Worst Practices…And How To Fix Them

Jeff Champagne

Staff Architect, Splunk

Disclaimer

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 the 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.

Who's This Dude?

Jeff Champagne

jchampagne@splunk.com

Staff Architect

- Started with Splunk in the fall of 2014
- 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

Am I In The Right Place?

Yes, if you…

- Are a Splunk Admin or Knowledge Manager
- Understand what a Distributed Splunk Architecture looks like
- Are familiar with the Splunk roles
  - Search Heads, Indexers, Forwarders
- Know what indexes are…and ideally buckets too
- Familiar with Index Clustering and Search Head Clustering

Agenda

- Data Collection
- Data Management
- Data Resiliency

DISCLAIMER

The stories you are about to hear are true; only the names have been changed to protect the innocent.

Lossless Syslog/UDP

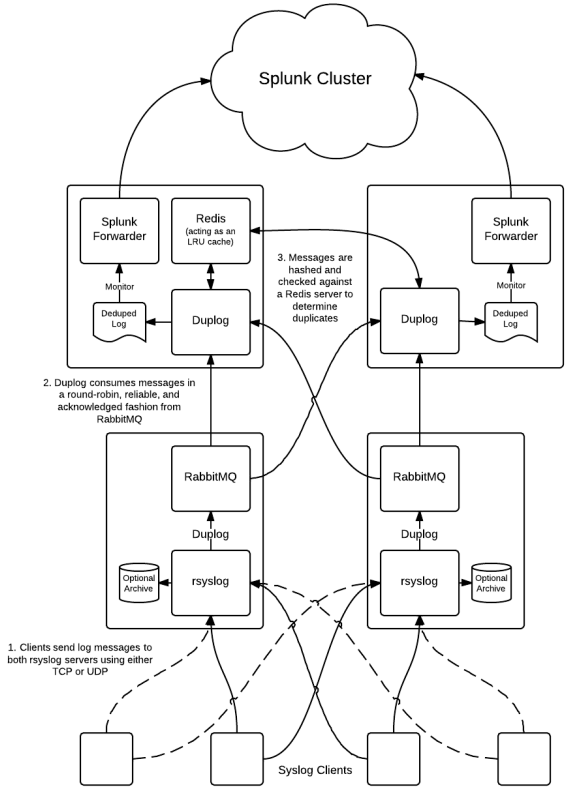
The Myth…

- 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

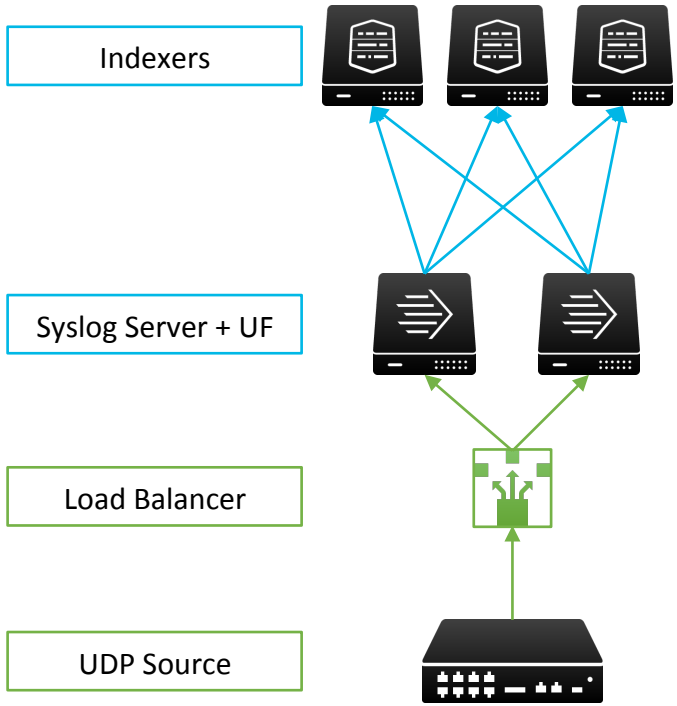
Over-Engineering

