

Splunk Validated Architectures

January 2021

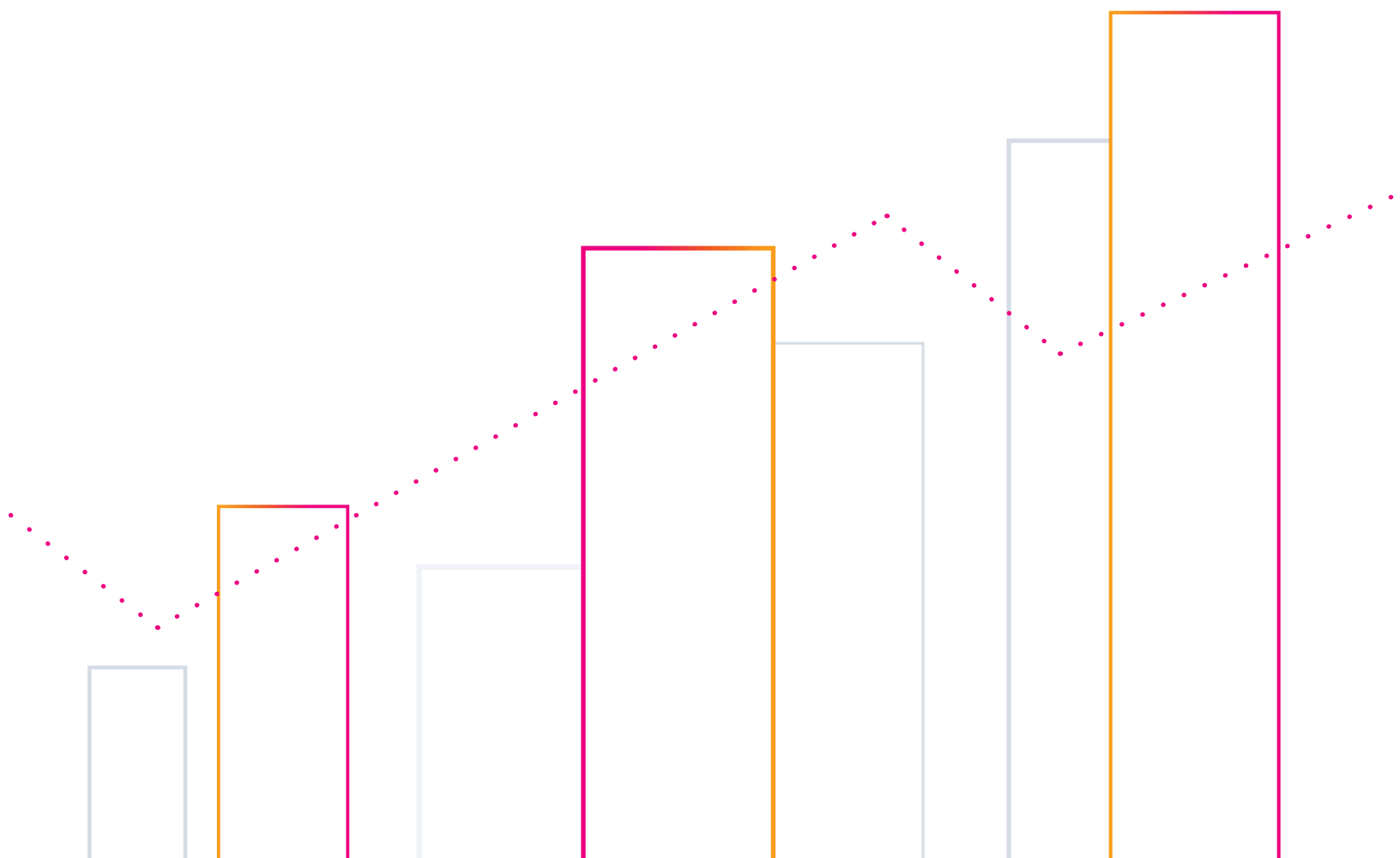


Table of Contents

Introduction.....	2
Document Structure	2
Reasons to Use Splunk Validated Architectures	3
Pillars of Splunk Validated Architectures	3
What to Expect From Splunk Validated Architectures	4
Roles and Responsibilities.....	4
Available Indexing and Search Topologies	5
Splunk Cloud Deployment Architecture (CLOUD)	6
Single Server Deployment (S1)	8
Distributed Non-Clustered Deployment (D1 / D11).....	9
Distributed Clustered Deployment - Single Site (C1 / C11)	11
Distributed Clustered Deployment + SHC - Single Site (C3 / C13)	13
Distributed Clustered Deployment - Multi-Site (M2 / M12).....	15
Distributed Clustered Deployment + SHC - Multi-Site (M3 / M13).....	17
Distributed Clustered Deployment + SHC - Multi-Site (M4 / M14).....	19
Splunk Indexer Architecture Options	21
Classic Indexer Architecture Using File System Storage.....	21
SmartStore Indexer Architecture Using Object Storage	22
Data Collection Architecture	23
Important Architectural Considerations and Why They Matter	23
General Data Collection Architecture Guidance	24
Data Collection Topology Overview.....	25
Data Collection Components	26
(UF) Universal Forwarder	26
(HF) Heavy Forwarder	26
(HEC) HTTP Event Collector	27
(DCN) Heavy Forwarder as Data Collection Node	29
(IF) Intermediary Forwarding Tier	30
(KAFKA) Consuming Log Data From Kafka Topics.....	32
(KINESIS) Consuming Log Data From Amazon Kinesis Firehose	33
(METRICS) Metrics Collection	34
(SYSLOG) Syslog Data Collection.....	35
Splunk Connect for Syslog (SC4S - recommended best practice)	35
Syslog (File monitoring in conjunction with a syslog server).....	37
Splunk UDP Input	38
High-Availability Considerations for Forwarding Tier components	38
Design Principles and Best Practices.....	39
Deployment Tiers.....	39
Aligning Your Topology With Best Practices.....	39
Best Practices: Tier-Specific Recommendations	39
Search Tier Recommendations	40
Indexing Tier Recommendations	41
Collection Tier Recommendations.....	42
Management / Utility Tier Recommendations.....	43
Summary & Next Steps	44
Appendix	45
Appendix "A": SVA Pillars Explained	45
Appendix "B": Topology Components.....	46

Introduction

Splunk Validated Architectures (SVAs) are proven reference architectures for stable, efficient and repeatable Splunk deployments. Many of Splunk's existing customers have experienced rapid adoption and expansion, leading to certain challenges as they attempt to scale. At the same time, new Splunk customers are increasingly looking for guidelines and certified architectures to ensure that their initial deployment is built on a solid foundation. SVAs have been developed to help our customers with these growing needs.

Whether you are a new or existing Splunk customer, SVAs will help you build an environment that is easier to maintain and simpler to troubleshoot. SVAs are designed to provide you with the best possible results while minimizing your total cost of ownership. Additionally, your entire Splunk foundation will be based on a repeatable architecture which will allow you to scale your deployment as your needs evolve over time.

SVAs offer topology options that consider a wide array of organizational requirements, so you can easily understand and find a topology that is right for your requirements. The Splunk Validated Architectures selection process will help you match your specific requirements to the topology that best meets your organization's needs. If you are new to Splunk, we recommend implementing a Validated Architecture for your initial deployment. If you are an existing customer, we recommend that you explore the option of aligning with a Validated Architecture topology. Unless you have unique requirements that make it necessary to build a custom architecture, it is very likely that a Validated Architecture will fulfill your requirements while remaining cost effective.

This document contains all SVA topologies that are available at time of publication. For a custom document that meets your specific requirements, please use the Interactive Splunk Validated Architecture (ISVA) tool available [here](#). The custom document will reflect the best practice approach to search, indexing and data collection architecture, given your specific requirements identified when working with the tool.

It is always recommended that you involve Splunk or a trusted Splunk partner to ensure that the recommendations in this document meet your needs.

If you need assistance implementing a Splunk Validated Architecture, contact [Splunk Professional Services](#).

Document Structure

SVAs are broken into three major content areas:

1. Indexing and search topology
2. Data collection architecture components
3. Design principles and best practices

Indexing and search covers the architecture tiers that provide the core indexing and search capabilities of a Splunk deployment. The data collection component section guides you in choosing the right data collection mechanism for your requirements.

Design principles and best practices apply to your architecture as a whole and will help you make the correct choices when working out the details of your deployment.

Reasons to Use Splunk Validated Architectures

Implementing a validated architecture will empower you to design and deploy Splunk more confidently. SVAs will help you solve some of the most common challenges that organizations face, including:

Performance

- Organizations want to see improvements in performance and stability

Complexity

- Organizations sometimes run into the pitfalls of custom-built deployments, especially when they have grown too rapidly or organically. In such cases, unnecessary complexity may have been introduced into the environment. This complexity can become a serious barrier when attempting to scale

Efficiency

- To derive the maximum benefits from the Splunk deployment, organizations must improve the efficiency of operations and accelerate time to value

Cost

- Organizations are seeking ways to reduce total cost of ownership (TCO), while fulfilling all of their requirements

Agility

- Organizations will need to adapt to change as they scale and grow

Maintenance

- Optimization of the environment is often necessary in order to reduce maintenance efforts

Scalability

- Organizations must have the ability to scale efficiently and seamlessly

Verification

- Stakeholders within the organization want the assurance that their Splunk deployment is built on best practice

Pillars of Splunk Validated Architectures

Splunk Validated Architectures are built on the following foundational pillars. For more information on these design pillars, refer to Appendix "A" below.

AVAILABILITY	PERFORMANCE	SCALABILITY	SECURITY	MANAGEABILITY
The system is continuously operational and able to recover from planned and unplanned outages or disruptions.	The system can maintain an optimal level of service under varying usage patterns.	The system is designed to scale on all tiers, allowing you to handle increased workloads effectively .	The system is designed to protect data, configurations, and assets while continuing to deliver value.	The system is centrally operable and manageable across all tiers .

These pillars are in direct support of the **Platform Management & Support** Service in the Splunk Center Of Excellence model.

What to Expect From Splunk Validated Architectures

Please note that SVAs do not include deployment technologies or deployment sizing. The reasoning for this is as follows:

- Deployment technologies, such as operating systems and server hardware, are considered implementation choices in the context of SVAs. Different customers will have different choices, so a generalization is not easily possible.
- Deployment sizing requires an evaluation of data ingest volume, data types, search volumes and search use cases, which tend to be very customer-specific and generally have no bearing on the fundamental deployment topology. When you are ready, please reach out to Splunk for help with properly sizing your deployment based on your expected ingest and search workload profile.

Summary of Current SVA Guidance:

SVAs <u>will</u> provide:	SVAs do <u>not</u> currently provide:
✓ Clustered and non-clustered deployment options. ✓ Diagrams of the reference architecture. ✓ Guidelines to help you select the architecture that is right for you ✓ Tier-specific recommendations. ✓ Best practices for building out your Splunk deployment ✓ Best practices for connecting to Splunk Cloud	✗ Implementation choices (OS, baremetal vs. virtual vs. Cloud etc.). ✗ Deployment sizing. ✗ A prescriptive approval of your architecture. Note: SVAs provide recommendations and guidelines, so you can ultimately make the right decision for your organization. ✗ A topology suggestion for every possible deployment scenario. In some cases, unique factors may require a custom architecture to be developed. Splunk experts are available to help with any custom solutions you need. If you are an existing customer, reach out to your Splunk Account Team. If you are new to Splunk, you can reach us here.

Roles and Responsibilities

Splunk Validated Architectures are highly relevant to the concerns of decision makers and administrators. Enterprise architects, consultants, Splunk administrators and managed service providers should all be involved in the SVA selection process. You will find a description of each of these roles below:

Role	Description
Enterprise Architects	Responsible for architecting Splunk deployments to meet enterprise needs.
Consultants	Responsible for providing services for Splunk architecture, design, and implementation.
Splunk Engineers	Responsible for managing the Splunk lifecycle.
Managed Service Providers	Entities that deploy and run Splunk as a service for customers.

Available Indexing and Search Topologies

Note

If you chose Splunk Cloud to run indexing and search, the topology for these tiers is chosen by Splunk based on your licensing agreement. You can jump directly to the section on Data Collection components in this document, which you will still need to consider for getting data into Splunk Cloud.

Note

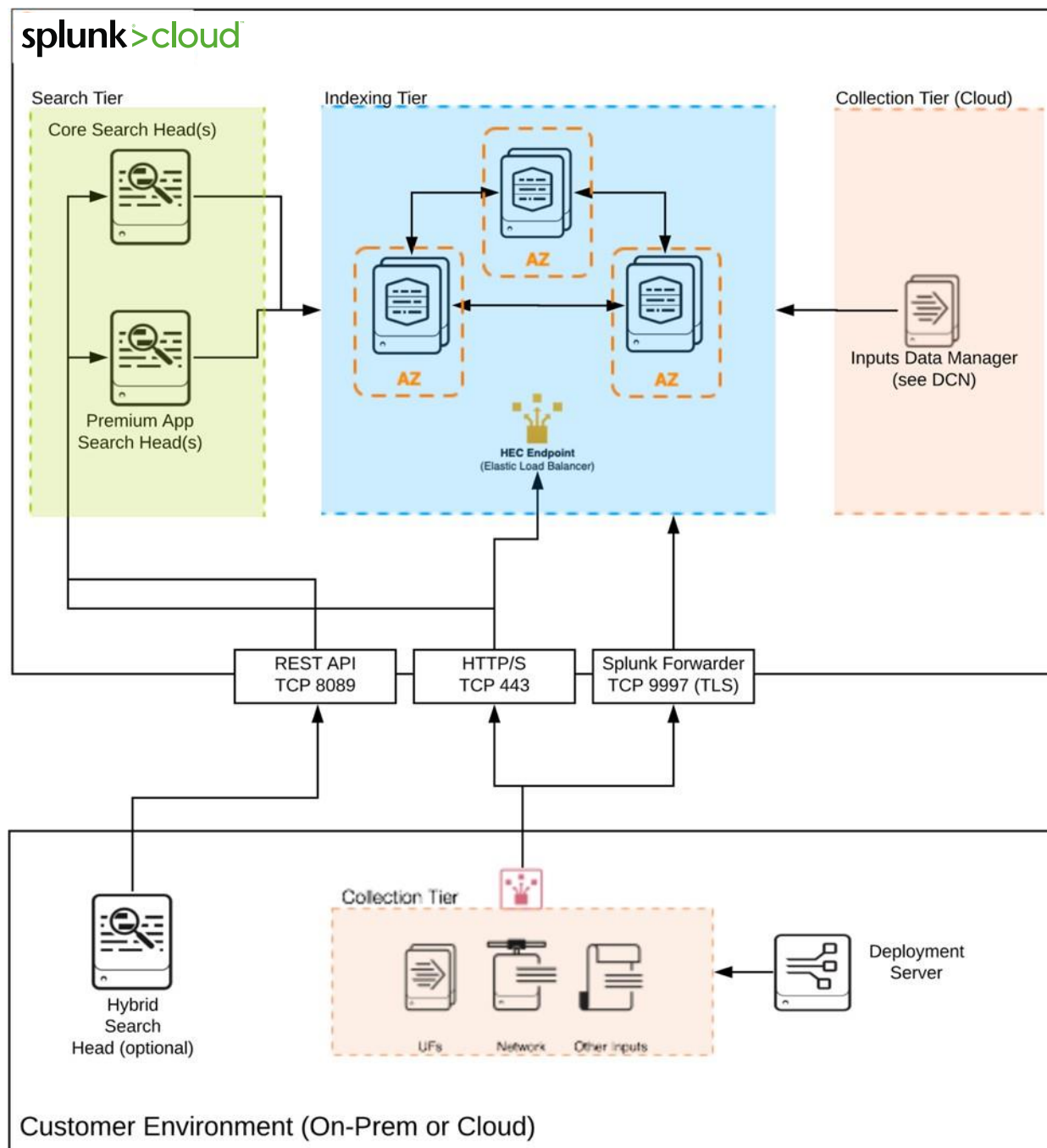
While you may choose to implement any topology that provides additional benefits beyond your immediate needs, keep in mind that this will likely result in unnecessary costs. Moreover, the introduction of additional complexity is often counterproductive to operational efficiency.

