- UFs → Indexers
  - Aim for 2:1 ratio
    - Parallelization or Instances
  - More UFs avoids Indexer starvation
- UF vs. HWF
  - Seared data vs. cooked
  - Less management required for conf files



Data Onboarding

Get it tight, get it right

splunk> .conf18

Sourcetype Recognition

Who is your daddy and what does he do?

Avoid automatic sourcetype recognition where possible

- Specify the sourcetype in inputs.conf

Inputs.conf

```
[monitor:///var/log]
sourcetype = mylog
```

- Don't let Splunk guess for you
  - Requires additional processing due to RegEx matching
    - "too small" sourcetypes may get created

Timestamps

What did this happen?

Don’t let Splunk guess

- Are you sensing a theme?

- Side Effects
  - Incorrect Timestamp/TZ extraction
  - Missing/Missed Events
  - Bucket Explosion

- These parameters are your friends

Props.conf

```
[mySourcetype]
TIME_PREFIX =
TIME_FORMAT =
MAX_TIMESTAMP_LOOKAHEAD =
```

What comes before the timestamp?

What does the timestamp look like?

How far into the event should Splunk look to find the timestamp?

Event Parsing

Break it down

- Line Breaking

Avoid Line Merging

- SHOULD_LINEMERGE = true
- BREAK_ONLY_BEFORE_DATE, BREAK_ONLY_BEFORE, MUST_BREAK_AFTER, MUST_NOT_BREAK_AFTER, etc…

LINE_BREAKER is much more efficient

Props.conf

```
[mySourcetype]
SHOULD_LINEMERGE = false
LINE_BREAKER = <regex>
```

- Uses RegEx to determine when the raw text should be broken into individual events

Indexed Extractions and Accelerations

Speeding things up

splunk> .conf18

What is an Indexed Extraction?

Splunk stores the Key-Value pair inside the TSIDX

- Created at index-time
- Lose Schema-on-the-fly flexibility
- Can improve search performance
  - Can also negatively impact performance

▶ Example

- KV Pair: Trooper=TK421
- Stored in TSIDX as: Trooper::TK421

Worst Practice

Indexed Extractions Gone Wild

- Indexing all ”important” fields
  - Unique KV pairs are stored in the TSIDX
  - KV Pairs with high cardinality increase the size of the TSIDX
    - Numerical values, especially those with high precision

Large TSIDX = slow searches

- Statistical queries vs. filtering events
  - Indexed extractions are helpful when filtering raw events
  - Accelerated Data Models are a better choice for statistical queries
    - A subset of fields/events are accelerated
    - Accelerations are stored in a different file from the main TSIDX

Best Practice

When should I use Indexed Extractions?

- The format is fixed or unlikely to change
  - You loose schema on the fly with indexed extractions
- Values appear outside of the key more often than not

```
index=myIndex Category=X1
```

```
2016-11-12 1:02:01 PM INFO Category=X1 ip=192.168.1.65 access=granted message=Access granted to X1 system
2016-11-15 12:54:12 AM INFO Category=F2 ip=10.0.0.66 message=passing to X1 for validation
```

- Almost always filter using a specific key (field)
  - Categorical values (low cardinality)
  - Don't index KV pairs with high cardinality
- Frequently searching a large event set for rare data
  - KV pair that appears in a very small % of events
  - foo!=bar or NOT foo=bar and the field foo nearly always has the value of bar

Restricted Search Terms

Lock it down

splunk> .conf18

What are Restricted Search Terms?

Nothing to see here…

- Filtering conditions
  - Added to every search for members of the role as AND conditions
    - All of their searches MUST meet the criteria you specify
  - Terms from multiple roles are OR'd together
- Where do I find this?
    - Access Controls > Roles > [Role Name] > Restrict search terms
- Not secure unless filtering against Indexed Extractions
  - Users can override the filters using custom Knowledge Objects
  - Indexed Extractions use a special syntax
    - key::value
      Ex: sourcetype::bluecoat

Worst Practice

All the hosts!

- Inserting 100s or 1,000s of filtering conditions
  - Hosts, App IDs
- "Just-In-Time" Restricted Terms
  - Built dynamically on the fly
    - Custom search commands/Macros
  - Can be complex/delay search setup

```
host=Gandalf OR host=frodo OR host=Samwise OR
host=Aragorn OR host=Peregrin OR host=Legolas OR
host=Gimli OR host=Boromir OR host=Sauron OR host=Gollum
OR host=Bilbo OR host=Elrond OR host=Treebeard OR
host=Arwen OR host=Galadriel OR host=Isildur
```

Best Practice

When should I filter?