- 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

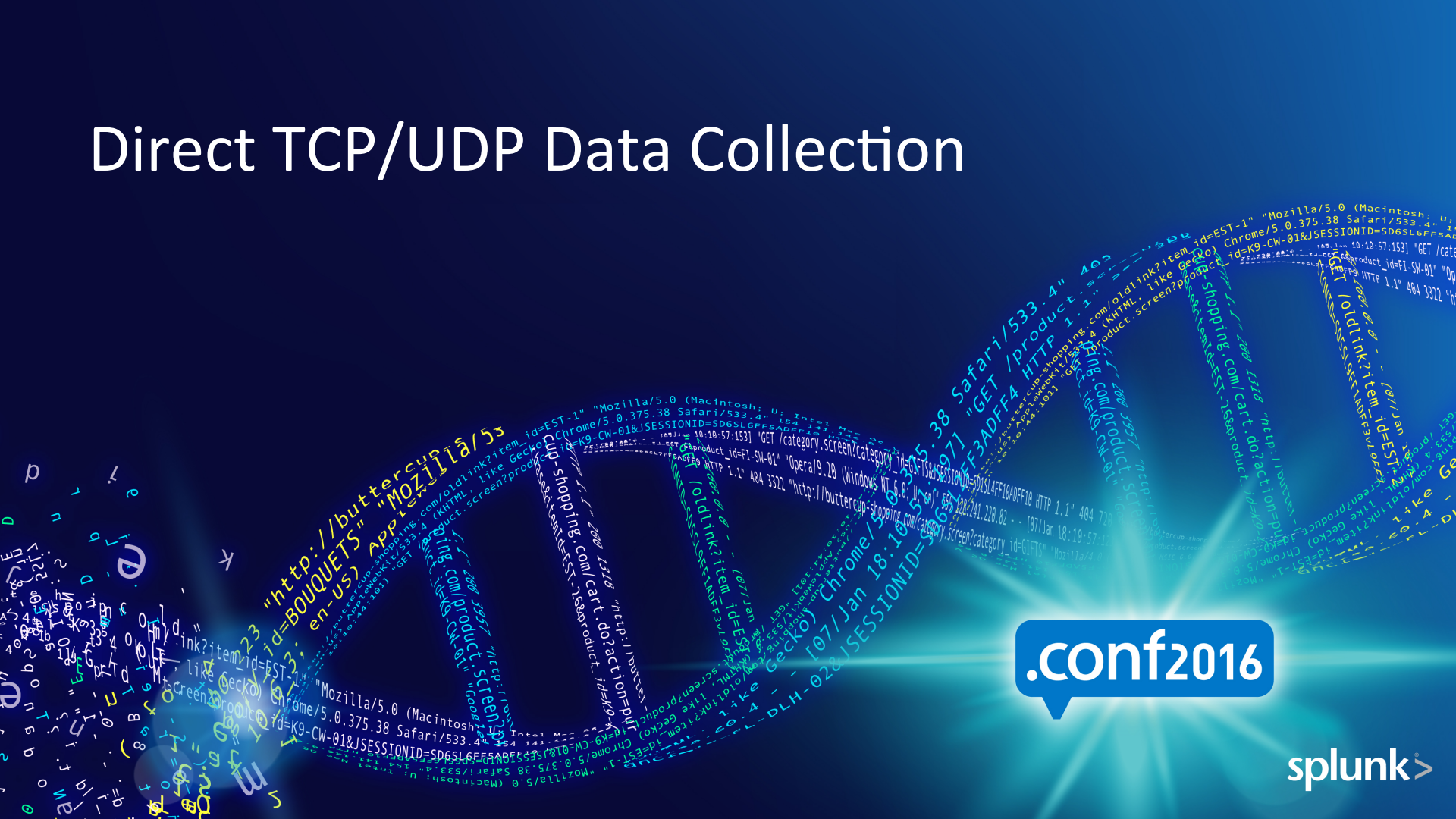


Best Practice

- 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

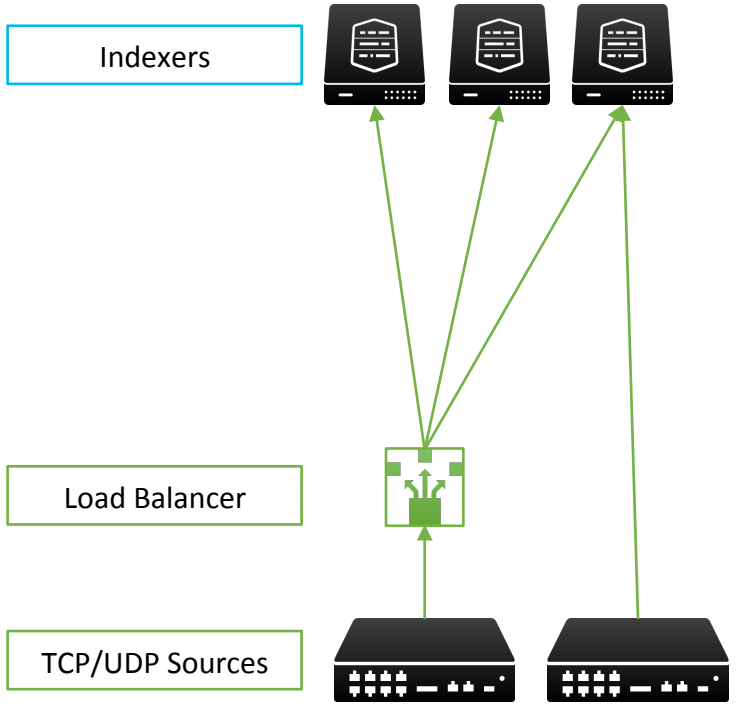


Direct TCP/UDP Data Collection



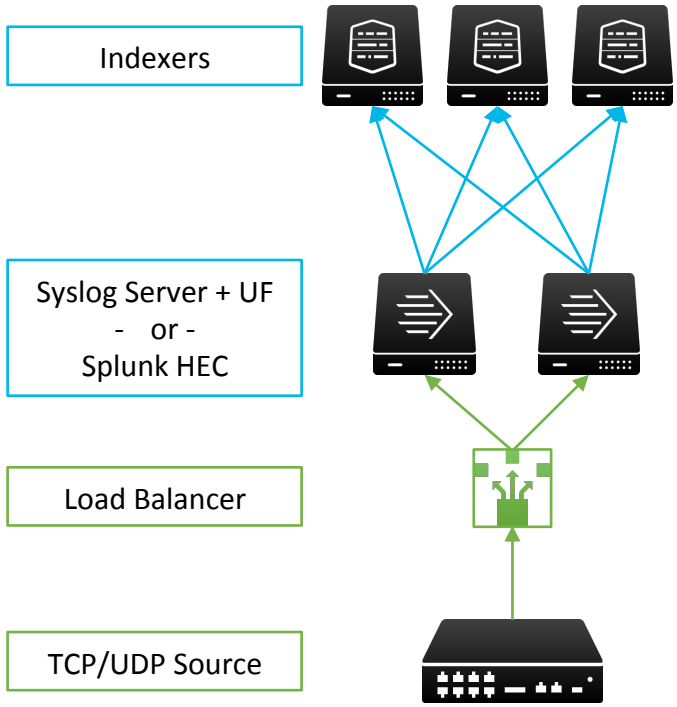
Worst Practice

- 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

- This looks familiar…
  - It should, it's the same as the recommended UDP/Syslog configuration
- Splunk AutoLB
  - Handles distributing events across Indexers automatically
  - forceTimebasedAutoLB 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
  - Time switching

outputs.conf

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

- 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:

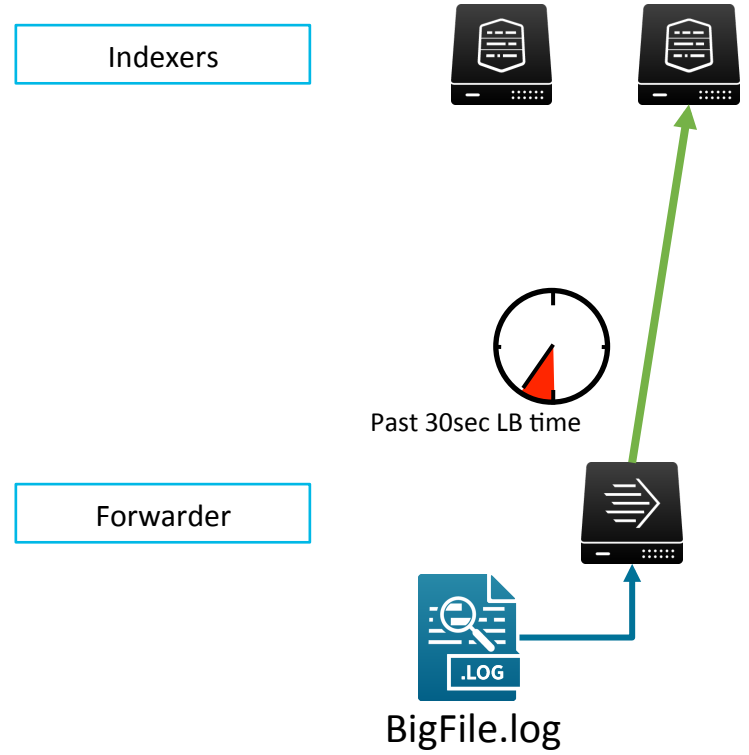
inputs.conf

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

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

Worst Practices

- 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
  - Settings that can help
    - [forceTimeBasedautoLB]
    - UF Event Breaking New!