Important note about topology diagrams

The icons in the topology diagrams represent **functional Splunk roles** and do not imply dedicated infrastructure to run them. Please see the appendix for guidance as to which Splunk roles can be co-located on the same infrastructure/server.

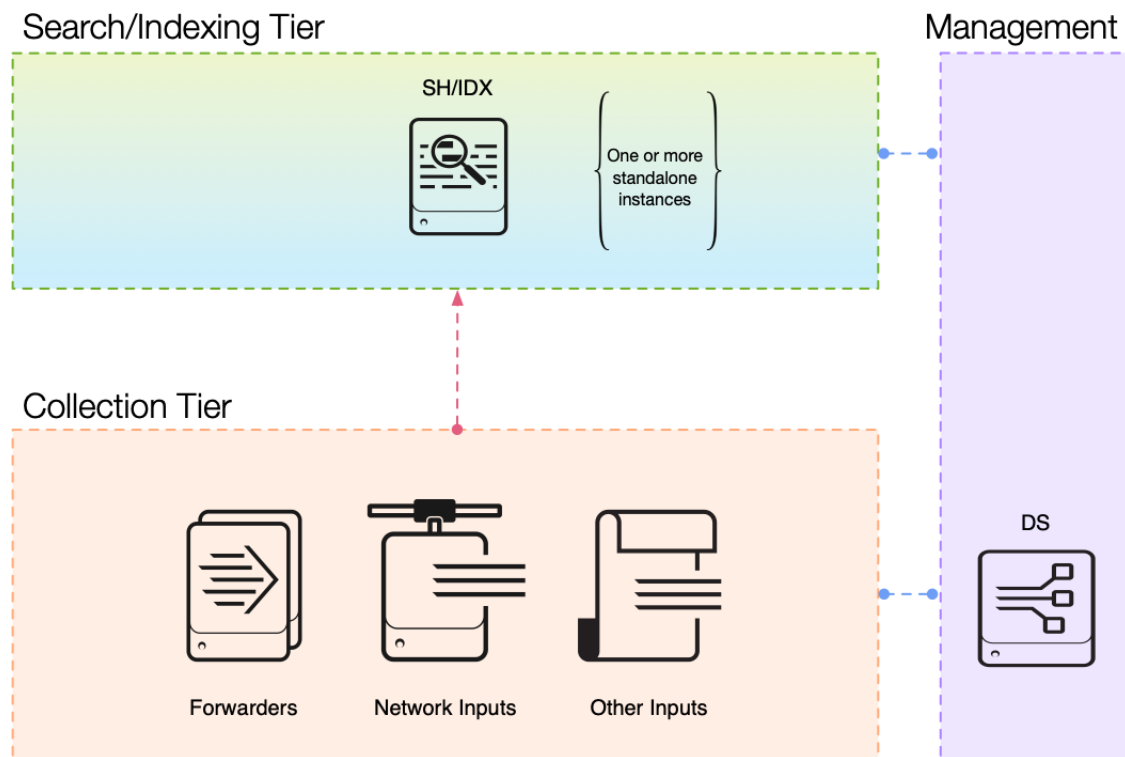
Splunk Cloud Deployment Architecture (CLOUD)

The following diagram represents the high-level architecture of a Splunk Cloud deployment and shows the integration points with your environment:



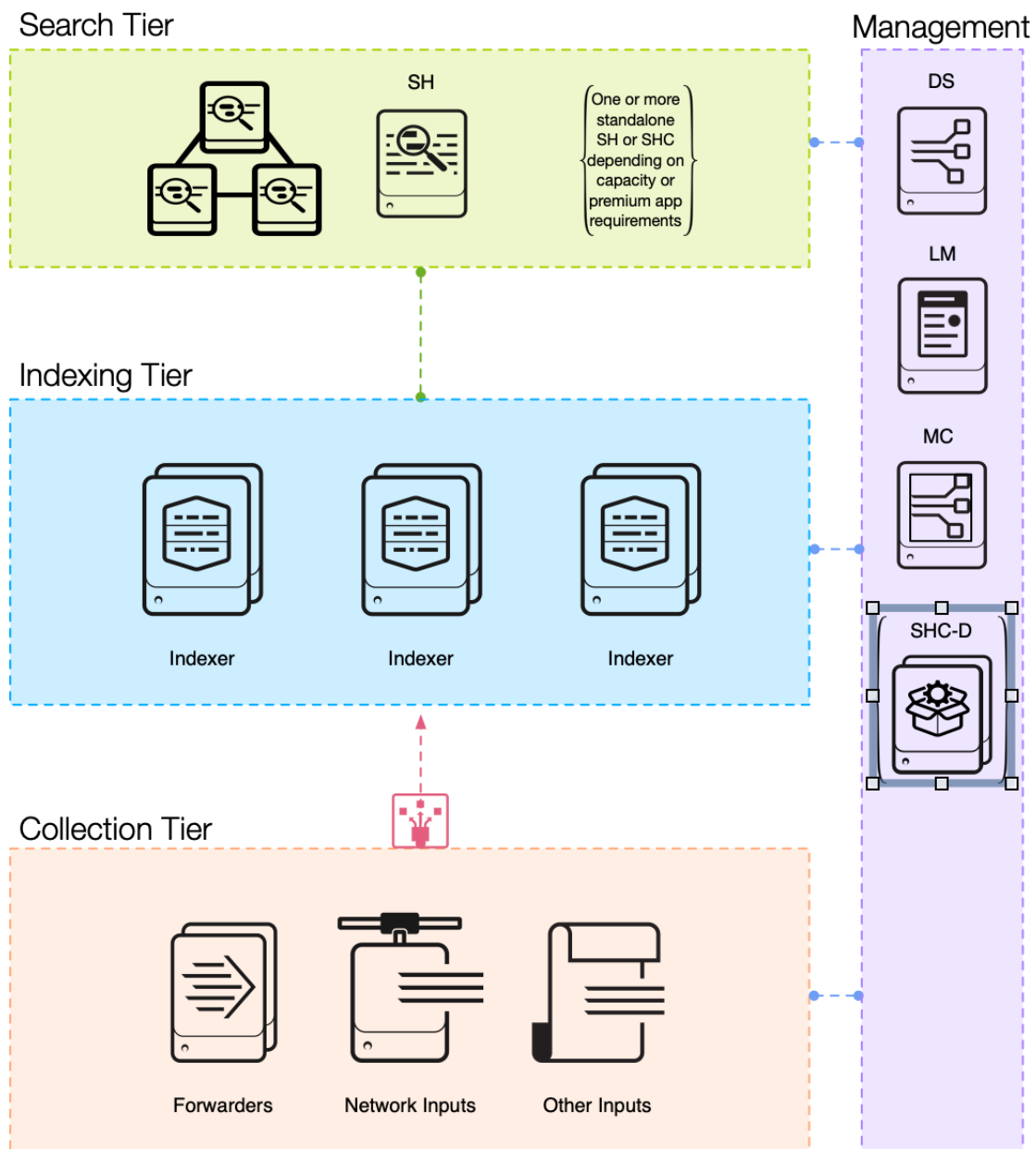
Description of the Splunk Cloud Deployment (CLOUD)	Limitations
When you are choosing Splunk Cloud, all deployment decisions regarding your indexing and search topologies have already been made for you. The Splunk Cloud team will build and operate your dedicated (single-tenant) AWS environment in a way that allows for meeting Splunk's compliance requirements and our service SLAs with you. The indexers that support your cloud environment are distributed across multiple fault domains to ensure a highly available service within a single region (for a list of supported regions, refer to the Splunk Cloud documentation). Besides listening on the standard TCP port (9997) for forwarder traffic, you can also send data via the HTTP Event Collector (HEC, see details on HEC in the data collection section of this document). You may also request access to the REST API endpoints via port 8089 via a support ticket to support programmatic access. The search head(s) are deployed into their own security group. Depending on your Splunk Cloud service agreement, that will either be a single search head or a search head cluster. If you license a premium app (like the Splunk App for Enterprise Security [ES]), Splunk Cloud will provision a dedicated search head as appropriate to meet the premium app's requirements. If you intend to run ES, you can forward adaptive response actions to your on-prem environment using the Adaptive Response Relay as described here. All aspects of operating and managing this AWS environment are the responsibility of the Splunk Cloud Operations team, so you can focus on onboarding data and getting value out of your Splunk deployment. Data archiving and restoring is supported.	All apps need to be vetted and certified by Splunk to ensure security of your environment. At the time of this writing, 900+ apps available on Splunkbase have been vetted and are Splunk Cloud Certified. Hybrid Search support is limited to an on-prem Search Head searching Splunk Cloud indexers, not vice versa. The IDM (Inputs Data Manager) shown in the diagram is the Splunk Cloud-managed implementation of a Data Collection Node (DCN) that supports scripted and modular inputs only. For data collection needs beyond that, you can deploy and manage a DCN in your environment using a Splunk Heavy Forwarder. Splunk Cloud will not manage or monitor your on-prem Splunk components.

Single Server Deployment (S1)



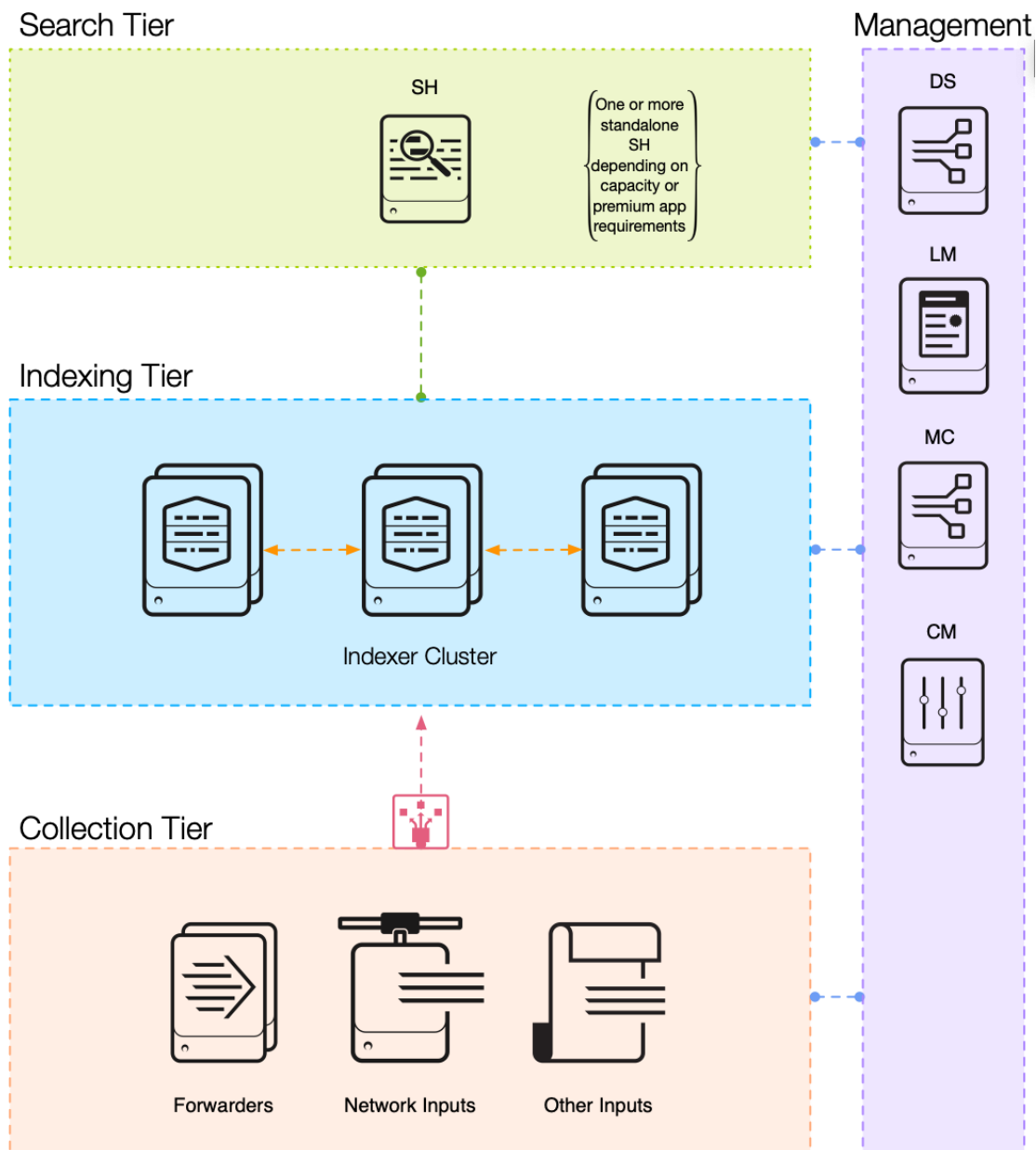
Description of the Single Server Deployment (S1)	Limitations
This deployment topology provides you with a very cost-effective solution if your environment meets all of the following criteria: a) You do not have any requirements to provide high-availability or automatic disaster recovery for your Splunk deployment b) Your daily data ingest is under ~300GB/day and c) You have a small number of users with non-critical search use cases This topology is typically used for smaller, non business-critical use-cases (often departmental in nature). Appropriate use cases include data onboarding test environments, small DevOps use cases, application test and integration environments and similar scenarios. The primary benefits of this topology include easy manageability, good search performance for smaller data volumes and a fixed TCO. Multiple independent single-instance deployments can be managed by a single management tier, as needed.	• No High Availability for Search/Indexing • Scalability limited by hardware capacity (straightforward migration path to a distributed deployment)