- Filter based on categorical fields that are Indexed
  - Remember…low cardinality
  - Indexed extractions are secure, Search-time extractions are not
    - Use key::value format

Less is more

- Reduce the # of KV-Pairs you're inserting into the TSIDX
  - Larger TSIDX = slower searches
- Limit the # of filters you're inserting via Restricted Search Terms
  - Find ways to reduce the # of roles a user belongs to
  - Don't create specific filters for data that doesn't need to be secured
    - Use an "All" or "Unsecured" category

Multi-Site Search Head Clusters

splunk> .conf18

Search Head Clustering

A Primer…

- SHC members elect a captain from their membership
- Minimum of 3 nodes required
  - Captain election vs. static assignment
- Odd # of SHC members is preferred
- Captain Manages
  - Knowledge object replication
  - Replication of scheduled search artifacts
  - Job scheduling
  - Bundle replication

Multi-Site SHC does not exist

- What?!
- SHC is not site-aware
  - You're creating a stretched-SHC

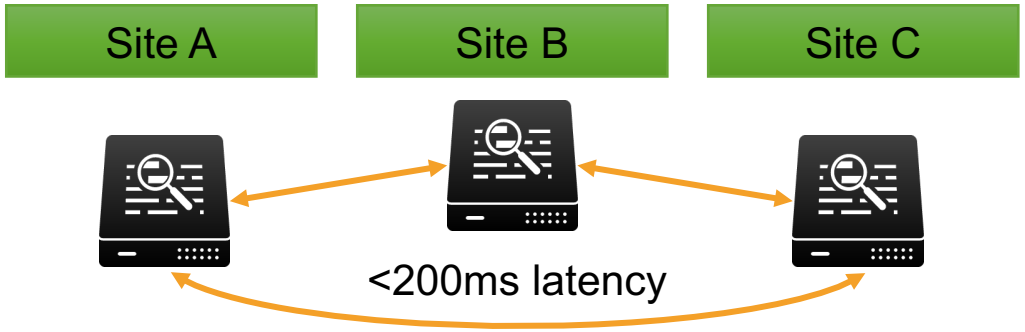
Worst Practice

A ship without a captain

- Captain Election not possible with site or link failure
  - No site has node majority
    - Original SHC size: 4 Nodes
    - Node Majority: 3 Nodes
  - Odd # of SHC members is preferred
- WAN Latency is too high
  - We've tested up to 200ms

Best Practices

Designing a better Search Head Cluster

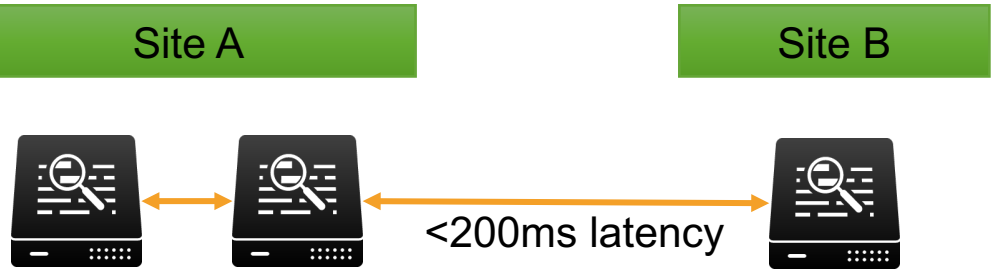


Two Sites: Semi-Automatic Recovery

- Site A has node majority
  - Captain can be elected in Site A if Site B fails
  - Captain must be statically assigned in Site B if Site A fails
- WAN latency is <200ms

server.conf

```
[shclustering]
adhoc_searchhead = true
preferred_captain = false
no_artifact_replication = true
```



Three Sites: Fully Automatic Recovery

- Node majority can be maintained with a single site failure
  - Keep Indexers in 2 sites
    - Simplifies index replication
  - Limit workload on SH in 3rd site