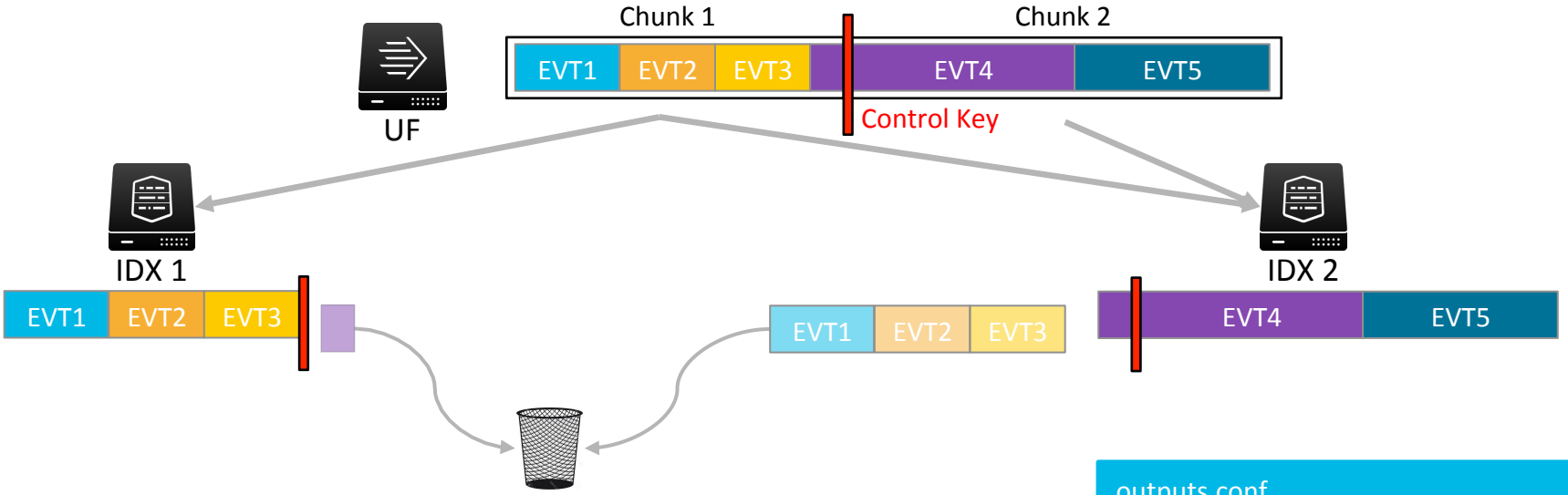


Best Practices

- If you're running 6.5 UFs…
  - Use UF event breaking 

- 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)

forceTimebasedAutoLB



outputs.conf

```
autoLB = true
autoLBFrequency = 30
forceTimebasedAutoLB = true
```

UF Event Breaking

- Brand Spankin' New 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

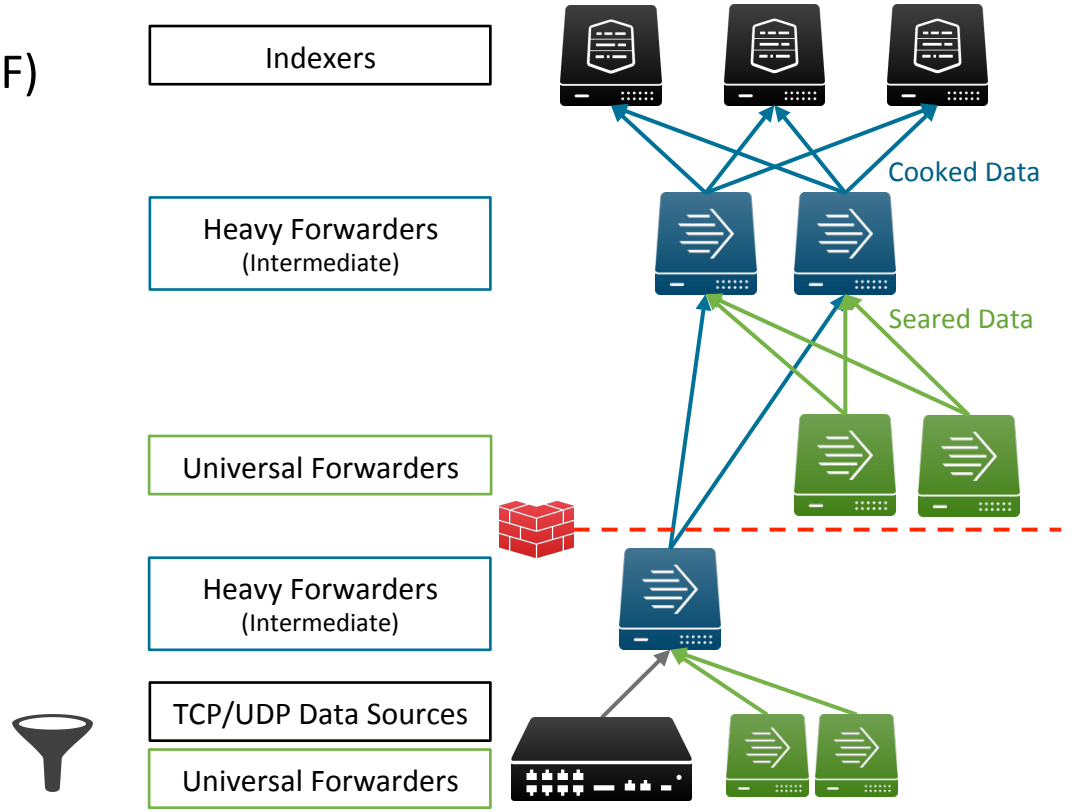
Intermediate forwarder

noun

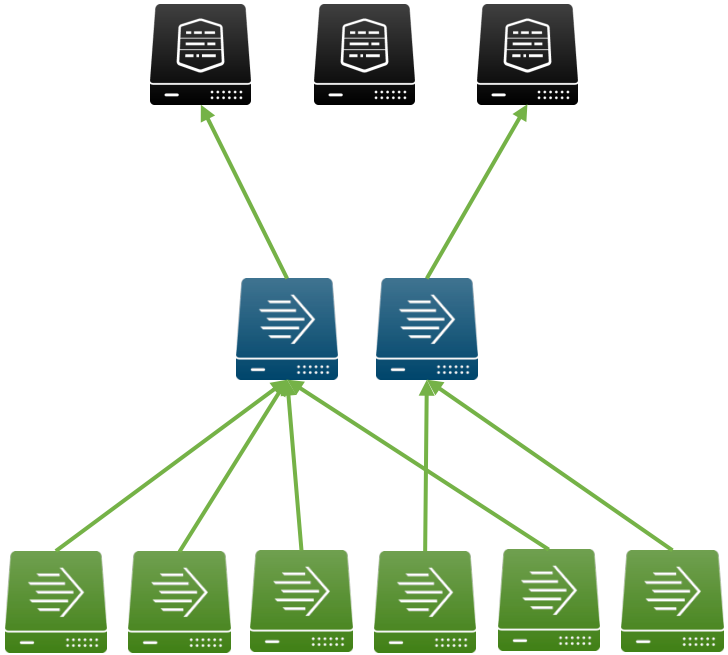
: A Splunk Forwarder, either Heavy or Universal, that sits between a Forwarder and an Indexer.

Worst Practice

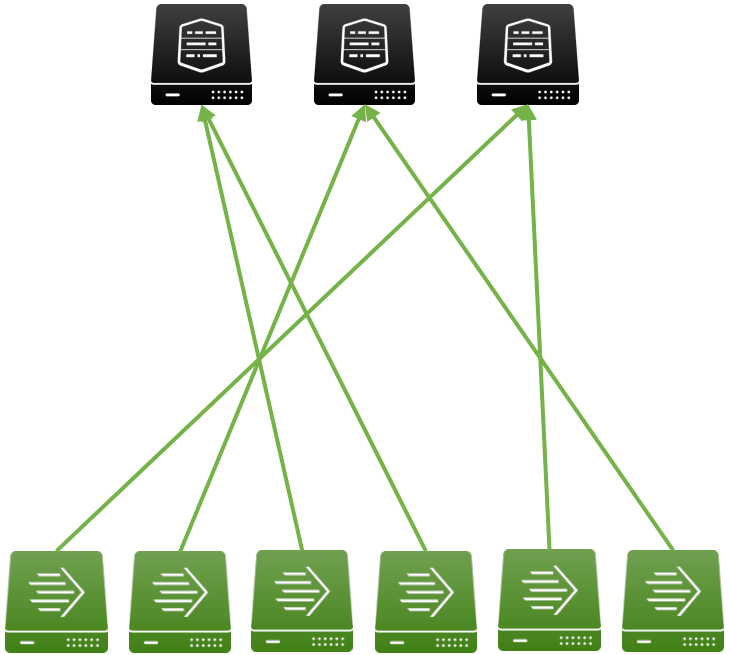
- Only use Heavy Forwarders (HWF) if there is a specific need
  - You need Python
  - Required by an App/Feature
    - HEC, DBX, Checkpoint, et…
  - Advanced Routing/Transformation
    - Routing individual events
    - Masking/SED
  - Need a UI on the Forwarder
- What's Wrong with my HWFs?
  - Additional administrative burden
    - More conf files needed on HWFs
    - Increases difficulty in troubleshooting
  - Cooked Data vs. Seared
  - UFs can usually do the same thing
    - Intermediate Forwarding
    - Routing based on data stream