Distributed Non-Clustered Deployment (D1 / D11)



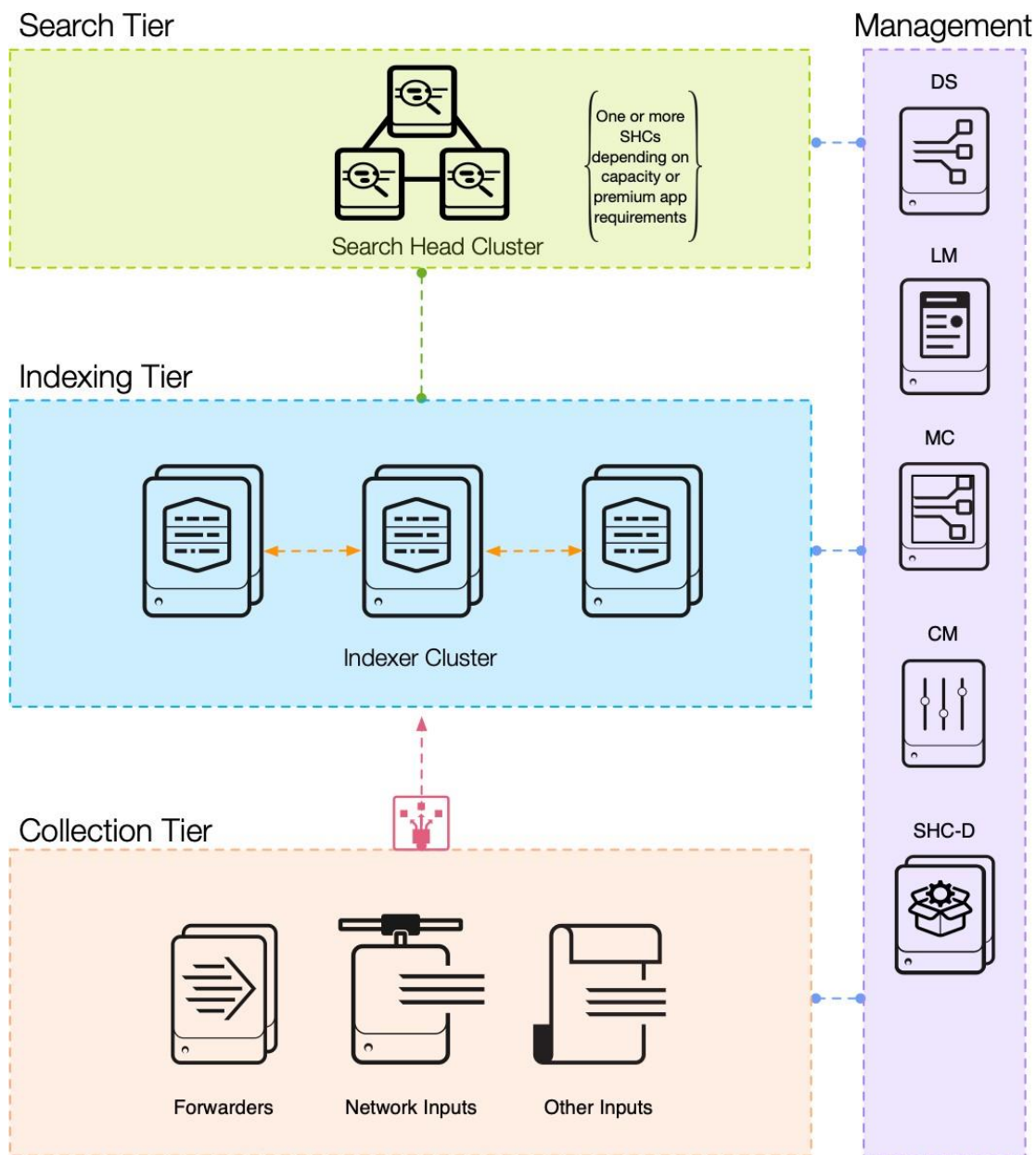
Description of the Distributed Non-Clustered Deployment (D1 / D11)	Limitations
You require a distributed topology in either of the following situations: a) Your daily data volume to be sent to Splunk exceeds the capacity of a single-server deployment or b) You want/need to provide highly available data ingest Deploying multiple, independent indexers will allow you to scale your indexing capacity linearly and implicitly increase the availability for data ingest. The TCO will increase predictably and linearly as you add indexer nodes. The recommended introduction of the Monitoring Console (MC) component allows you to monitor the health and capacity of your distributed deployment. Additionally, the MC provides a centralized alerting system so you will be notified of unhealthy conditions in your deployment. The search head(s) will need to be explicitly configured with the list of available search peers every time new indexers are added. Note for ES Customers: If your category code is D11 (i.e. you intend to deploy the Splunk App for Enterprise Security), a single dedicated search head is required to deploy the app (this is not pictured in the topology diagram). The collection tier needs to be configured with the list of target indexers (via a deployment server) every time new indexers are added. This deployment topology can scale linearly to over 1000 indexer nodes and can thus support extremely high data ingest and search volumes. Search performance can be maintained across large datasets through parallel search execution across many indexers (map/reduce). While not specifically broken out as a separate topology, a search head cluster can be used to increase search capacity on the search tier. Depending on specific requirements for total search capacity and/or premium apps, multiple standalone Search Heads or a Search Head Cluster can be configured and managed centrally.	• No High Availability for Search Tier • Limited High Availability for indexing tier, node failure may cause incomplete search results for historic searches, but will not impact data ingest.

Distributed Clustered Deployment - Single Site (C1 / C11)



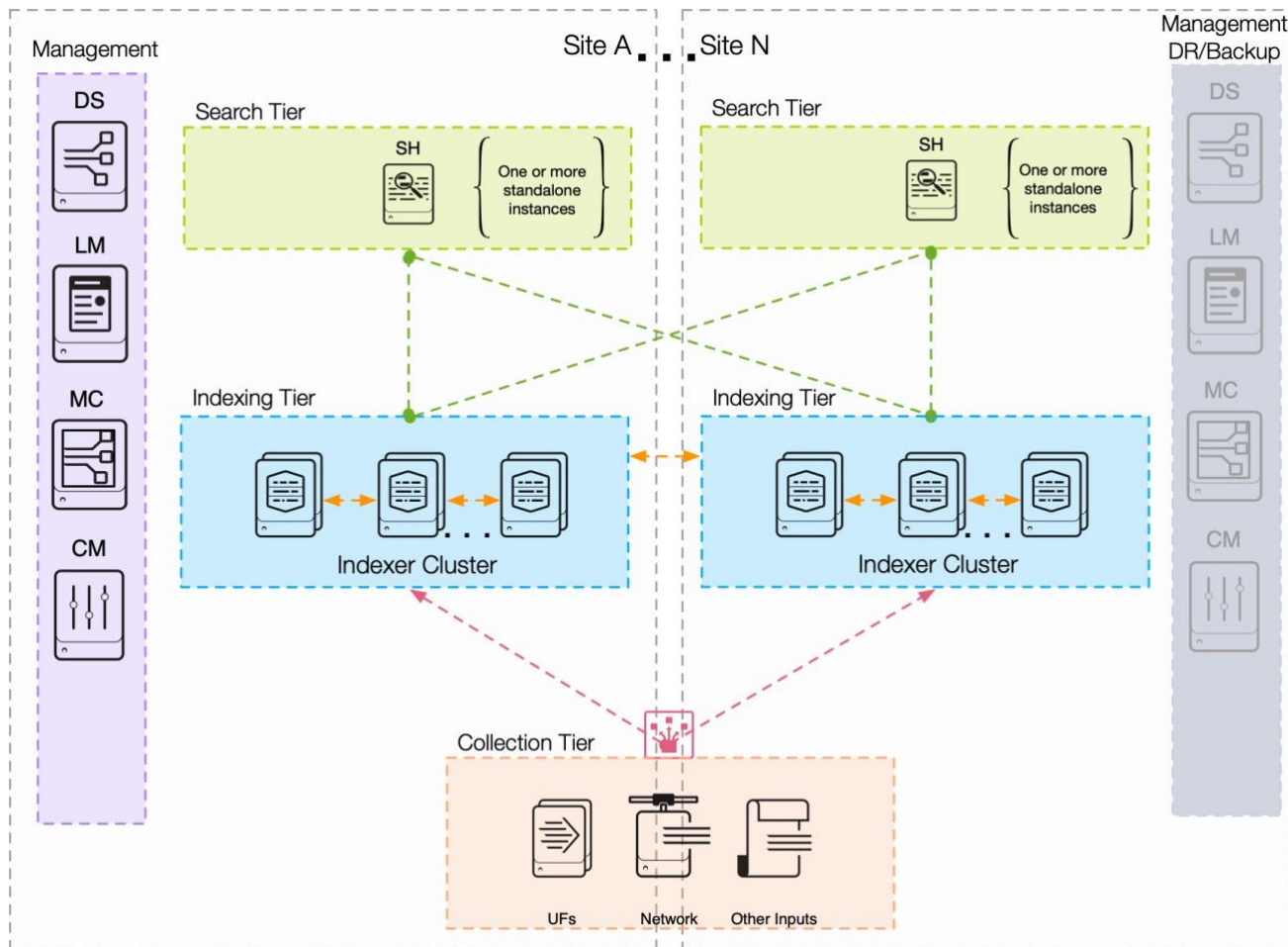
Description of Distributed Clustered Deployment - Single Site (C1 / C11)	Limitations
This topology introduces indexer clustering in conjunction with an appropriately configured data replication policy. This provides high-availability of data in case of indexer peer node failure. However, you should be aware that this applies only to the indexing tier and does not protect against search head failure. Multiple independent search heads can be used for availability/capacity reasons or to run Splunk premium app solutions, like ES, although the recommended approach to scale search capacity is to employ Search Head Clustering. Note for ES customers: If your category code is C11 (i.e. you intend to deploy the Splunk App for Enterprise Security), a single dedicated search head is required to deploy the app (this is not pictured in the topology diagram). This topology requires an additional Splunk component named the Cluster Manager (CM). The CM is responsible for coordination and enforcement of the configured data replication policy. The CM also serves as the authoritative source for available cluster peers (indexers). Search Head configuration is simplified by configuring the CM instead of individual search peers. You can optionally configure the forwarding tier to discover available indexers via the CM. This simplifies the management of the forwarding tier. Be aware that data is replicated within the cluster in a non-deterministic way. You will not have control over where requested copies of each event are stored. Additionally, while scalability is linear, there are limitations with respect to total cluster size (~50PB of searchable data under ideal conditions). Splunk recommends deployment of the Monitoring Console (MC) to monitor the health of your Splunk environment.	• No High Availability for Search Tier • Total number of unique buckets in indexer cluster limited to 20MM (V8.x), 40MM total buckets • No automatic DR capability in case of data center outage

Distributed Clustered Deployment + SHC - Single Site (C3 / C13)



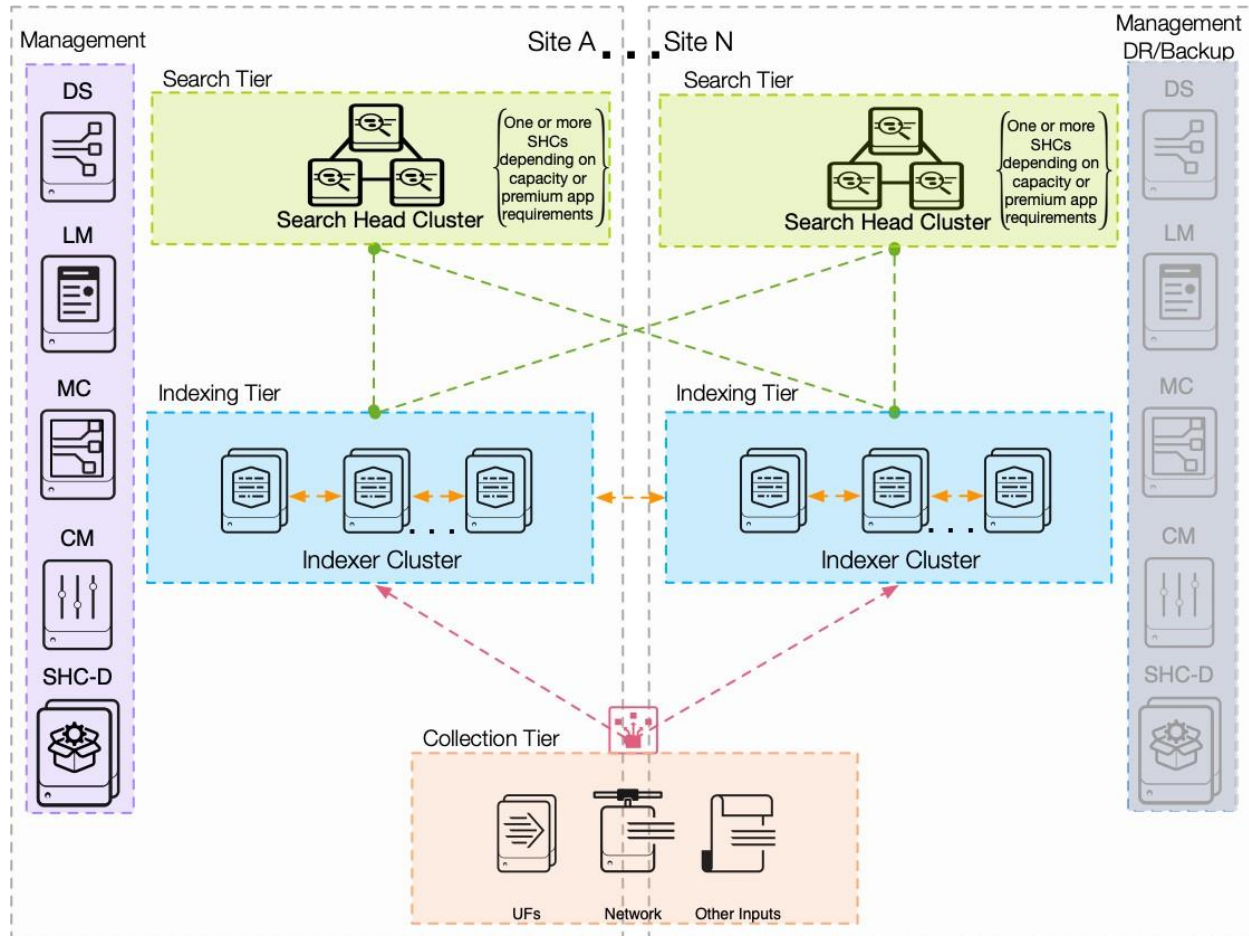
Description of Distributed Clustered Deployment + SHC - Single Site (C3 / C13)	Limitations
Search Head Clustering (SHC) adds horizontal scalability and removes the single point of failure from the search tier. A minimum of three search heads are required to implement a SHC. To manage the SHC configuration, an additional Splunk component called the Search Head Cluster Deployer is required for each SHC. This component is necessary in order to deploy changes to configuration files in the cluster. The Search Head Cluster Deployer has no HA requirements (no runtime role). The SHC provides the mechanism to increase available search capacity beyond what a single search head can provide. Additionally, the SHC allows for scheduled search workload distribution across the cluster. The SHC also provides optimal user failover in case of a search head failure. A network load-balancer that supports sticky sessions is required in front of the SHC members to ensure proper load balancing of users across the cluster. Note for ES customers: If your category code is C13 (i.e. you intend to deploy the Splunk App for Enterprise Security), a dedicated search head cluster is required to deploy the app (this is not pictured in the topology diagram). The search tier can contain clustered and non-clustered Search Heads depending on your capacity and organizational needs (this is also not pictured in the topology diagram).	• No DR capability in case of data center outage ES requires dedicated SH/SHC • Professional services recommended for implementing ES on a SHC • SHC cannot have more than 100 nodes