Index Management

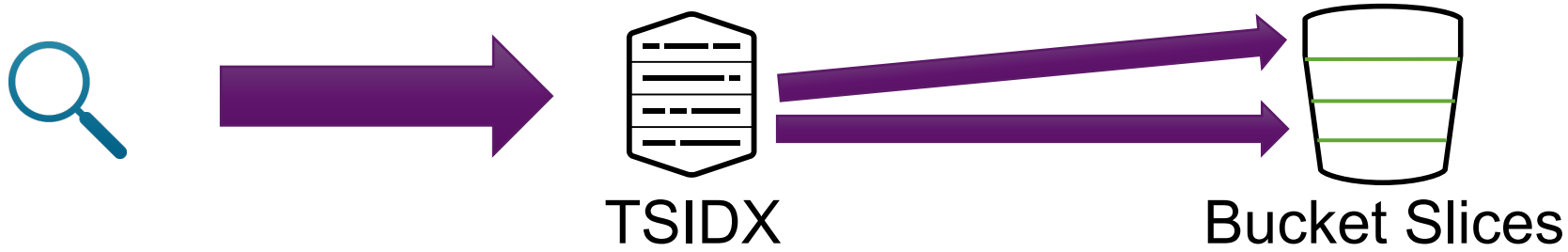
Where should you put your data?

splunk> .conf18

Search Goals

How do I make my searches fast?

- Find what we're looking for quickly in the Index (TSIDX)
  - Lower cardinality in the dataset = fewer terms in the lexicon to search through
- Decompress as few bucket slices as possible to fulfill the search
  - More matching events in each slice = fewer slices we need to decompress
- Match as many events as possible
  - Unique search terms = less filtering after schema is applied
  - Scan Count vs. Event Count

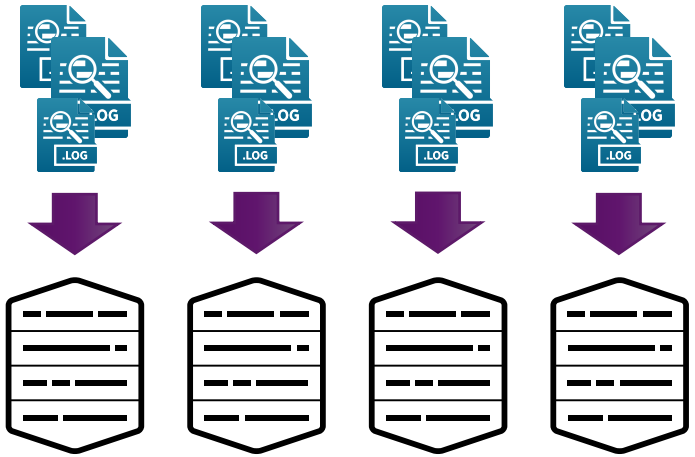


Worst Practice

When should I create Indexes?

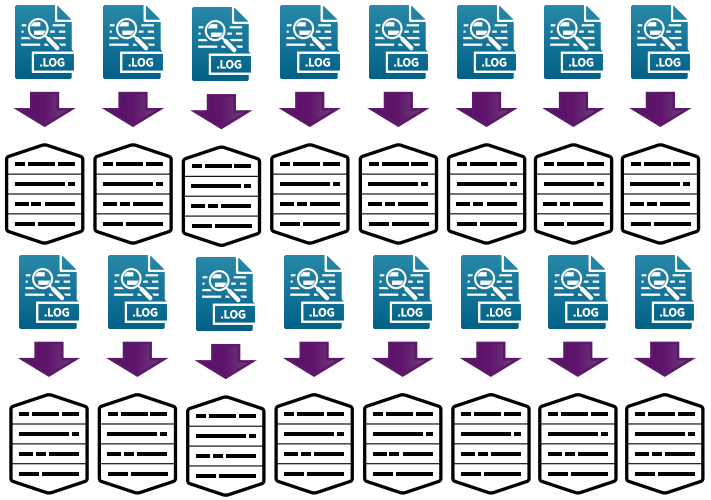
Goldilocks for Your Splunk Deployment

Mix of data in a handful of Indexes



This deployment has too few Indexes…

Dedicated Indexes for Sourcetypes



This deployment has too many Indexes…

Too Few Indexes

…and the problems it creates

- What do we write to the Index (TSIDX)?
  - Unique terms
  - Unique KV Pairs (Indexed Extractions)
- Higher data mix can mean higher cardinality
  - More unique terms = Larger TSIDX
    - Larger TSIDX files take longer to search
- More raw data to deal with
  - Potentially uncompressing more bucket slices
  - Searches can become less dense
    - Lots of raw data gets filtered out after we apply schema

Too Many Indexes

If small indexes are faster, why not just create a lot of them?