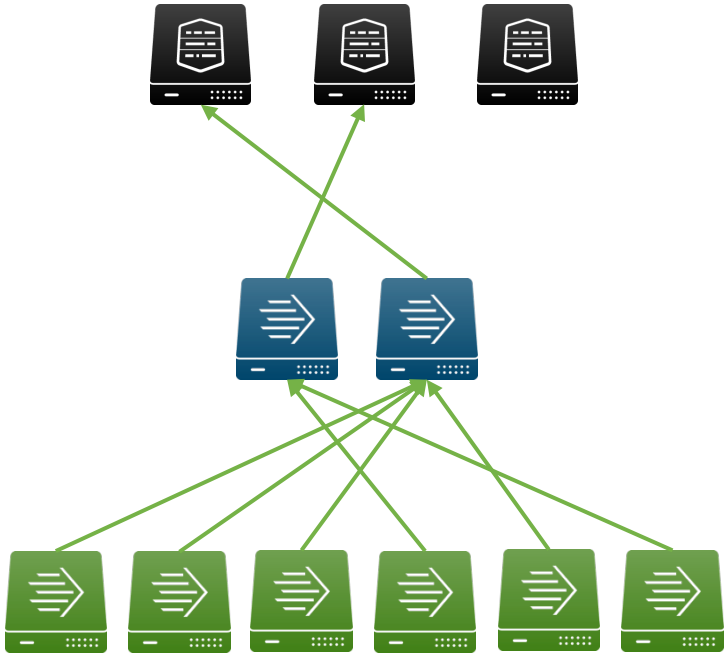
The Funnel Effect



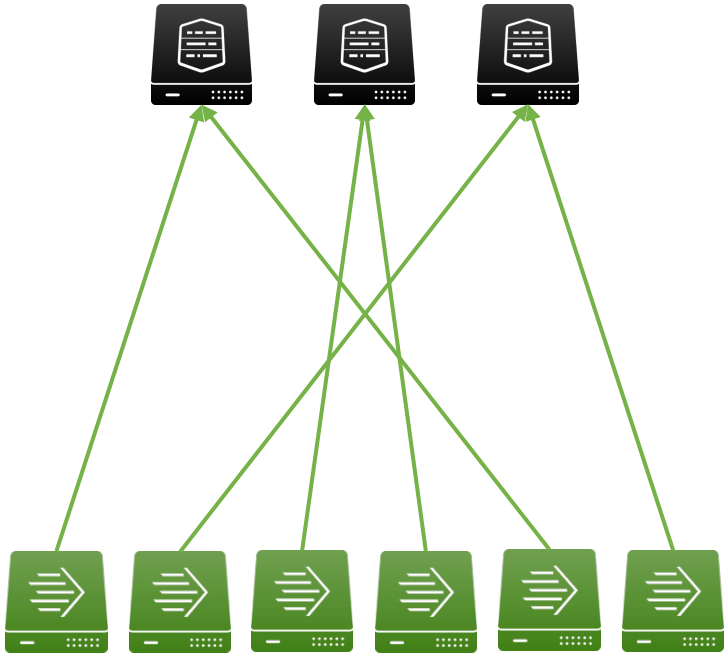
-vs-



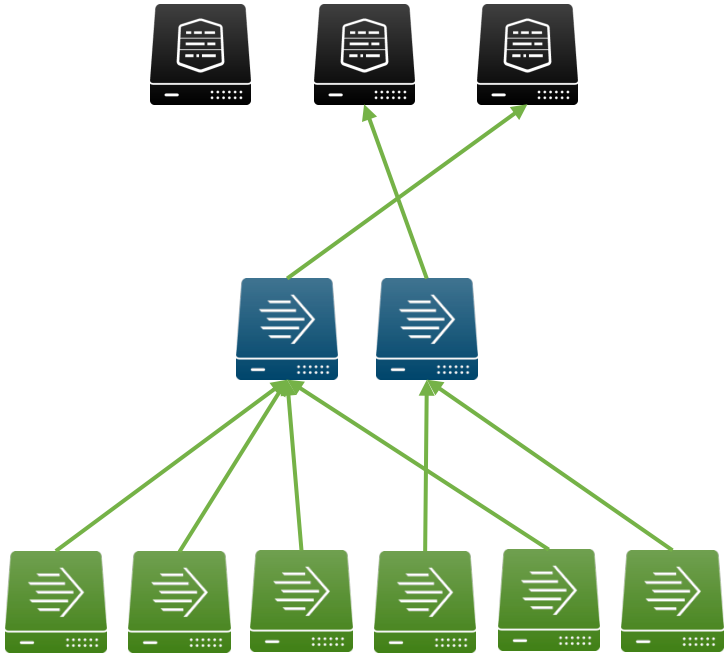
The Funnel Effect



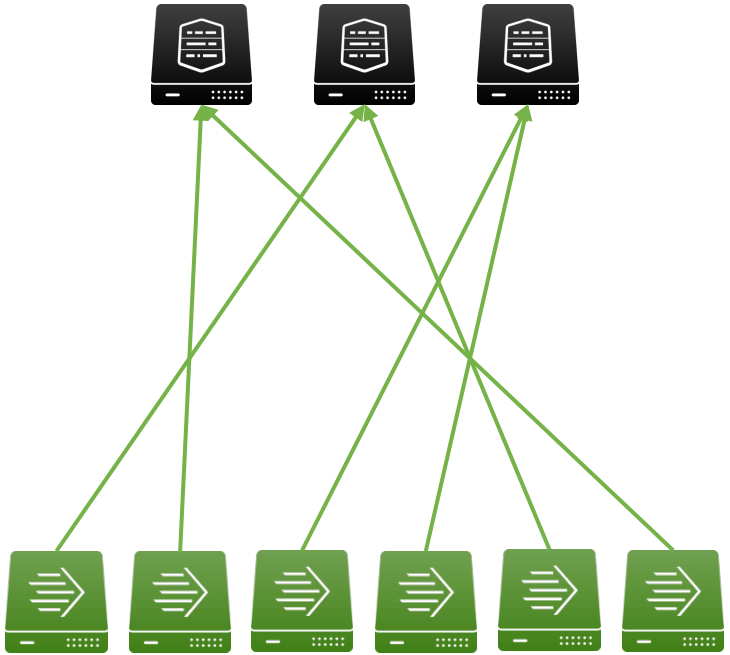
-vs-



The Funnel Effect

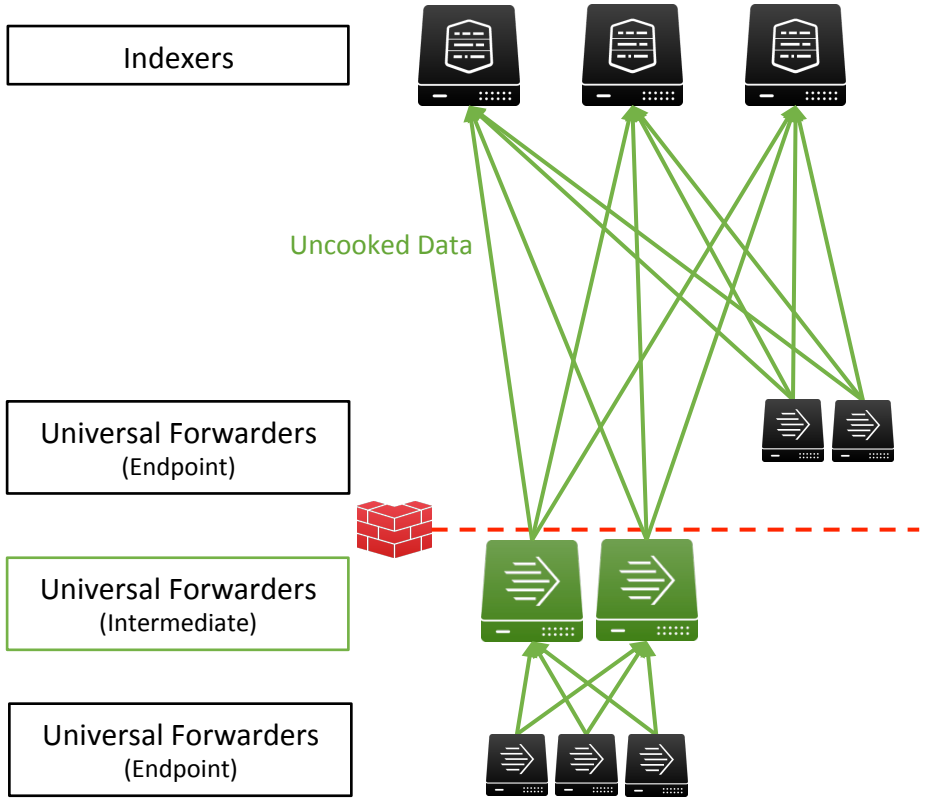


-vs-



Best Practice

- 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



Want To Know More?

- **Harnessing Performance and Scalability with Parallelization**
  by Tameem Anwar, Abhinav, Sourav Pal
  - Tuesday, Sept 27th 5:25PM – 6:10PM

Data Onboarding

Sourcetype Recognition

- Avoid automatic sourcetype recognition where possible
- Specify the sourcetype in *inputs.conf*

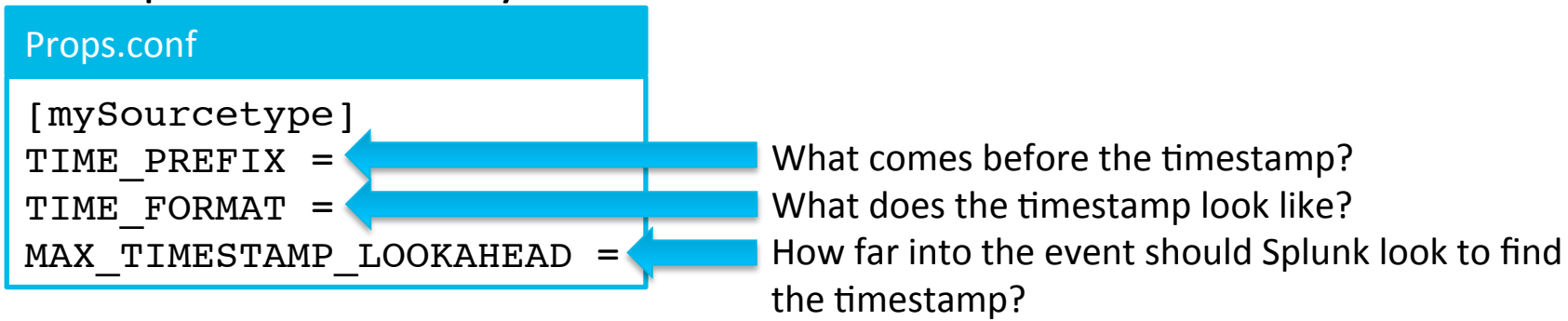
Inputs.conf

```
[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

- 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



Event Parsing

- 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

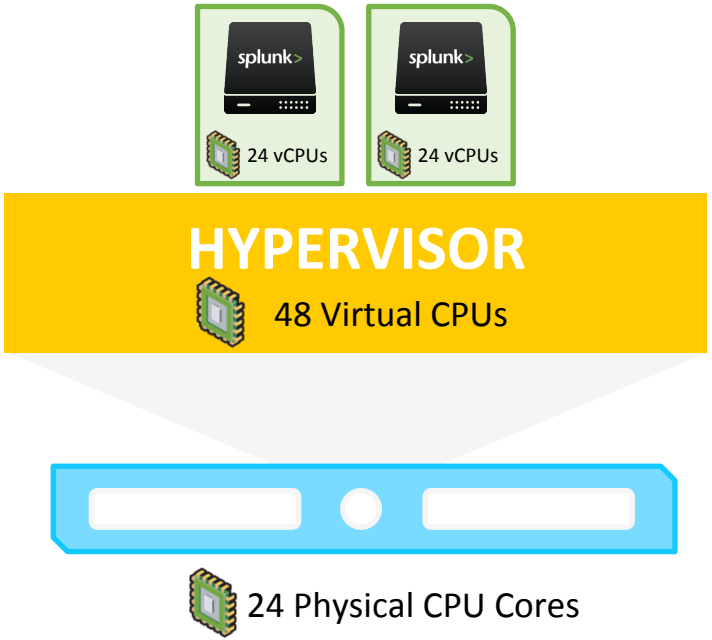
```