Distributed Clustered Deployment - Multi-Site (M2 / M12)



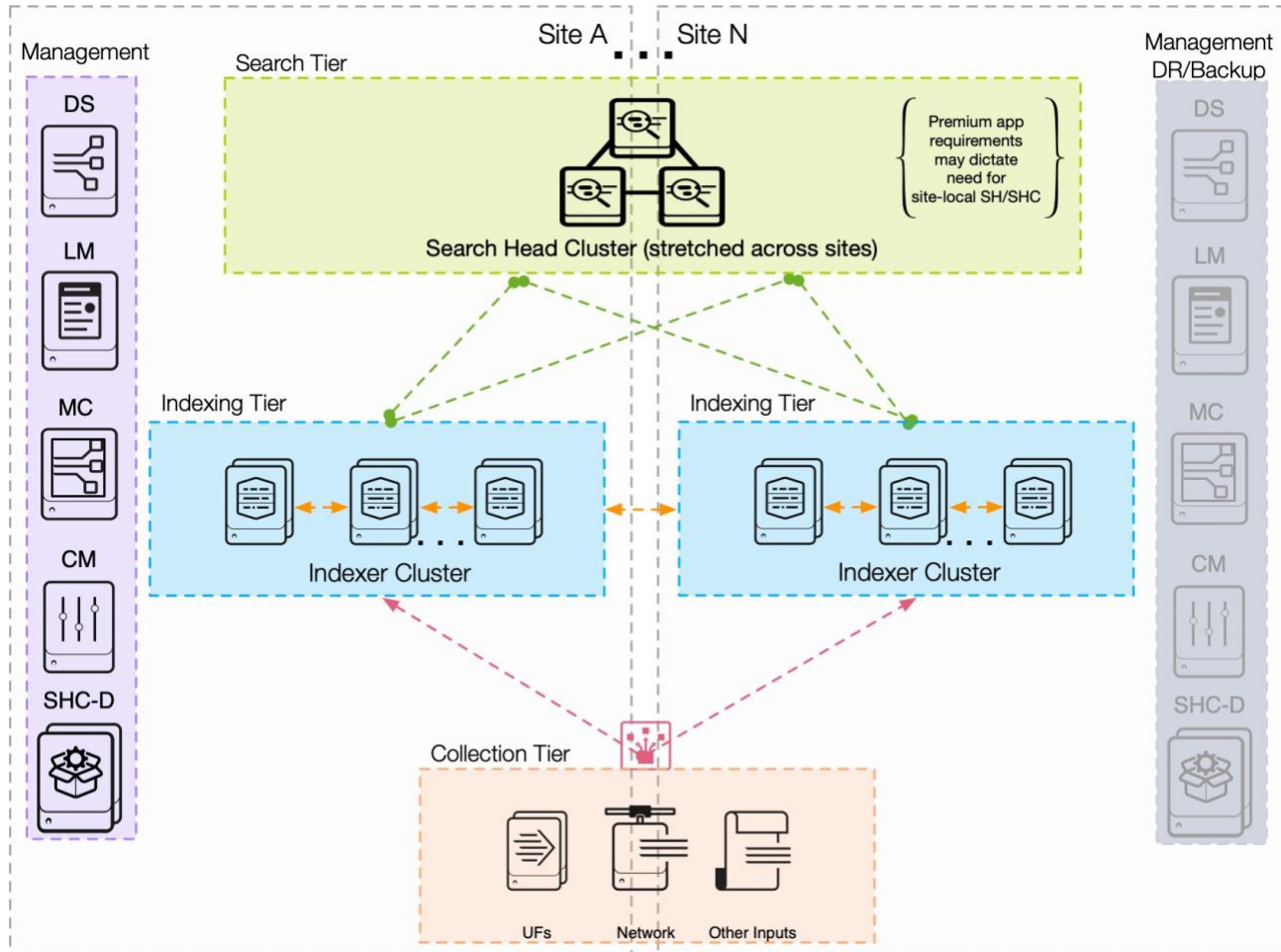
Description of Distributed Clustered Deployment - Multi-Site (M2 / M12)	Limitations
To provide near-automatic disaster recovery in case of a catastrophic event (like a data center outage), multi-site clustering is the deployment architecture of choice. A healthy multi-site cluster requires acceptable inter-site network latency as specified in the Splunk documentation. This topology allows you to deterministically replicate data to two or more groups (=sites) of indexer cluster peers by configuring the site replication and search factor. This site-replication factor allows you to specify where replica copies are being sent to and ensures data is distributed across multiple failure domains. A multi-site cluster is still managed by a single cluster manager, which has to be failed over to the DR site in case of a disaster. Multi-site clustering provides data redundancy across physically separated distributed locations, with the possibility for geographically separated distribution (subject to limits mentioned above). Available search peer (indexer) capacity across sites can be utilized for search execution in an active/active model. Site-affinity can be configured to ensure that users logged on to a specific site's search head(s) will only search local indexers. Note for ES customers: If your category code is M12 (i.e. you intend to deploy the Splunk App for Enterprise Security), a single dedicated search head is required to deploy the app (this is not pictured in the topology diagram). For the ES search head, failover involves setting up a "shadow" search head in the failover site that is only activated and used in a DR situation. Please engage Splunk Professional Services to design and implement a site failover mechanism for your Enterprise Security deployment.	• No sharing of available Search Head capacity and no search artifact replication across sites • Failure of Management functions need to be handled outside of Splunk in case of site failure • Cross-site latency for index replication must be within recommended limits

Distributed Clustered Deployment + SHC - Multi-Site (M3 / M13)



Description of Distributed Clustered Deployment + SHC - Multi-Site (M3 / M13)	Limitations
This topology utilizes a search head cluster to add horizontal scalability and removes the single point of failure from the search tier in each site. A minimum of three search heads are required to implement a SHC (per site). One or more independent SHCs can be deployed to meet specific requirements, for example to run some of Splunk's premium apps that require dedicated search environments. To manage the SHC configuration, an additional Splunk component called the Search Head Cluster Deployer is required for each SHC. This component is necessary in order to deploy changes to configuration files in the cluster. The Search Head Cluster Deployer has no HA requirements (no runtime role). The SHC provides the following benefits: a) Increased available search capacity beyond what a single search head can provide b) Scheduled search workload distribution across the cluster and c) Optimal user failover in case of a search head failure A network load-balancer that supports sticky sessions is required in front of the SHC members in each site to ensure proper load balancing of users across the cluster. Note for ES customers: If your category code is M13 (i.e. you intend to deploy the Splunk App for Enterprise Security), a single dedicated search head cluster contained within a site is required to deploy the app (this is not explicitly pictured in the topology diagram). To be able to recover an ES SH environment from a site failure, 3rd party technology can be used to perform a failover of the search head instances or a "warm standby" ES SH can be provisioned and kept in synch with the primary ES environment. It is strongly recommended to engage with Splunk Professional Services when deploying ES in a HA/DR environment.	• No search artifact replication across sites, SHCs are standalone • Cross-site latency for index replication must be within documented limits • A single SHC cannot contain more than 100 nodes

Distributed Clustered Deployment + SHC - Multi-Site (M4 / M14)



Description of Distributed Clustered Deployment + SHC - Multi-Site (M4 / M14)	Limitations
This is the most complex validated architecture, designed for deployments that have strict requirements around high-availability and disaster recovery. We strongly recommend involving Splunk Professional Services to implement this kind of deployment. When properly deployed, this topology provides continuous operation of your Splunk infrastructure for data collection, indexing and search. This topology involves implementation of one or more "stretched" search head clusters that span two or more sites. This provides optimal failover for users in case of a search node or data center failure. Search artifacts and other runtime knowledge objects are replicated in the SHC. Careful configuration is required to ensure that replication will happen across sites, as the SHC itself is not site-aware (i.e. artifact replication is non-deterministic). Site-affinity can be configured to ensure the WAN link between sites is utilized only in cases when a search cannot be satisfied locally. A network load-balancer that supports sticky sessions is required in front of the SHC members to ensure proper load balancing of users across the cluster. Note for ES customers: If your category code is M14 (i.e. you intend to deploy the Splunk App for Enterprise Security), a single dedicated search head cluster contained within a site is required to deploy the app (this is not explicitly pictured in the topology diagram). ES requires a consistent set of runtime artifacts to be available and this cannot be guaranteed in a stretched SHC when a site outage occurs. To be able to recover an ES SH environment from a site failure, 3rd party technology can be used to perform a failover of the search head instances or a "warm standby" ES SH can be provisioned and kept in synch with the primary ES environment. It is strongly recommended to engage with Splunk Professional Services when deploying ES in a HA/DR environment.	• Network latency across sites must be within documented limits • Failover of the SHC may require manual steps if only a minority of cluster members survive

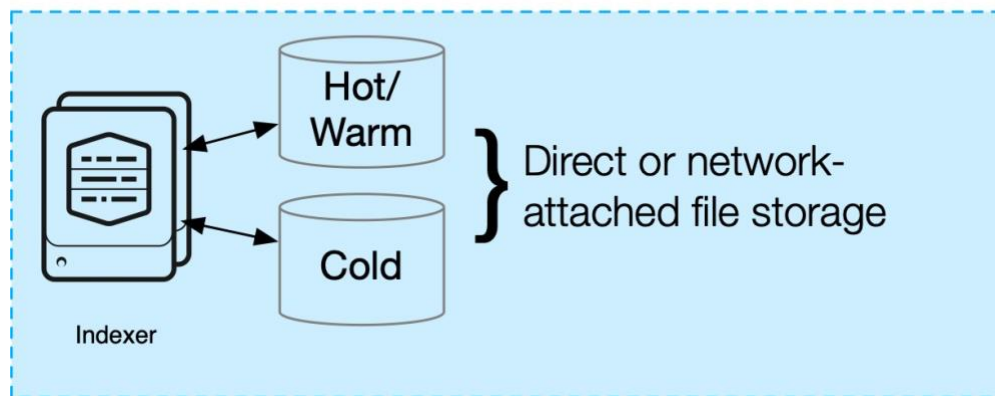
Splunk Indexer Architecture Options

Splunk Indexers can be configured using two main architecture options. Each option has its benefits and drawbacks. Choosing which storage option to utilize requires understanding of the implications. Changing the architecture later is not straightforward and requires a migration process that involves both infrastructure changes as well as data movement. This chapter should help you in understanding those implications and making the right choice to meet your specific requirements. **Note** that the indexer architecture for Splunk Cloud is chosen by Splunk.

Classic Indexer Architecture Using File System Storage

In the classic, tried-and-true deployment model, indexers store data across the whole data lifecycle on a server-accessible file system. This can be a direct-attached storage (DAS) filesystem only, or a combination of DAS storage and network-attached file storage:

Classic Indexer Storage Architecture



Note: Please refer to the Splunk documentation for information on supported storage and file system types.

This architecture tightly couples indexer compute and storage, is able to provide a consistent search performance profile across all data at rest and has few external dependencies. While scaling out is relatively straightforward in this model, scaling down is not easily possible and requires potentially time-consuming procedures. In clustered indexer topologies, Splunk maintains multiple copies of the data across the configured retention period. This requires a potentially significant amount of storage, especially when requirements for long-term data retention exist, and increases the TCO for the solution accordingly. This architecture is recommended when you have requirements for either

- Short-term data retention (≤ 3 months) or
- Long-term retention and performance-critical search use cases that frequently access older historic data

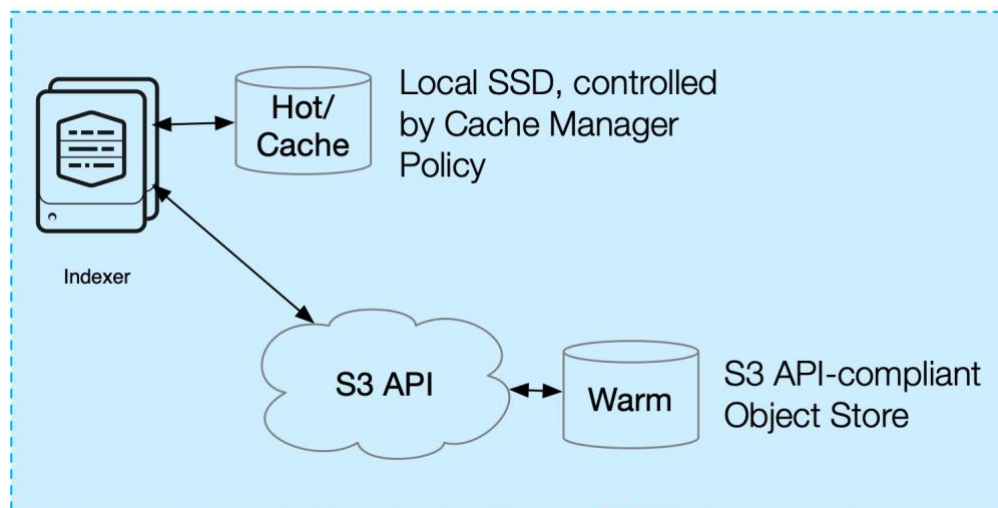
SmartStore Indexer Architecture Using Object Storage

Splunk SmartStore Architecture was created primarily to provide a solution for decoupling of compute and storage on the indexing tier to enable a more elastic indexing tier deployment.