- Complex to manage
- Index Clustering has limitations
  - Cluster Master can only manage so many buckets
    - Total buckets = original and replicas

Version	Unique Buckets	Total Buckets
6.3 & 6.4	1M	3M
6.5	1.5M	4.5M
6.6, 7.0, 7.1	5M	15M
7.2	9.5M	28.5M

- What if I'm not using Index Clustering?
  - Create as many indexes as you want!

Best Practice

When to Create Indexes

- Retention
  - Data retention is controlled per index
- Security Requirements
  - Indexes are the best and easiest way to secure data in Splunk
- Keep “like” data together in the same Index
  - Service-level Indexes
    - Sourcetypes that are commonly searched together
    - Match more events per bucket slice
  - Sourcetype-Level Indexes
    - Data that has the same format
    - Lower cardinality = smaller TSIDX

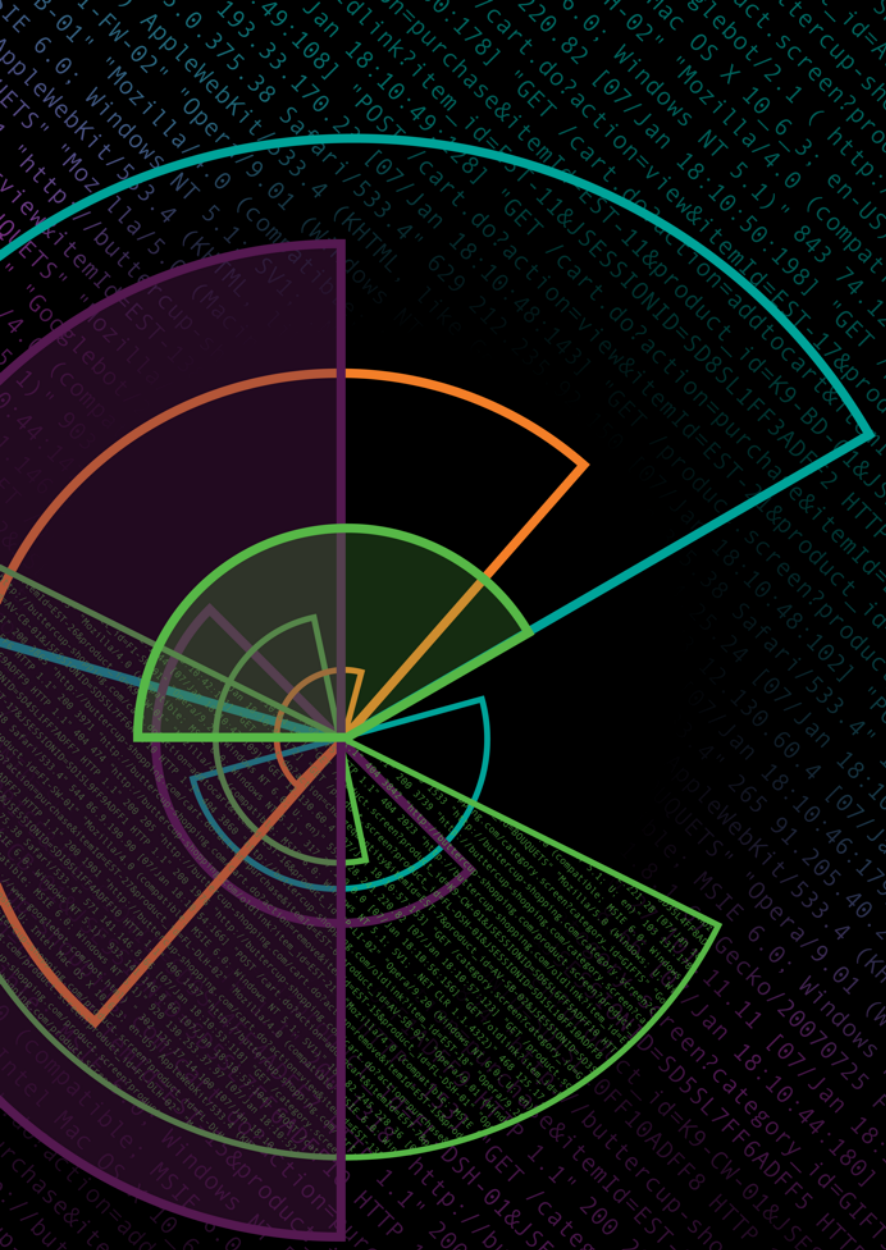
What if I need thousands of Indexes to secure my data?