[mySourcetype]
SHOULD_LINEMERGE = false
LINE_BREAKER =
```

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

Virtualization

Worst Practice

- Who can spot the problem?
- vCPUs != Physical Cores
- Intel Hyper-threading
  - Doubles # of *logical* CPUs
  - Can improve performance 10-15%
    - Average gain
    - Some scenarios is 0% (dense searches)
    - Some scenarios it is more than 15%
  - Not magic

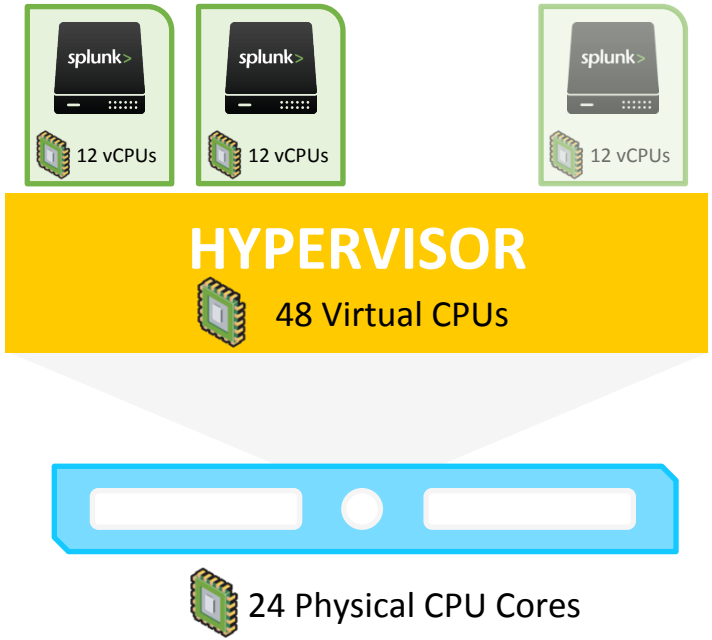


Best Practice

- Beware of Oversubscription
  - % Ready should be <2%
  - With available resources, consider adding VMs
- VM vCPU Allocation
  - Do not exceed physical cores
  - Allocate wide & flat
    - 1 core per virtual socket
- Know your NUMA boundaries
  - Align vCPU/Memory allocation
  - Smaller VMs are easier to align
- Don't put multiple Indexers on the same Host
  - Disk I/O is a big bottleneck, Indexers need a lot
- Consider increasing SH concurrency limits
  - Only if CPU utilization is low

Limits.conf