Elasticity affords you the opportunity to add compute as needed, for example to satisfy higher search workload demands, without also having to add storage. Similarly, increasing the amount of storage to extend your data retention interval does not incur any additional infrastructure cost for compute.

This model can have significant positive impact on the TCO of your Splunk deployment, especially when you retain data for long periods of time. SmartStore utilizes a fast, SSD-based cache on each indexer node to keep recent data locally available for search. When data rolls to WARM lifecycle stage (see Splunk bucket lifecycle), it is uploaded to an S3 API-compliant object store for persistence, but remains in local cache until it is evicted as needed based on cache manager policy:

SmartStore Indexer Storage Architecture



All data reliability responsibility is transferred to the object store to ensure lossless storage. That means that redundant bucket copies created in the classic architecture do no longer need to be created by Splunk in the object store, leading to the substantial storage savings for clustered deployments mentioned above.

Note: You can find a more detailed architecture diagram for SmartStore [here](#).

Splunk's own analysis of customer search profiles has shown that 95%+ of all searches are run over time periods 7 days or less. If your use cases fall into the same category, SmartStore can be a great choice to provide you more flexibility with scaling compute and storage at a lower TCO. For environments with small total data volume and short data retention, or environments where elasticity is not required, it is typically more economical to use the classic architecture.

To utilize the SmartStore indexer architecture, you will need an S3 API-compliant object store. This is readily available in AWS, but may not be available in other cloud and on-prem environments. This object store needs to provide the same or better availability as your indexing tier.

For details on how to implement SmartStore, please refer to the documentation [here](#). For a list of current SmartStore restrictions, please refer to the documentation [here](#).

Important Note: If you intend to deploy SmartStore on-prem with an S3-compliant object store in a multi-site deployment spanning data centers, please make sure you understand the specific requirements for the object store documented [here](#).

Data Collection Architecture

The data collection tier is a core component of a Splunk deployment. It enables any device in your environment to forward data to the indexing tier for processing, thereby making it available for search in Splunk. The most important factor here is to ensure that forwarding and indexing happen in the most efficient and reliable way, as this is critical to the success and performance of your Splunk deployment.

Consider the following aspects for your data collection tier architecture:

- The origin of your data. Does it come from log files, syslog sources, network inputs, OS event logging facilities, applications, message bus or elsewhere?
- Requirements for data ingest latency and throughput
- Ideal event distribution across the indexers in your indexing tier
- Fault tolerance and automatic recovery (HA)
- Security and data sovereignty requirements

This section of SVAs focuses on the common data collection methods. This section also discusses architecture and best practices for each data collection method and calls out potential issues to consider when making your implementation choice.

Important Architectural Considerations and Why They Matter

Given the essential role of the data collection tier, it's important to understand the key considerations involved in designing the architecture.

While some of these considerations may or may not be relevant to you based on your requirements, the considerations in bold text in the table below describe fundamental items that are relevant for every environment.

Consideration	Why is this important?
Data is ingested properly (timestamps, line breaking, truncation)	If data is not ingested properly because event timestamps and line breaking are not correctly configured, searching this data will become very difficult. This is because event boundaries have to be enforced at search time. Incorrect or missing timestamp extraction configurations can cause unwanted implicit timestamp assignment. This will confuse your users and make getting value out of your data much more difficult than it needs to be.
Data is optimally distributed across available indexers	The importance of ideal event distribution across indexers cannot be overstated. The indexing tier works most efficiently when all available indexers are equally utilized. This is true for both data ingest as well as search performance. A single indexer that handles significantly more data ingest compared to peers can negatively impact search response times. For indexers with limited local disk storage, uneven event distribution may also cause data to be prematurely aged out before meeting the configured data retention policy.
All data reaches the indexing tier reliably and without loss	Any log data that is collected for the purpose of reliable analytics needs to be complete and valid, such that searches performed on the data provide valid and accurate results.
All data reaches the indexing tier with minimum latency	Delays in data ingest will increase the time between a potentially critical event occurring and the ability to search for and react to it. Minimal ingest latency is often crucial for monitoring use cases that trigger alerts to staff or incur automated action.

Consideration	Why is this important?
Data is secured while in transit	If the data is either sensitive or has to be protected while being sent over non-trusted networks, encryption of data may be required to prevent unauthorized third-party interception. Generally, we recommend all connections between Splunk components to be SSL enabled.
Network resource use is minimized	The network resource impact of log data collection must be minimized so as not to impact other business critical network traffic. For leased-line networks, minimizing network utilization also contributes to a lower TCO of your deployment.
Authenticate/authorize data sources	To prevent rogue data sources from affecting your indexing environment, consider implementing connection authentication/authorization. This may be covered by using network controls, or by employing application-level mechanisms (e.g., SSL/TLS).

Because of its vital role for your deployment, the guidance in this document focuses on architectures that support ideal event distribution. When a Splunk environment does not provide expected search performance, it is in almost all cases either caused by not meeting minimum storage performance requirements and/or uneven event distribution that limits exploiting search parallelization.

Before we take a look at the recommended data collection architecture components we identified during your use of the iSVA tool, let's share some general guidance for the data collection tier.

General Data Collection Architecture Guidance

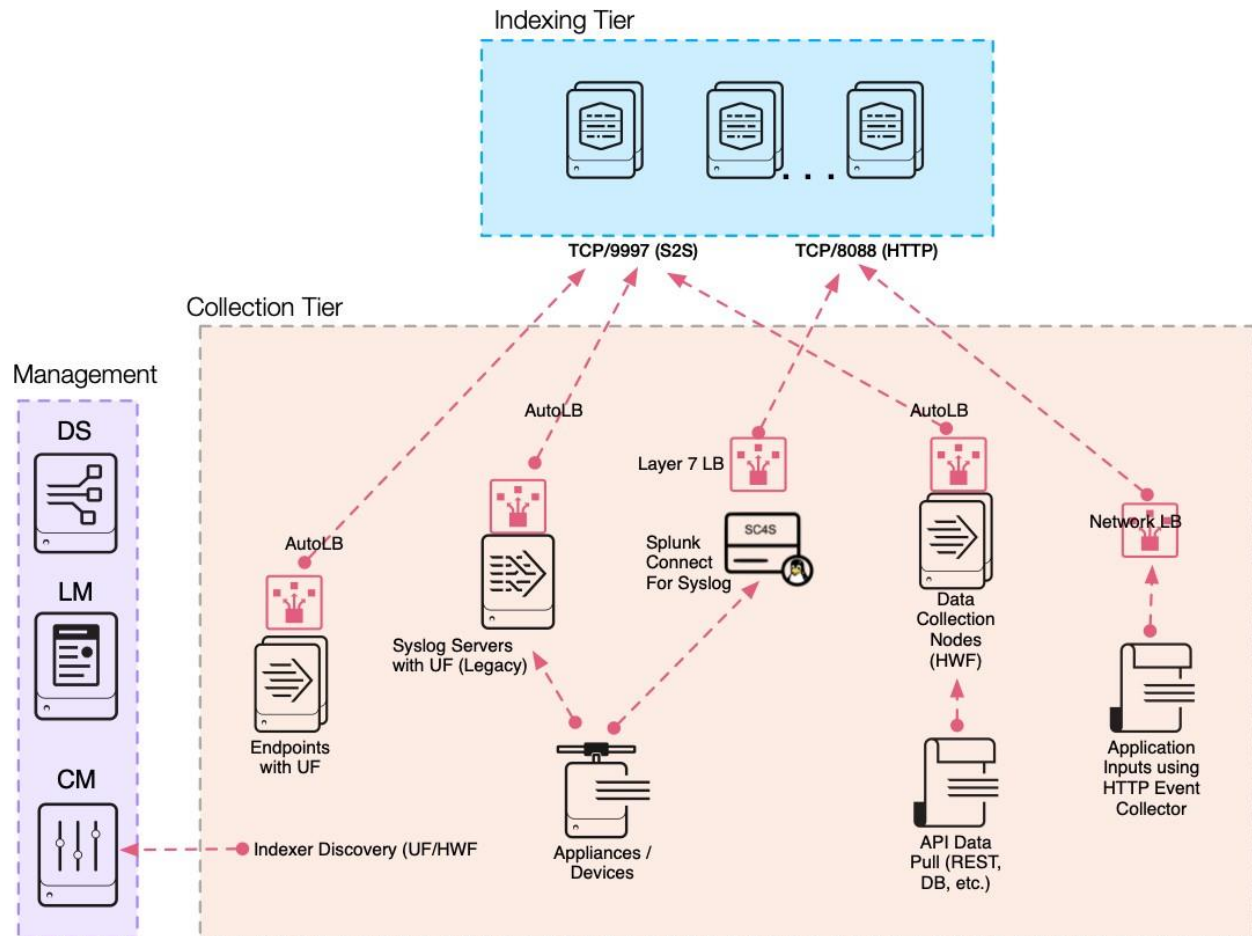
Ideally, the data collection tier is as "flat" as possible. This means that data sources are collected locally by a universal forwarder and forwarded directly to the indexing tier. This is a best practice because it ensures minimal data ingest latency (time to search) and enables proper event distribution across available indexers. Following this best practice leads to ease of management and operational simplicity. We often see customers deploy an intermediary forwarding tier. Generally speaking, avoid this unless requirements cannot be met otherwise. Due to the potential impact of intermediary forwarders, this document contains a separate section on this topic with more details.

There are endpoints that do not allow installation of the universal forwarder (network devices, appliances, etc.) and log using the syslog protocol. A separate best practice architecture to collect such data sources is outlined in the section titled Syslog Data Collection.

For data sources that have to be collected using programmatic means (e.g. via APIs, database access), deploying a data collection node (DCN) based on a full Splunk enterprise install is recommended. This is also known as a heavy forwarder. It is not recommended that you run these kinds of inputs on the search head tier in anything other than a development environment.

The following diagram shows a general data collection architecture that reflects this guidance.

Data Collection Topology Overview



The diagram above shows the Deployment Server (DS) in the management tier, which is used to manage the configurations on data collection components. Also, the License Manager (LM) is shown here since data collection nodes require access to the LM to enable Splunk Enterprise features. The Cluster Manager (CM), if available, can be used by forwarders for indexer discovery, removing the need to manage available indexes in the forwarder output configuration.

In the above diagram, AutoLB represents the Splunk built-in auto-load balancing mechanism. This mechanism is used to ensure proper event distribution for data sent using the Splunk proprietary S2S protocol (default port 9997). Note: Using an external network load-balancer for S2S traffic is currently not supported and not recommended.

To load-balance traffic from data sources that communicate with an industry-standard protocol (like HTTP or syslog), a Layer 7 network load balancer is used to ensure even load and event distribution across indexers in the indexing tier.

Data Collection Components

The data collection tier is arguably the most critical part of your Splunk deployment topology. Splunk supports a wide range of data collection mechanisms to help you ingest data into Splunk easily, such that it can be indexed and made available to search.

This section of the document provides you with guidance on recommended data collection architecture component deployment for various collection needs.

(UF) Universal Forwarder

The universal forwarder (UF) is the best choice for a large set of data collection requirements from systems in your environment. It is a purpose-built data collection mechanism with minimal resource requirements. The UF should be the default choice for collecting and forwarding log data.

The UF provides:

- Checkpoint/restart function for lossless data collection
- Efficient protocol that minimizes network bandwidth utilization
- Throttling capabilities
- Built-in, load-balancing across available indexers
- Optional network encryption using SSL/TLS
- Data compression (use only without SSL/TLS)
- Multiple input methods (files, Windows Event logs, network inputs, scripted inputs)
- Limited event filtering capabilities (Windows event logs only)
- Parallel ingestion pipeline support to increase throughput/reduce latency

With few exceptions for well-structured data (json, csv, tsv), the UF does not parse log sources into events, so it cannot perform any action that requires understanding of the format of the logs. It also ships with a stripped down version of Python, which makes it incompatible with any modular input apps that require a full Splunk stack to function.

It is normal for a large number of UFs (100s to 10,000s) to be deployed on endpoints and servers in a Splunk environment. The configuration of these UF deployments can be centrally managed, either with the Splunk deployment server or a third-party configuration management tool (like e.g. Puppet or Chef).

(HF) Heavy Forwarder

The heavy forwarder (HF) is a full Splunk Enterprise deployment configured to act as a forwarder with indexing disabled. A HF generally performs no other Splunk roles. The key difference between a UF and a HF is that the HF contains the full parsing pipeline, performing the identical functions an indexer performs, without actually writing and indexing events on disk. This enables the HF to understand and act on individual events, for example to mask data or to perform filtering and routing based on event data. Since it is a full Splunk Enterprise install, it can host modular inputs that require a full Python stack to function properly for data collection or serve as an endpoint for the Splunk HTTP event collector (HEC).