- Don't. ☺
  - More indexes = more buckets = bad for your Index Cluster

Look for ways to reduce the complexity of your security model

- Organize by Service
  - Collection of apps/infrastructure
- Organize by groups
  - Org, Team, Cluster, Functional Group

- Consider Indexed Extractions & Restricted Search Terms



Index Replication

Give me 10 of everything!

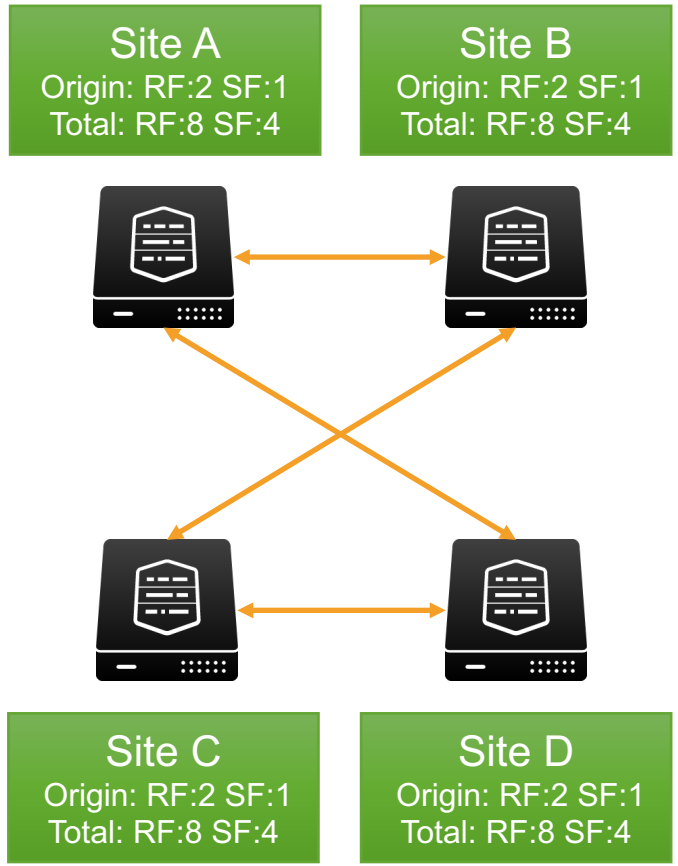
Worst Practice

Replicate all the things!

- Lots of Replicas & Sites
  - 8 replicas in this example
  - 4 sites

Index Replication is Synchronous

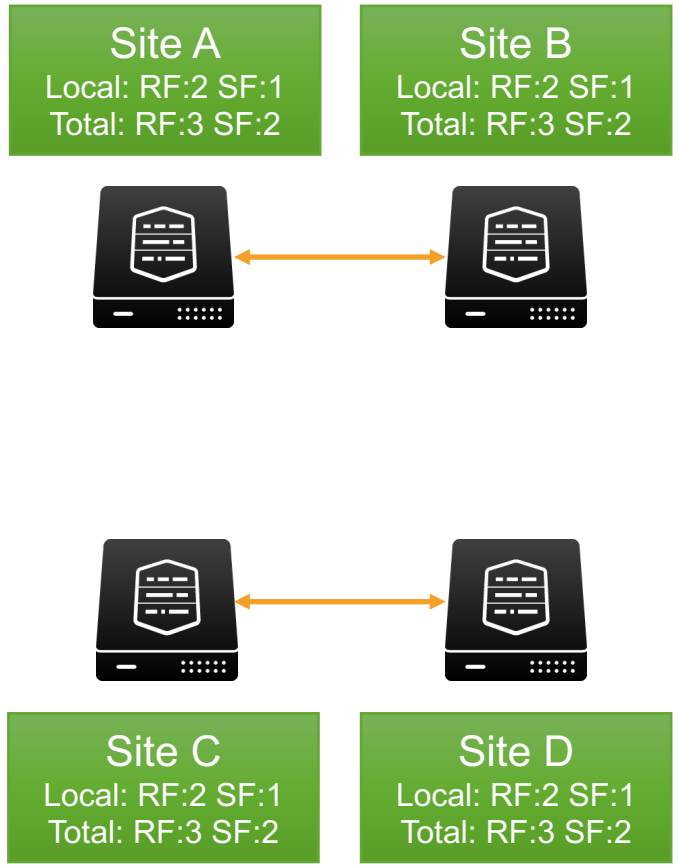
- Bucket slices are streamed to targets
  - Excess replication can slow down the Indexing pipeline
- Replication failures cause buckets to roll from hot to warm prematurely
  - Creates lots of small buckets



Best Practice

K.I.S.S.