```
[search]
base_max_searches = 6
max_searches_per_cpu = 1
```



Indexed Extractions And Accelerations

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
Indexed Extraction Considerations

- The format is fixed or unlikely to change
  - Schema on the fly doesn't work 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
```

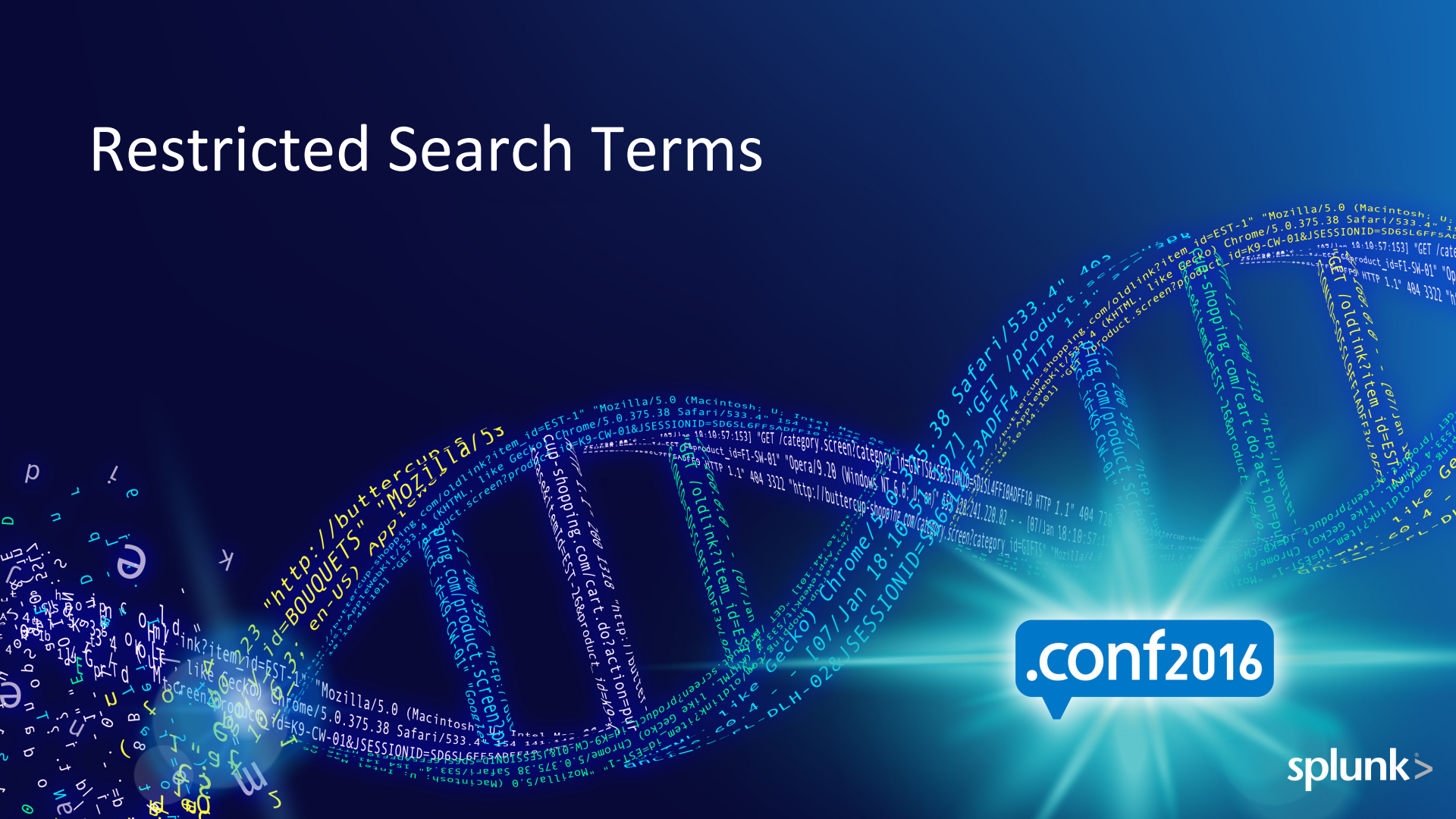
- Almost always filter using a specific key (field)
  - Categorical values (low cardinality)
  - Don't index KV pairs with high cardinality
- Frequently searching a large event set for rare data
  - KV pair that appears in a very small % of events
  - **foo!=bar** or **NOT foo=bar** and the field **foo** *nearly always* has the value of **bar**
- Don't go nuts!
  - Lots of indexed extractions = large indexes = slow performance
  - An Accelerated Data Model may be a better choice

Want To Know More?

Fields, Indexed Tokens and You by Martin Müller

- Wednesday, Sept 28th 11:00AM – 11:45PM

Restricted Search Terms



What Are Restricted Search Terms?