The HF:

- Parses data into events
- Filters and routes based on individual event data
- Has a larger resource footprint than the UF
- Has a larger network bandwidth footprint than the UF (up to 5x)
- Provides a GUI for management

In general, HFs are not installed on endpoints for the purpose of data collection. Instead, they are used on standalone systems to implement data collection nodes (DCN) or intermediary forwarding tiers. **Use a HF only when requirements to collect data from other systems cannot be met with a UF.**

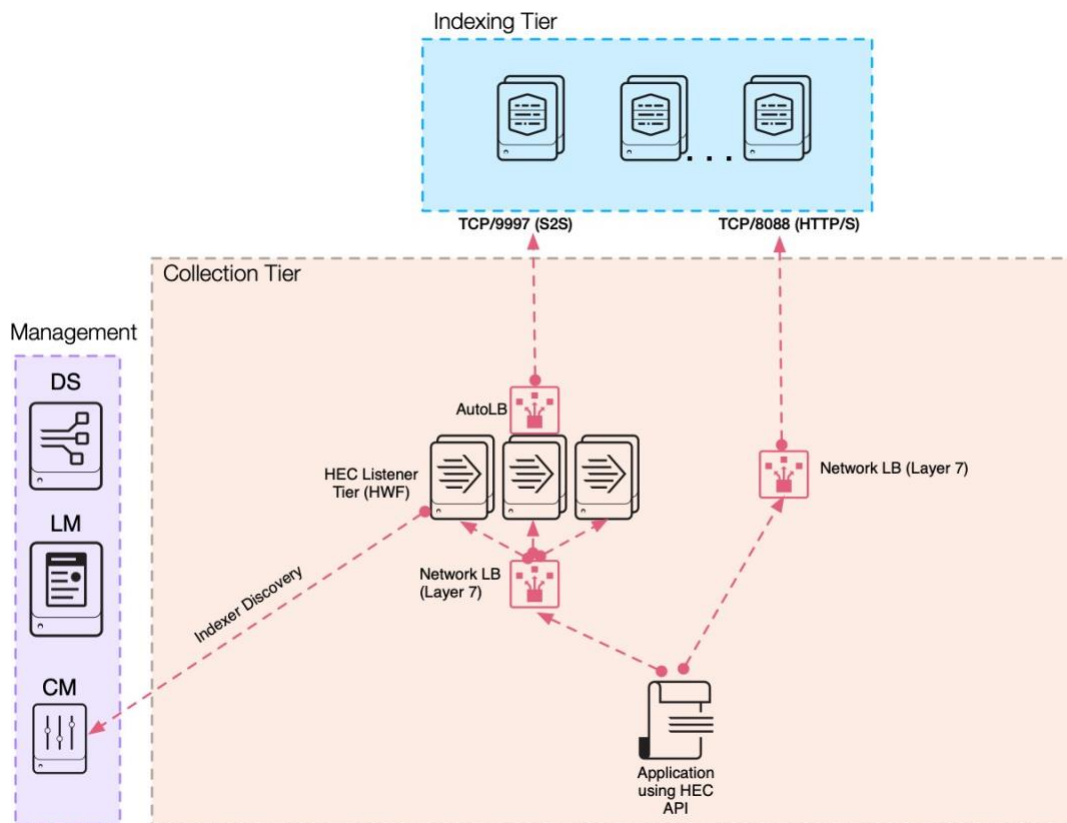
Examples of such requirements include:

- Reading data from RDBMS for the purpose of ingesting it into Splunk (database inputs)
- Collecting data from systems that are only reachable via an API (cloud services, VMWare monitoring, proprietary systems, etc.)
- Providing a dedicated tier to host the HTTP event collector service (HEC)
- Implementing an intermediary forwarding tier that requires a parsing forwarder for routing/filtering/masking

(HEC) HTTP Event Collector

The HEC provides a listener service that accepts HTTP/S connections on the server side, and an API on the client side, allowing applications to post log data payloads directly to either the indexing tier or a dedicated HEC receiver tier consisting of one or more heavy forwarders. HEC provides two endpoints that support data to be sent either in raw format or in JSON format. Utilizing JSON can allow for additional metadata to be included in the event payload that may facilitate greater flexibility when searching the data later.

The following diagram illustrates the two deployment options for HEC:



The management tier contains the license manager (required by HF) as well as the deployment server to manage the HTTP inputs on the listening components. Note: If the indexing tier is clustered and receives HEC traffic directly, HEC configuration is managed via the cluster manager instead of the deployment server.

The decision for which deployment topology you choose depends largely on your specific workload. For example, if you need to provide connectivity for a very large number of HTTP clients, a dedicated listener tier that can provide the required system resources may be needed to not starve indexers for those resources. On the other hand, if a large volume of data is coming in from a relatively small number of producers, ingesting this traffic directly on indexers is preferable. Also, consider that a dedicated HEC listener tier introduces another architectural component into your deployment, so avoid it unless your requirements dictate otherwise. On the positive side, it can be scaled independently and provides a level of isolation from the indexing tier from a management perspective. Also, since the dedicated HEC tier requires a HF, it will parse all inbound traffic, removing some or all of that workload off of the indexers.

On the other hand, hosting the HEC listener directly on the indexers will likely ensure better event distribution across the indexing tier, because HTTP is a well-understood protocol for all network load balancers and the appropriate load balancing policy can help ensure that incoming data is evenly spread across available indexers.

In the spirit of deploying the simplest possible architecture that meets your requirements, we recommend you consider hosting your HEC listener on the indexers, assuming you have sufficient system capacity to do so (see above). This decision can easily be reverted later if the need arises, simply by deploying an appropriately sized and configured HF tier and changing the LB configuration to use the HF's IP addresses instead of the indexers'. That change should be transparent to client applications.

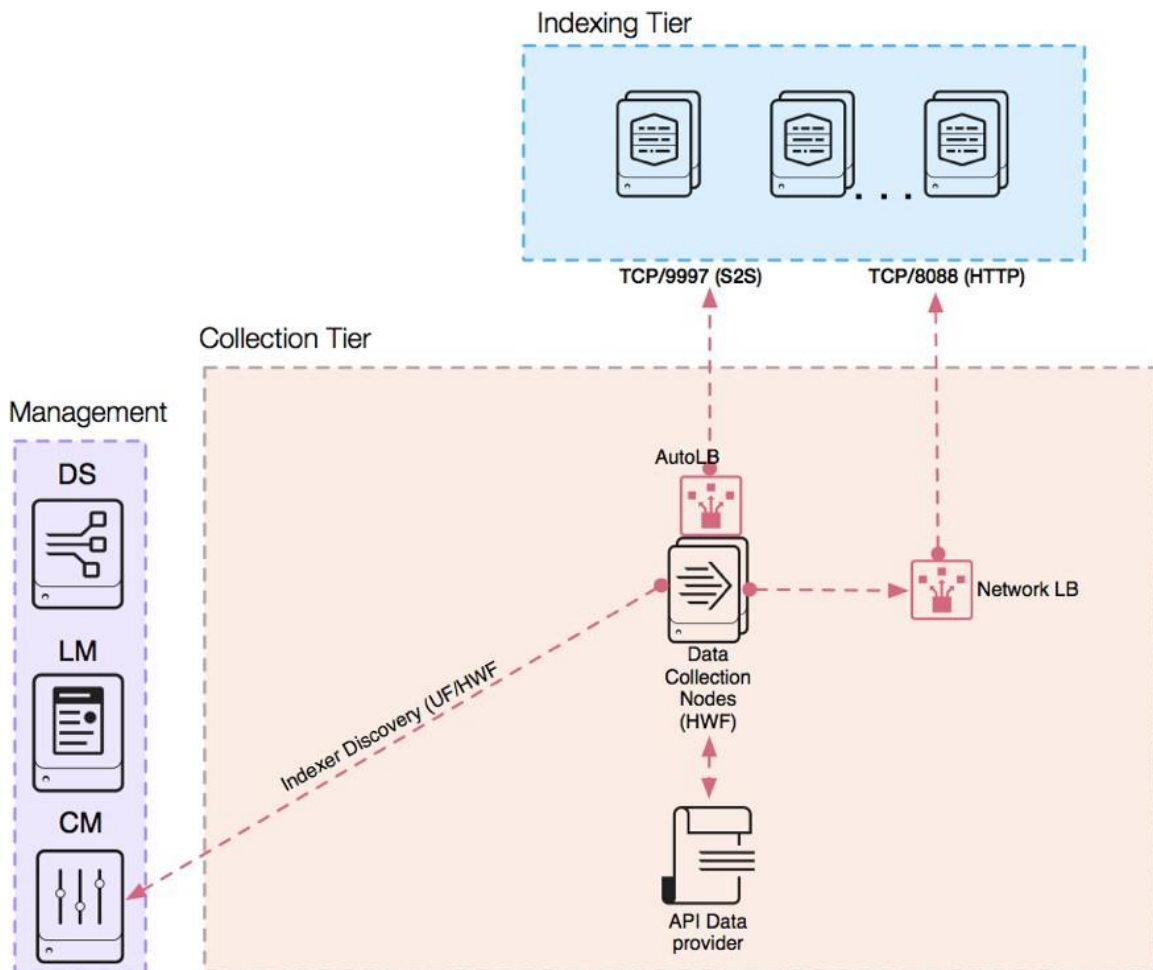
Note: If you do require indexer acknowledgment for data sent via HEC, a dedicated HEC listener tier is recommended to minimize duplicate messages caused by rolling indexer restarts.

Note: This HEC deployment architecture is also used for providing the transport for some of the other data collection components discussed later, specifically Syslog and metrics data collection.

(DCN) Heavy Forwarder as Data Collection Node

Some data sources require collection by using some sort of an API. These APIs can include REST, web services, JMS and/or JDBC as the query mechanism. Splunk as well as third-party developers provide a wide variety of applications that allow these API interactions to occur. Most commonly, these applications are implemented using the Splunk Modular Input framework, which requires a full Splunk enterprise software install to properly function. The best practice to realize this use case is to deploy one or more servers to work as a heavy forwarder configured to work as a Data Collection Node (DCN).

Data Collection Node Topology

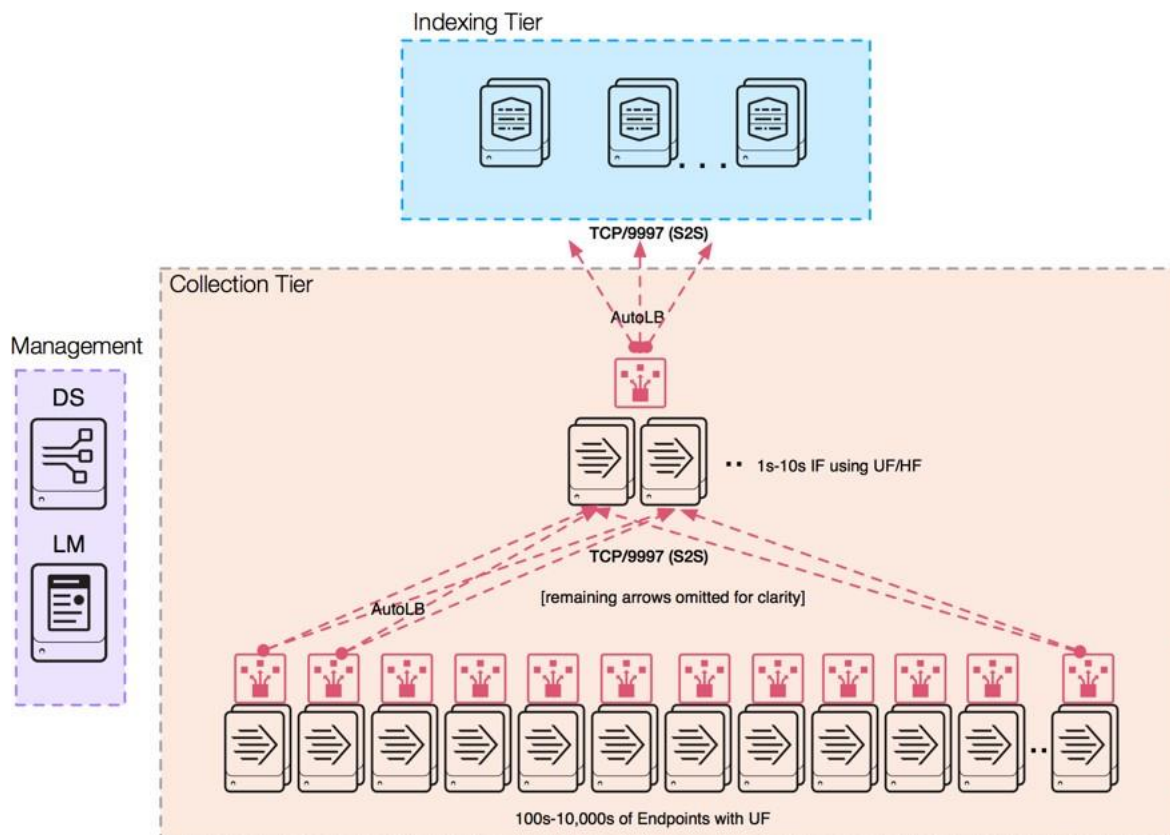


(IF) Intermediary Forwarding Tier

In some situations, intermediary forwarders are needed for data forwarding. Intermediary forwarders receive log streams from endpoints and forward it on to an indexer tier. Intermediary forwarders introduce architectural challenges that require careful design in order to avoid negative impacts to the overall Splunk environment. Most prominently, intermediary forwarders concentrate connections from 100s to 10,000s of endpoint forwarders and forward to indexers using a far smaller number of connections. This can materially affect the data distribution across the indexing tier, because only a subset of indexers is receiving traffic at any given point in time. However, these negative side effects can be mitigated by proper sizing and configuration.

The following diagram illustrates this challenge well:

Intermediary Forwarding Topology



In a scenario with a single intermediary forwarder, all endpoints connect to this single forwarder (potentially thousands), and the intermediary forwarder in turn only connects to one indexer at any given time. This is not an optimal scenario because the following consequences are likely to occur:

- A large data stream from many endpoints is funneled through a single pipe that exhausts your system and network resources
- Limited failover targets for the endpoints in case of IF failure (your outage risk is inversely proportional to the number of IFs)
- A small number of indexers are served at any given point in time. Searches over short time periods will not benefit from parallelization as much as they could otherwise

Intermediary forwarders also add an additional architecture tier to your deployment which can complicate management and troubleshooting and adds latency to your data ingest path. Try to avoid using intermediary forwarding tiers unless this is the only option to meet your requirements. You may consider using an intermediary tier if you have:

- Sensitive data that needs to be obfuscated/removed before sending across the network to indexers. An example is when you must use a public network
- Strict security policies that do not allow for direct connections between endpoints and indexers such as multi-zone networks or cloud-based indexers
- Bandwidth constraints between endpoints and indexers requiring a significant subset of events to be filtered
- Requirements to perform event-based routing to dynamic targets

Consider sizing and configuration needs for any intermediary forwarding tier to ensure availability of this tier, provide sufficient processing capacity to handle all traffic and support good event distribution across indexers. The IF tier has the following requirements:

Sufficient number of data processing pipelines overall Redundant IF infrastructure

Properly tuned Splunk load-balancing configuration. For example, `autoLBVolume`, `EVENT_BREAKER`, `EVENT_BREAKER_ENABLE`, possibly `forceTimeBasedAutoLB` as needed

The general guideline suggests having twice as many IF processing pipelines as indexers in the indexing tier.