- Reduce the number of replicas
  - 2 local copies and 1 remote is common
- Reduce the number of remote sites
  - Disk space is easier to manage with 2 sites
- WAN Latency
  - Recommended: <75ms
    - Max: 100ms
- Keep an eye on replication errors
  - Avoid small buckets



High Availability

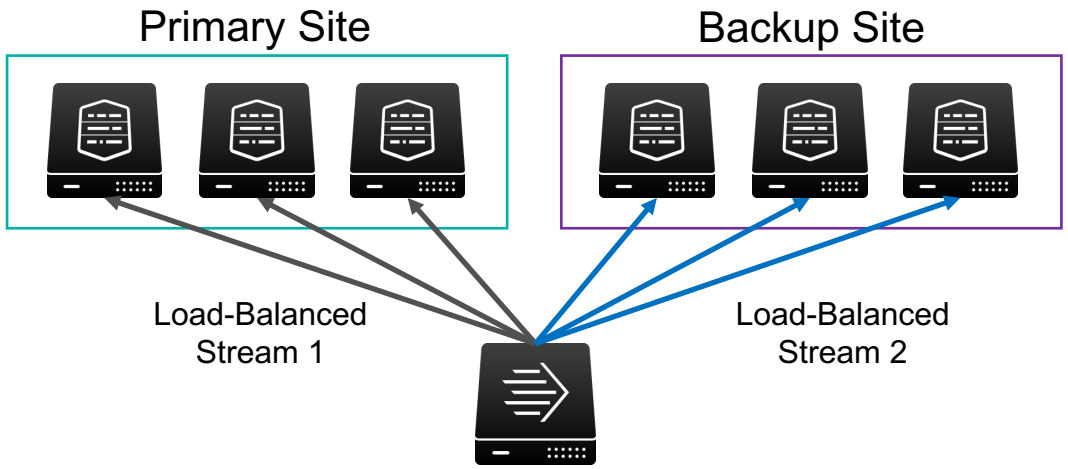
MacGyver Style

splunk> .conf18

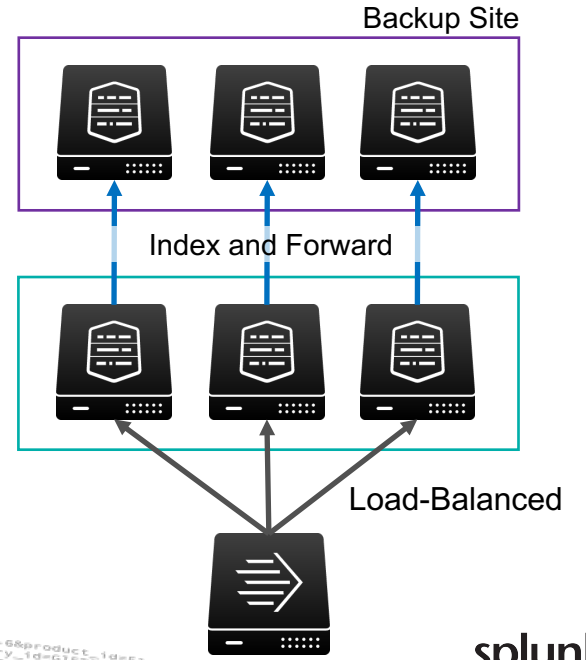
Some Worst Practices

Quick 'n Dirty HA

- Cloned Data Streams
  - Data is sent to each site
  - Inconsistency is likely
    - If a site is down, it will miss data
  - Difficult to re-sync sites



- Index and Forward
  - RAID1-style HA
    - Failover to backup Indexer
  - Forwarders must be redirected manually
  - Complex recovery