- 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

- 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
OR host=Figwit OR host=Lurtz OR host=Elendil OR host=Celeborn
```

Best Practice

- 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

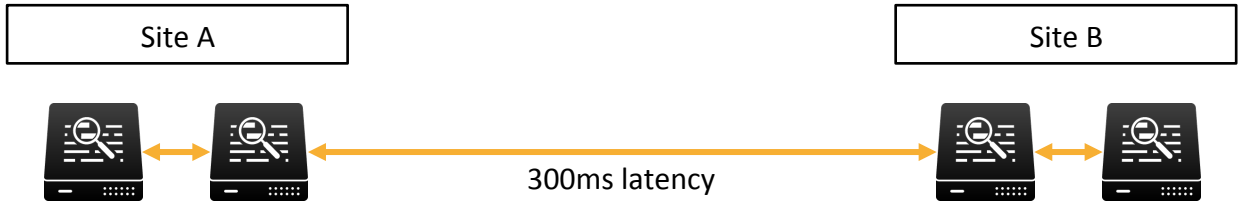
Multi-Site Search Head Clusters

Search Head Clustering

A Primer…

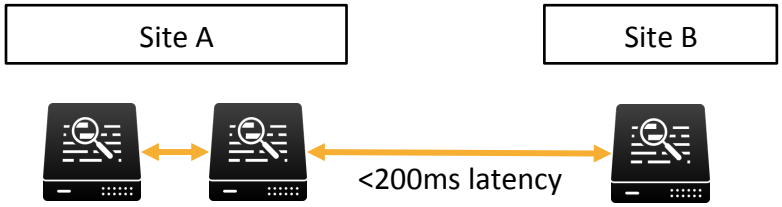
- SHC members elect a captain from their membership
- Minimum of 3 nodes required
  - Captain election vs. static assignment
- Odd # of SHC members is preferred
- Captain Manages
  - Knowledge object replication
  - Replication of scheduled search artifacts
  - Job scheduling
  - Bundle replication
- Multi-Site SHC does not exist
  - What?!
  - SHC is not site-aware
    - You're creating a stretched-SHC

Worst Practice



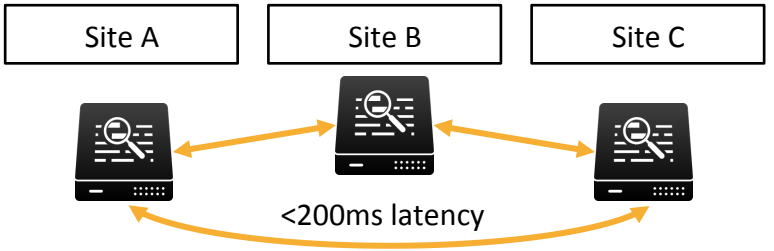
- Captain Election not possible with site or link failure
  - No site has node majority
    - Original SHC size: 4 Nodes
    - Node Majority: 3 Nodes
  - Odd # of SHC members is preferred
- WAN Latency is too high
  - We've tested up to 200ms

Best Practices



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



Three Sites: Fully Automatic Recovery

- Node majority can be maintained with a single site failure
- Keep Indexers in 2 sites
  - Simplifies index replication
  - Avoid sending jobs to SH in 3rd site

server.conf

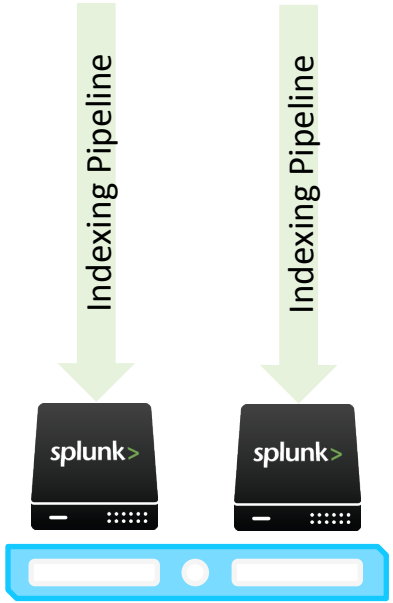
```
[shclustering]
adhoc_searchhead = true
```

Multi-Instance Indexers

Worst Practice

Two instances of Splunk on the same server

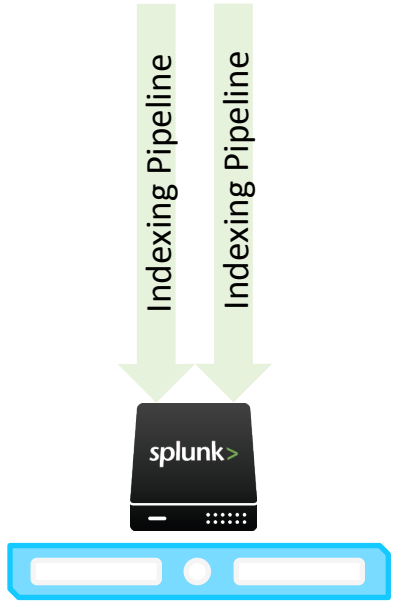
- Why would someone do this?
  - Prior to 6.3 Splunk was not able to fully utilize servers with high CPU density
- Additional Management & Overhead
  - Instances must be managed independently
    - More conf files
  - Unnecessary processes running for each instance
- Instances compete for system resources
  - CPU time, Memory, I/O



Best Practice

Single Instances with Parallelization

- Parallelization is your friend
  - Available in Splunk 6.3+
  - Single instance to manage
  - Multiple pipelines can be created for various features
    - Indexing, Accelerations, and Batch Searching
- Pay attention to system resources
  - Don't enable if you don't have excess CPU cores and I/O capacity

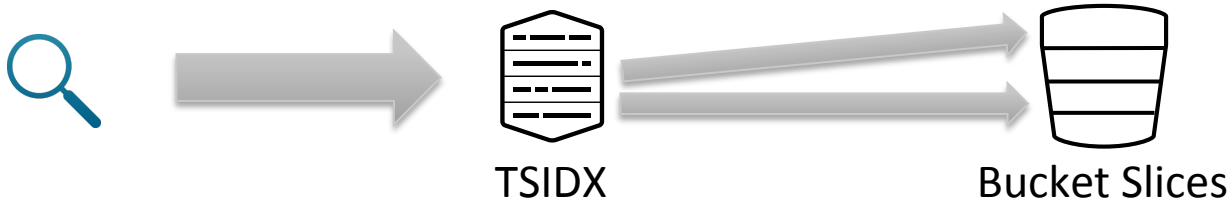


Index Management

Search Goals

How do I make my searches fast?