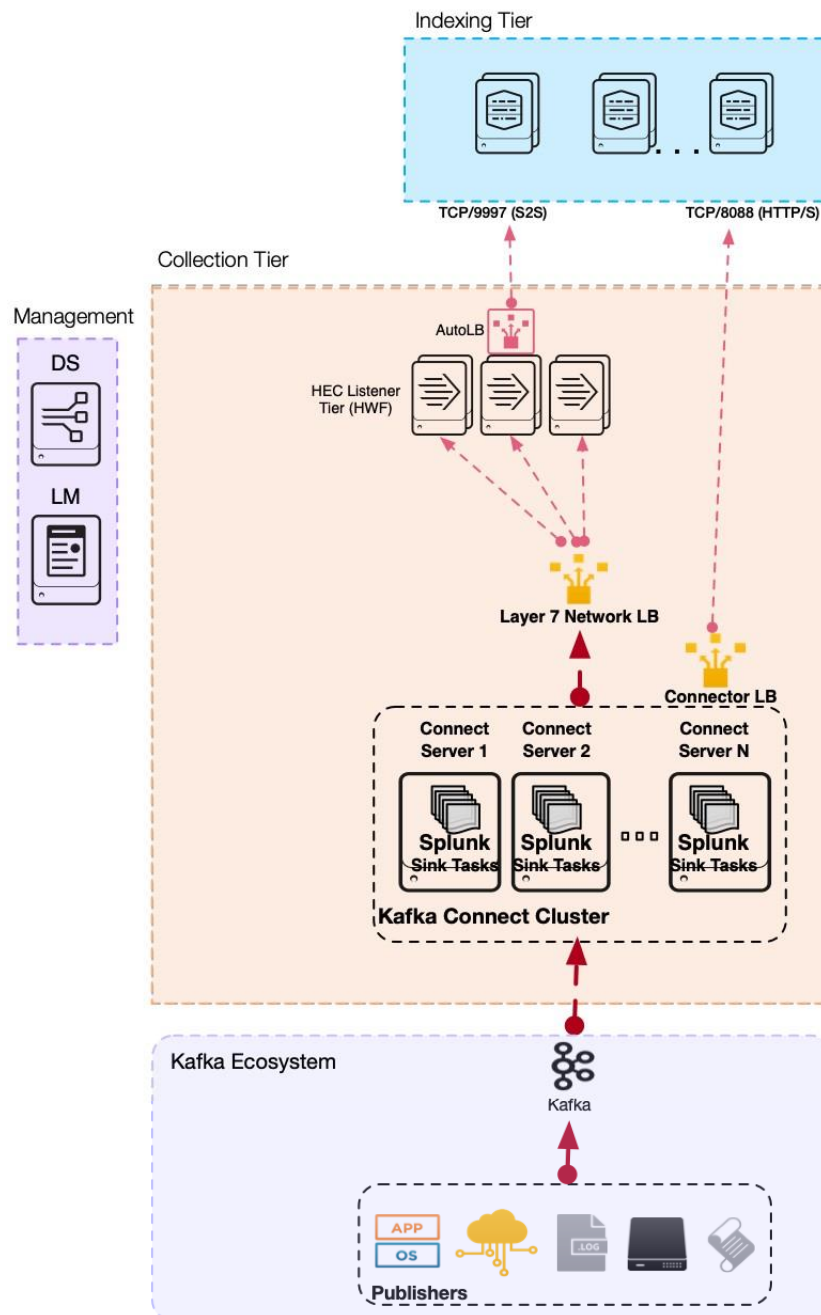
Note: A processing pipeline does not equate to a physical IF server. Provided sufficient system resources (CPU cores, memory and NIC bandwidth) are available, a single IF can be configured with multiple processing pipelines.

If you need an IF tier (see questionnaire), default to using UF for the tier since they provide higher throughput at a lower resource footprint for both the system and network. Use HF if the UF capabilities do not meet your requirements.

(KAFKA) Consuming Log Data From Kafka Topics

Splunk provides a supported sink connector for consuming data from Kafka topics called "Splunk Connect for Kafka." See [Apache Kafka Connect](#) in the Splunk Connect for Kafka Manual for detailed product documentation. The Splunk Connect for Kafka package is installed into a properly sized Kafka Connect cluster (outside of Splunk), where it can subscribe to topics as configured and send consumed events through the HEC to be indexed:

Data Collection Topology Using Kafka and HEC

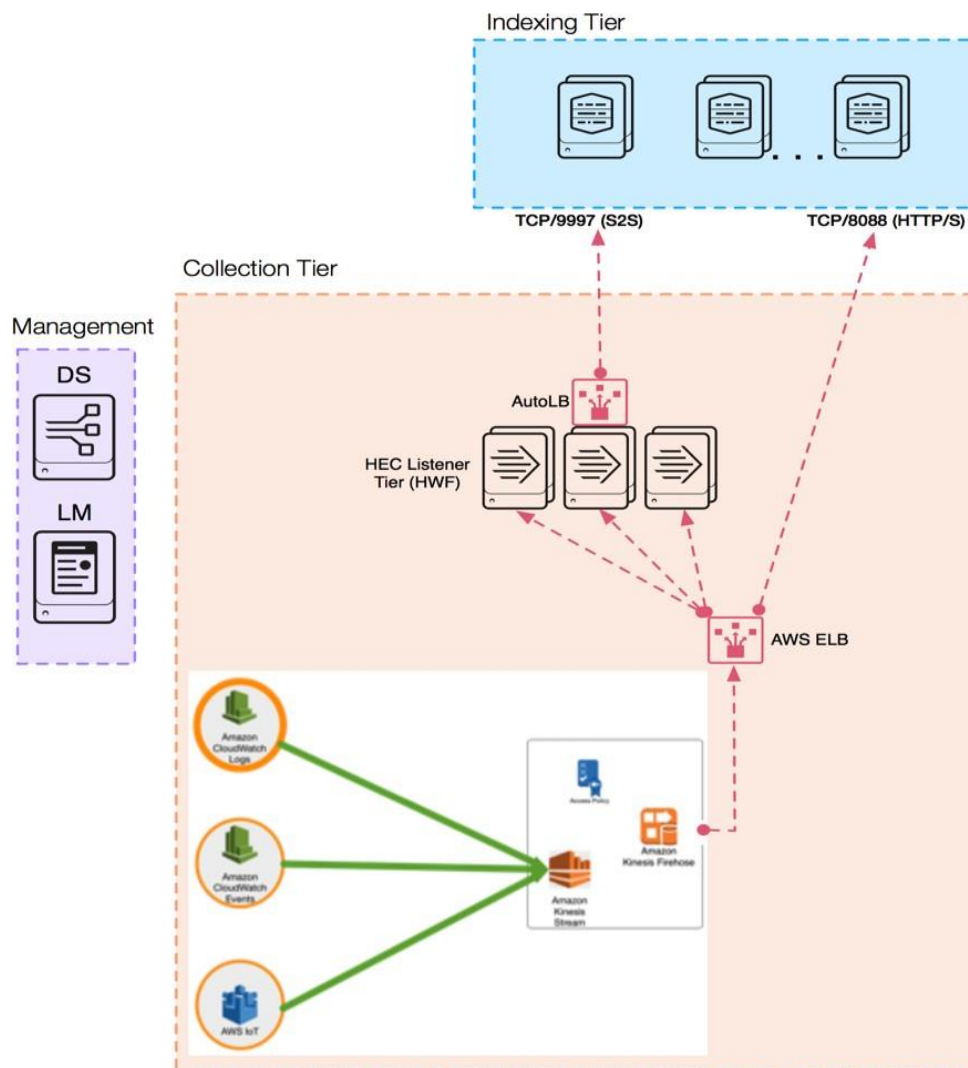


The diagram shows Kafka Publishers sending messages to the Kafka bus. The tasks hosted in the Kafka Connect cluster consume those messages via the Splunk Connect for Kafka and send the data to the HEC listening service using a network load balancer. Again, the HEC listening service can be either hosted directly on the indexers, or on a dedicated HEC listener tier. Please refer to the HEC section for details. Management tier components are only required if a dedicated HF tier is deployed to host HEC listeners.

(KINESIS) Consuming Log Data From Amazon Kinesis Firehose

Splunk and Amazon have implemented an integration between Kinesis and the Splunk HEC that enables you to stream data from AWS directly to a HEC endpoint, configurable via your AWS console. This is complemented by the [Splunk Add-On for Kinesis Firehose](#) which provides CIM-compliant knowledge for various data sources originating in AWS.

Data Collection Topology Using Amazon Kinesis



The diagram shows AWS log sources being sent using a Kinesis stream to the Firehose, which with proper configuration will send the data to the HEC listening service via an AWS ELB. Again, the HEC listening service can be either hosted directly on the indexers, or on a dedicated HEC listener tier. Please refer to the HEC section for details.

Management tier components shown are only required if a dedicated HF tier is deployed to host HEC listeners.

(METRICS) Metrics Collection

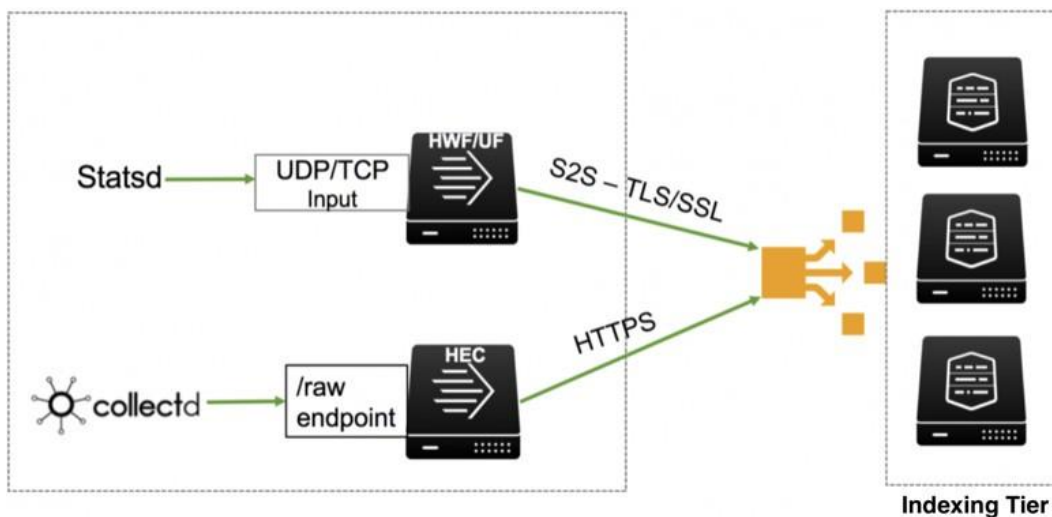
Splunk has the capability to receive and collect system and application performance data, or metrics data, from a variety of 3rd party software. Metrics in the Splunk platform use a custom index type that is optimized for metrics storage and retrieval.

There are different ways to consume metrics data and the collection method is based upon the technology used. The most common form of metrics collections comes in the form of a software daemon, such as **collectd**, **statsd** or using a customized metrics data file and valid configuration for the data source.

There are primarily two methods for getting metrics into Splunk when using agents such **statsd** and **collectd**. Either using a **Direct TCP/UDP** input or via the **HEC**.

Using **HEC** is considered a best practice due to the resiliency and scalability of the **HEC** endpoint, and the ability to horizontally scale the collection tier easily.

Metrics Data Collection Topology



Statsd currently supports **UDP** and **TCP** transport, which you can use as a direct input on a Splunk Forwarder, or Indexer. However, it is not a best practice to send TCP/UDP traffic directly to forwarders in production as the architecture is not resilient and prone to event loss (see Syslog collection) caused by required Splunk forwarder restarts.

(SYSLOG) Syslog Data Collection

The syslog protocol delivers a ubiquitous source for log data in the enterprise. Most scalable and reliable data collection tiers contain a syslog ingestion component. There are multiple ways to get syslog data into Splunk:

- **Splunk Connect for Syslog (SC4S):** This is the current best practice recommendation to collect syslog data. It provides a Splunk-supported turn-key solution and utilizes the HTTP Event Collector to send data to Splunk for indexing. It scales well and addresses the shortcomings of other methods.
- **Universal forwarder (UF)/heavy forwarder (HF):** Use a Splunk UF or HF to monitor (ingest) files written out by a syslog server (such as rsyslog or syslog-ng). While still widely in use, we no longer recommend this as a best-practice approach in favor of SC4S.
- **Direct TCP/UDP Input:** Splunk has the ability to listen on a TCP or UDP port (default port is UDP 514) and accept syslog traffic directly. While this is acceptable for lab and test environments, Splunk **strongly discourages** this practice in any production environment.

Splunk Connect for Syslog (SC4S - recommended best practice)

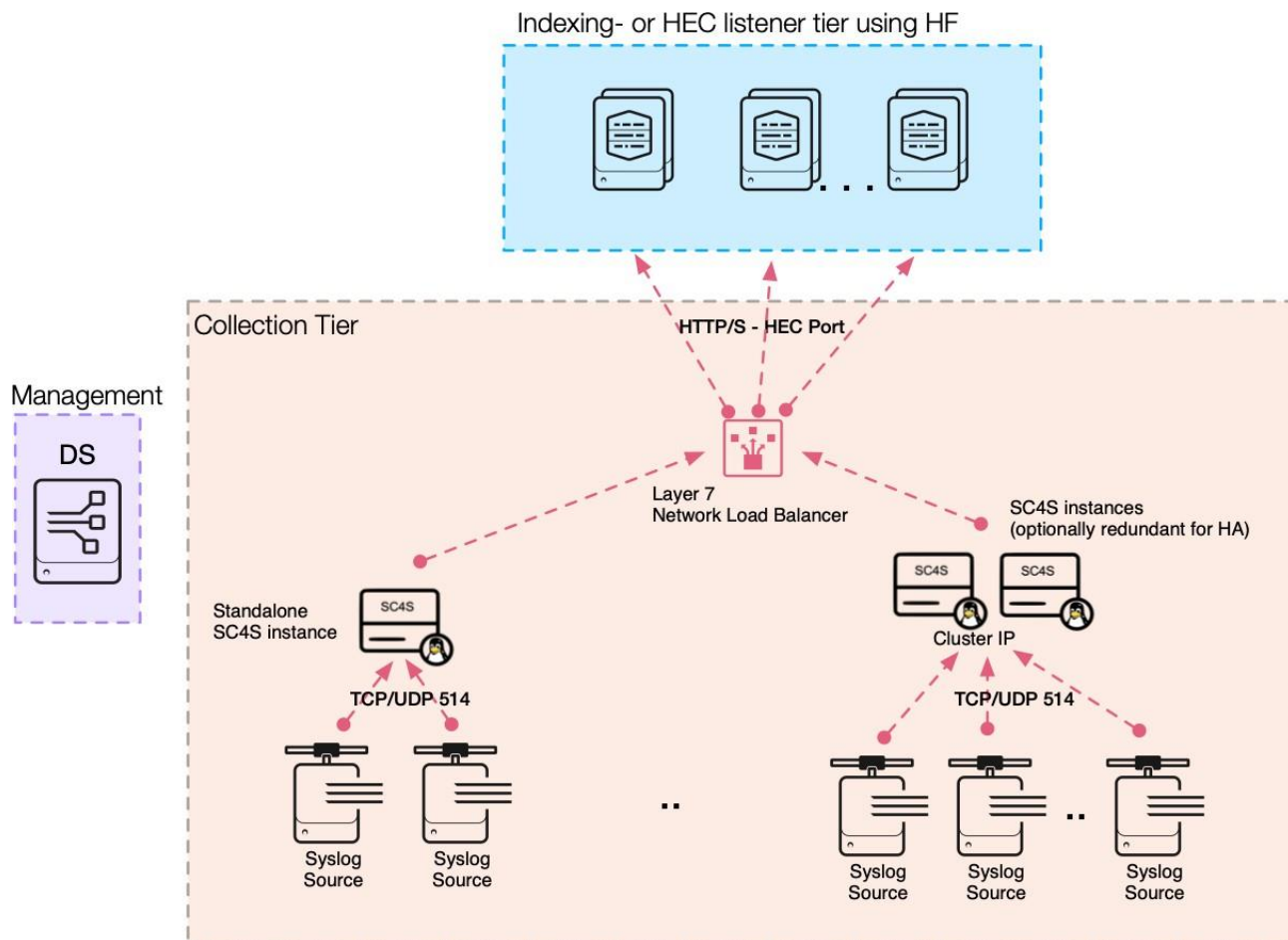
Many of the most common data sources that power Splunk product use cases require a syslog server for data collection. Highly specific expertise is required to successfully design, deploy and configure a syslog server to properly work with Splunk at scale. Additionally, the Universal Forwarder or Heavy Forwarder approach to syslog collection has several issues with scale and complexity. Some customers send syslog events directly to Splunk to avoid architecting a syslog server, which introduces further problems. To help customers address these issues, Splunk developed Splunk Connect for Syslog (SC4S).