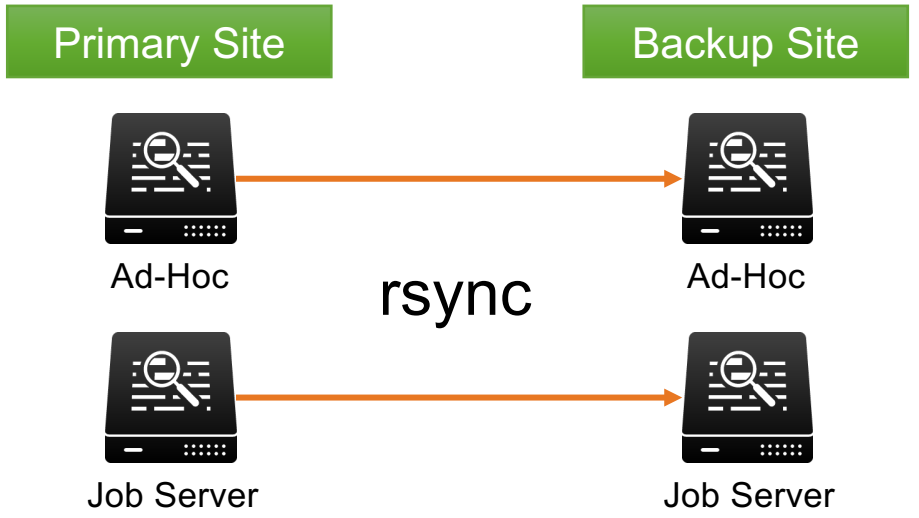
splunk> .conf18

Another Worst Practice

Job servers are so 2006

Rsync & Dedicated Job Servers

- Wasted ”standby” capacity in DR
- Inefficient use of resources between Ad-Hoc and Job Servers
- Conflict management is tricky if running active-active
- Search artifacts are not proxied or replicated
  - Jobs must be re-run at backup site