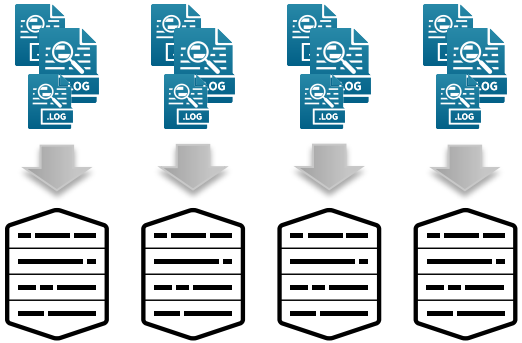
- Find what we're looking for quickly in the Index (TSIDX)
  - Lower cardinality in the dataset = fewer terms in the lexicon to search through
- Decompress as few bucket slices as possible to fulfill the search
  - More matching events in each slice = fewer slices we need to decompress
- Match as many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



Worst Practice

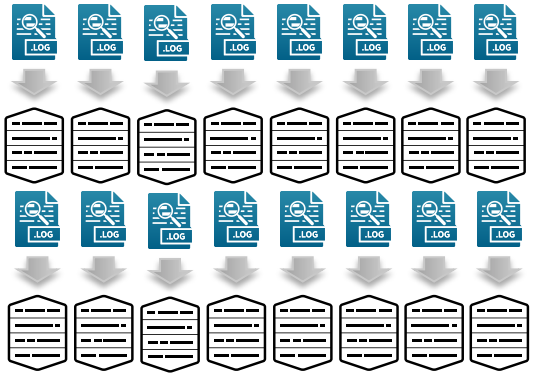
Goldilocks for Your Splunk Deployment

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

- 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
  - **6.3 & 6.4:** 1M Total buckets
  - **6.5:** 1.5M Total buckets
- 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

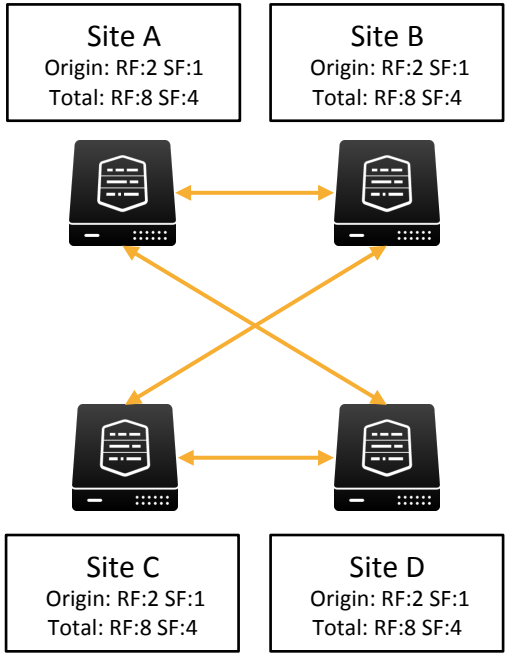
What If I Need Thousands Of Indexes To Secure My Data?

- 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
  - More on this later…

Index Replication

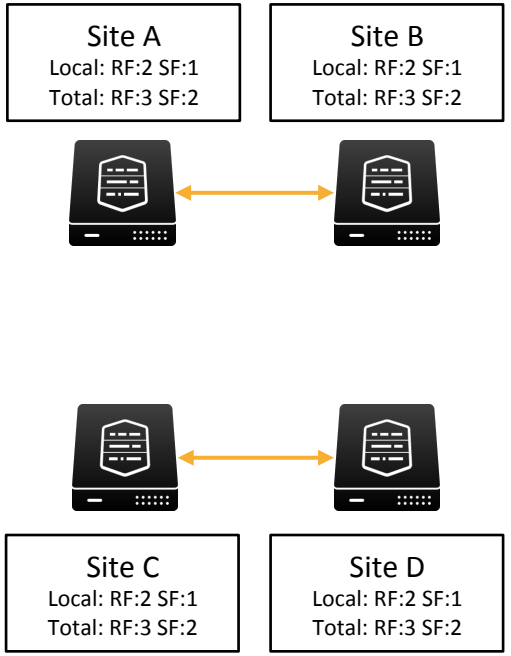
Worst Practice

- 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

- 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



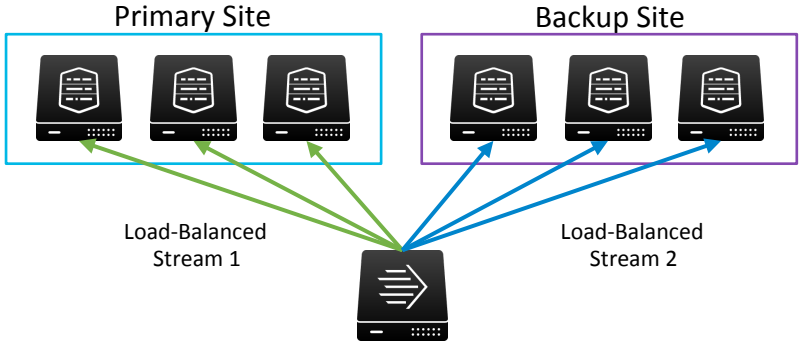
High Availability: MacGyver Style


He is the DR plan

Some Worst Practices

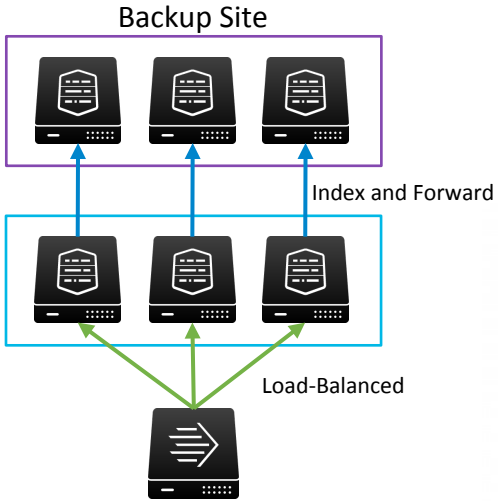
Cloned Data Streams

- 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

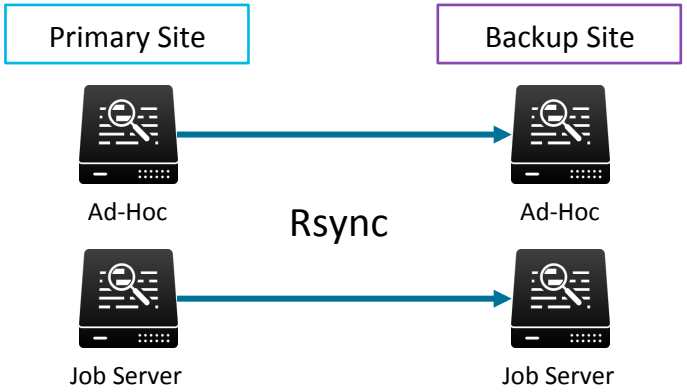
- RAID1-style HA
  - Failover to backup Indexer
- Forwarders must be redirected manually
- Complex recovery



Another Worst Practice

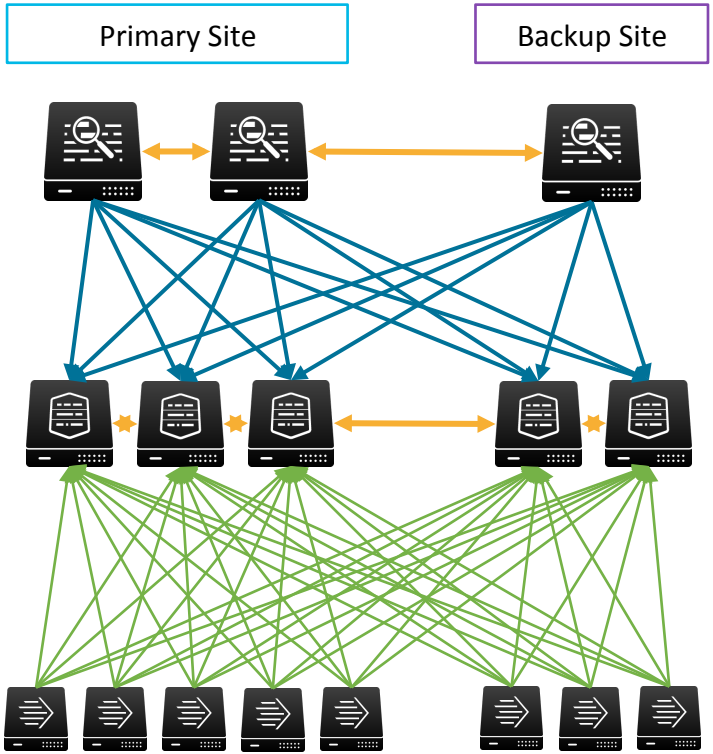
Rsync & Dedicated Job Servers

- 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



Some Best Practices

- 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



Want To Know More?

- **Indexer Clustering Internals, Scaling, and Performance** by Da Xu, Chole Yeung
  - Tuesday, Sept 27th 3:15 PM – 4:00 PM

- **Architecting Splunk for High Availability and Disaster Recovery** by Dritan Bitincka
  - Tuesday, Sept 27th 5:25PM – 6:10PM

What Now?

Related breakout sessions and activities…

- **Best Practices and Better Practices for Admins** by Burch Simon
  - Tuesday, Sept 27th 11:35AM – 12:20PM
- **It Seemed Like a Good Idea at the Time…Architectural Anti-Patterns** by David Paper Duane Waddle
  - Tuesday, Sept 27th 11:35AM – 12:20PM
- **Observations and Recommendations on Splunk Performance** by Dritan Bitincka
  - Wednesday, Sept 28th 12:05PM – 12:50PM

THANK YOU

splunk>