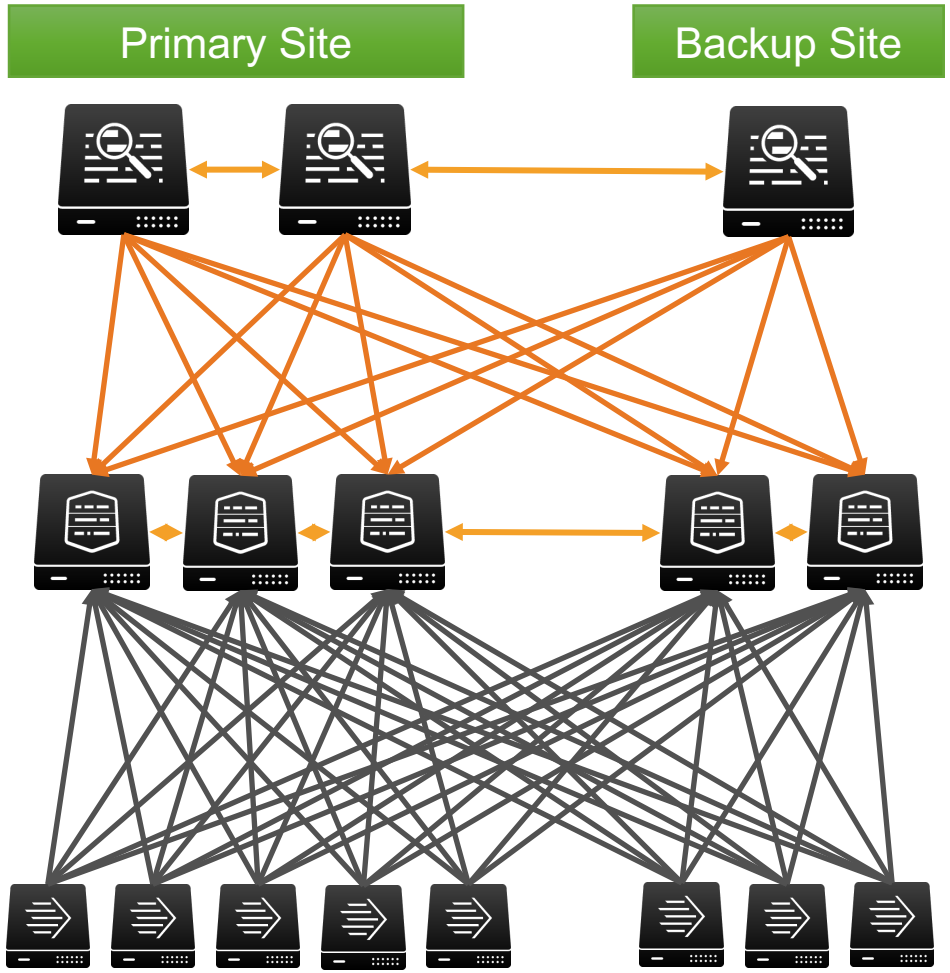


splunk> .conf18

Some Best Practices

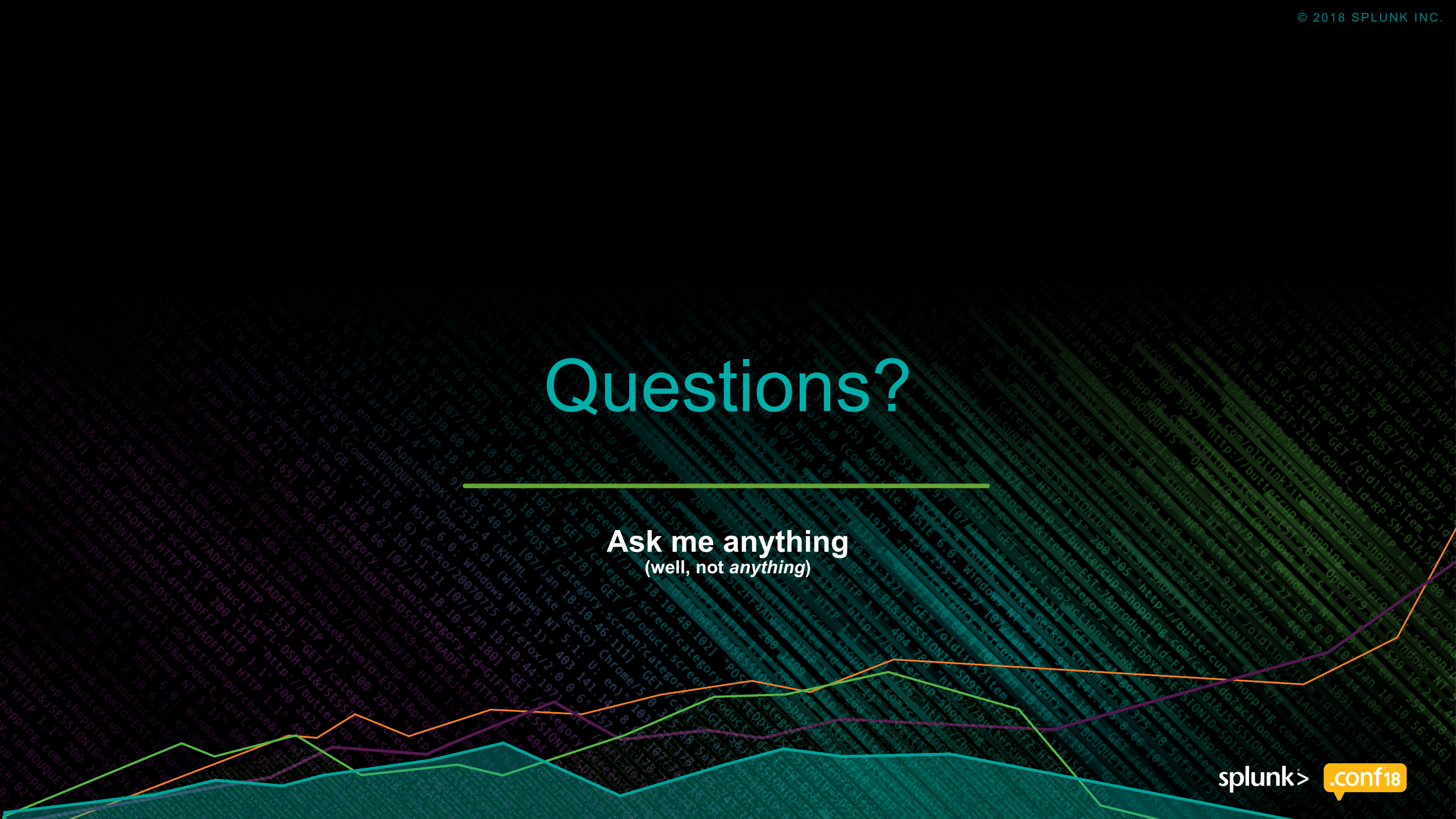
Splunk HA

- Index Clustering
  - Indexes are replicated
  - Failure recovery is automatic
- Search Head Clustering
  - Relevant Knowledge Objects are replicated
  - Search artifacts are either proxied or replicated
  - Managed Job scheduling
    - No dedicated job servers
    - Failure recovery is automatic
- Forwarder Load Balancing
  - Data is spread across all sites
  - Replicas are managed by IDX Clustering
  - DNS can be used to ”failover” forwarders between sites or sets of Indexers



Questions?

Ask me anything
(well, not *anything*)



splunk> .conf18

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