Splunk Connect for Syslog is offered in two flavors: An OCI-compliant container for ease of deployment and OOTB functionality, and a "Bring Your Own Environment" (BYOE) option for customers that are prevented from utilizing the containerized option for some reason. Both options are based on Syslog-ng server software and encapsulate the same set of configurations, while differing on the details of how Syslog-ng itself is instantiated. Splunk strongly recommends the SC4S containerized solution for all customers able to adopt it.

The benefits of Splunk Connect for Syslog include the following:

- Repeatable, concise and prescriptive Splunk solution for syslog
- Removal of the UF reduces configuration and management effort
- Reduce/remove need for add-on installation and management on indexers
- Turnkey deployment via the SC4S container architecture
- Custom deployment via a "Bring Your Own Environment" distribution
- A library of data source filters we will be continually extending with the help of the community
- Exceptionally even event distribution across the Splunk Indexers (benefits search performance, storage utilization and data retention)
- Enhanced data enrichment beyond the standard Splunk metadata of timestamp, host, source and sourcetype
- Easily add custom filters for additional sourcetypes beyond those supported out of the box
- Complete deployment documentation available [here](#)

The diagram below shows syslog sources sending data to the SC4S instance that is physically closer to them. Events from a source for which SC4S has a 'filter' log path configuration will be identified, parsed and formatted into JSON before being streamed to Splunk via HTTP/HTTPS. Best practice is to implement a network load balancer between SC4S and the Indexers to achieve the best data distribution possible. The events hitting the Indexer are already prepared so there is no need for an Add-On on the Indexer in most cases.



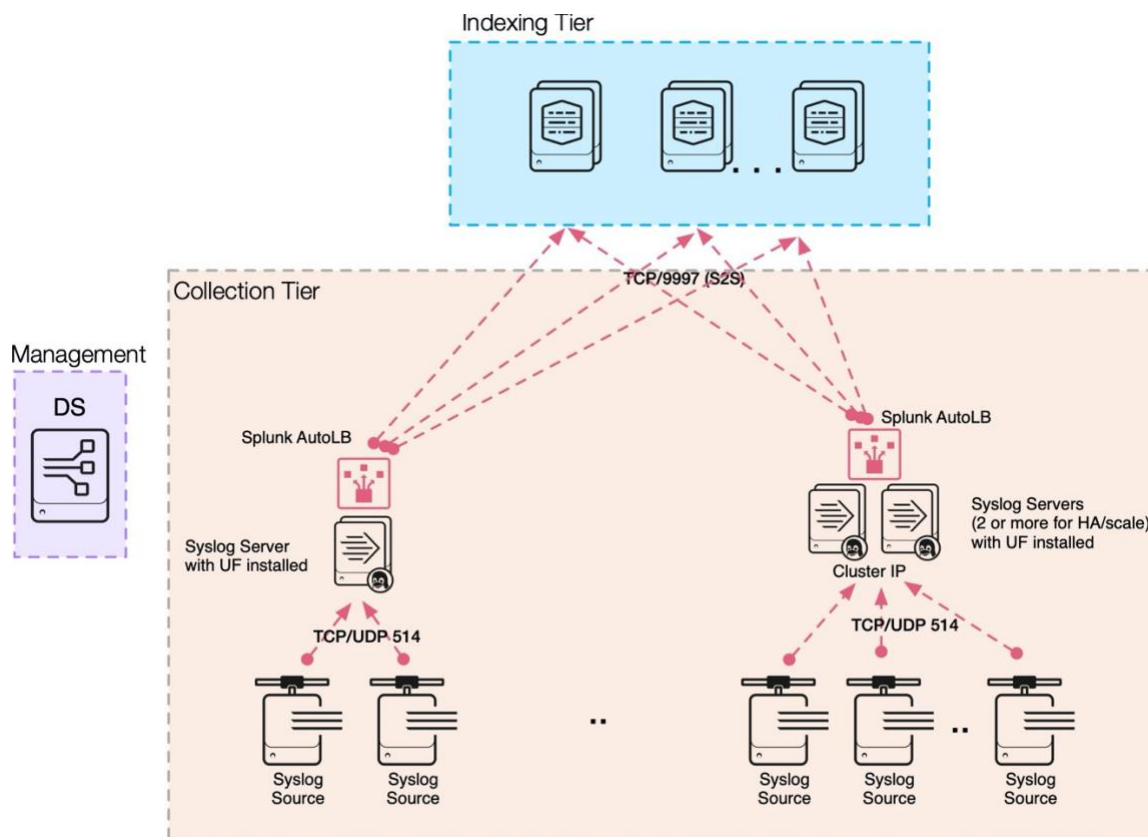
Syslog (File monitoring in conjunction with a syslog server)

Splunk can use monitoring (using `inputs.conf`) on a UF/HF to process and ingest syslog sources that are written to disk on an endpoint by a syslog server. Most commonly encountered, [rsyslog](#), [syslog-ng](#) and [Fastvue](#) offer commercial and free solutions that are both scalable and simple to integrate and manage in both low volume environments and large scale distributed environments.

To learn more about how to configure monitors, see [Monitor files and directories](#) in Getting Data In.

This architecture supports proper data onboarding in the same way a universal forwarder does on any other endpoint. You can configure the syslog server to identify multiple different log types and write out log events in appropriate files and directories where a Splunk forwarder can pick them up. This also adds a level of persistence to the syslog log stream by writing events to disk, which can limit exposure to data loss for messages sent using the unreliable UDP as transport.

Syslog Data Collection Topology using UF



The diagram shows syslog sources sending data using TCP or UDP on port 514 to a load-balanced pool of syslog servers. Multiple servers ensure HA for the collection tier and can prevent data loss during maintenance operations. Each syslog server is configured to apply rules to the syslog stream that result in syslog events being written to dedicated files/directories for each source type (firewall events, OS syslog, network switches, IPS, etc.). The UF that is deployed to each server monitors those files and forwards the data to the indexing tier for processing into the appropriate index. Splunk AutoLB is used to distribute the data evenly across the available indexers.

The deployment server shown in the management tier can be used to centrally manage the UF configuration.

Splunk UDP Input

Splunk can utilize a direct UDP input on a UF or HF to receive data from syslog. To learn about configuring TCP and UDP ports, see [Get data from TCP and UDP ports](#) in Getting Data In. The ability to receive events on UDP 514 relies on the ability of the UF/HF to run as root. Additionally, the agent must be available 100% of the time to avoid the possibility of data loss. Forwarders may be restarted frequently to apply configuration changes, which pretty much guarantees data loss. For these and other reasons, **this is not considered a best practice for a production deployment.**

High-Availability Considerations for Forwarding Tier components

There is a common concept of high availability (HA) in the digital world. However, depending upon the organization, the meaning can vary and be more in line with disaster recovery (DR) as opposed to high availability. These two concepts, while similar, do have different meanings. HA is a characteristic of a system, which aims to ensure an agreed level of operational performance, usually uptime, for a higher than normal period. DR involves a set of policies, tools, and procedures to enable the recovery or continuation of vital technology infrastructure and systems following a disaster.

The following outlines various forms of HA at the intermediate/aggregation tier:

Intermediate tier

- For customers with intermediate/aggregate tier deployments, HA of forwarders is mission-critical. At the application layer, Splunk currently does not have a native support method for HA. There are other strategies for providing HA at the operating system level that are not native to Splunk. Common solutions include VMWare VMotion, AWS Autoscaling Groups and Linux Clustering. Consult with your Splunk Architect to discuss other design options available to you.
- For environments with an HA requirements for a dedicated HEC tier, it's a best practice to use a network traffic load balancer (NTLB), such as NGINX, in front of multiple Splunk heavy forwarders. This provides the advantage of maximum throughput, scale and availability. You have a dedicated pool of HTTP event collector instances whose only job is to receive and forward data. You can add more HEC instances without necessarily having to add more indexers. If your indexers become a bottleneck, add additional indexers.
- Using a network load-balancer (L4/L7) to provide a highly available syslog collection infrastructure is discouraged due to potential data loss and fidelity concerns (IP lost in proxy). The recommended approach is to use a cluster resource manager solution (e.g. pacemaker/keepalived) or a VM-based syslog server deployment instead. Splunk does not have any insight into available technologies at your company, so please engage the in-house experts in your network team to design and deploy such an HA solution.
- Using a DNS based solution (Global Load Balancer) should be limited to resolving the closest receiver. Most common source devices do not honor DNS TTL and can be subject to data loss for minutes to hours if a DNS-based approach is chosen for load distribution.

Finally, where possible, UDP is preferred over TCP for syslog sources. While TCP can recover in some cases (single packet loss at high throughput), the recovery is more than likely to result in the loss of more than one event.

Forwarding tier

At the forwarding (endpoint) tier, HA for the agent itself is dependent upon the underlying OS. At the very minimum, you should ensure that any services that implement forwarding functionality are restarted automatically when the host OS restarts. Outside of that, best practices for the forwarders would involve the configuration and proper use of AutoLB from the forwarders to multiple indexers. This also may involve use of the indexer acknowledgement feature in order to guarantee data arrives at the indexing tier at least once (potential duplicates with indexer acknowledgement must be handled when searching).

Design Principles and Best Practices

Below you will find design principles and best practices separated by deployment tier.

Deployment Tiers

SVA design principles cover all of the following deployment tiers:

Tier	Definition
Search	Search heads
Indexing	Indexers
Collection	Forwarders Modular Inputs Network HEC (HTTP Event Collector) etc.
Management / Utility	CM DS LM DMS SHC-D

Aligning Your Topology With Best Practices

You will need to keep your requirements and topology in mind in order to select the appropriate design principles and best practices for your deployment. Therefore, you should consider best practices only after you have completed Steps 1 and 2 of the Splunk Validated Architectures selection process above.

Best Practices: Tier-Specific Recommendations

Below you will find design principles and best practices recommendation for each deployment tier. Each design principle reinforces one or more of the SVA pillars: Availability, Performance, Scalability, Security and Manageability.

Search Tier Recommendations

DESIGN PRINCIPLES / BEST PRACTICES (Your requirements will determine which practices apply to you)		SVA PILLARS				
		AVAILABILITY	PERFORMANCE	SCALABILITY	SECURITY	MANAGEABILITY
1	<i>Keep search tier close (in network terms) to the indexing tier</i> Any network delays between search and indexing tier will have direct impact on search performance		✓			
2	<i>Avoid using multiple independent search heads</i> Independent search heads do not allow sharing of Splunk artifacts created by users. They also do not scale well with respect to resource utilization across the search tier. Unless there is a specific need to have isolated search head environments, Search Head Clustering is a better option to scale.	✓		✓	✓	✓
3	<i>Exploit Search Head Clustering when scaling the search tier</i> A search head cluster replicates user artifacts across the cluster and allows intelligent search workload scheduling across all members of the cluster. It also provides a high availability solution.	✓		✓		
4	<i>Forward all search heads' internal logs to indexing tier</i> All indexed data should be stored on the indexing tier only. This removes the need to provide high- performing storage on the search head tier and simplifies management. Note: This also applies to any other Splunk roles.		✓			
5	<i>Consider using LDAP auth whenever possible</i> Centrally managing user identities for authentication purposes is a general enterprise best practice, simplifies management of your Splunk deployment and increases security.		✓			✓
6	<i>Ensure enough cores to cover concurrent search needs</i> Every search requires a CPU core to execute. If no cores are available to run a search, the search will be queued, resulting in search delays for the user. Note: Applicable to Indexing Tier as well.				✓	✓
7	<i>Utilize scheduled search time windows as possible / smooth scheduled search load</i> Often, scheduled searches run at specific points in time (on the hour, 5/15/30 minute after the hour, at midnight). Providing a time window that your search can run in helps avoiding search concurrency hotspots.	✓	✓	✓		
8	<i>Limit the number of distinct search head clusters so as not to overwhelm indexing tier</i> Search workload can only be governed automatically within a given SH environment. Independent SHCs have the potential to create more concurrent search workload than the indexer (search peer) tier can handle. The same is true for carefully planning the number of standalone search heads.	✓		✓		
9	<i>When building Search Head Clusters, use an odd number of nodes (3,5,7, etc.)</i> SHC captain election is performed using a majority- based protocol. An odd number of nodes ensures that a SHC can never be split into even numbers of nodes during network failures.	✓				✓

Indexing Tier Recommendations

DESIGN PRINCIPLES / BEST PRACTICES (Your requirements will determine which practices apply to you)		SVA PILLARS				
		AVAILABILITY	PERFORMANCE	SCALABILITY	SECURITY	MANAGEABILITY
1	<i>Enable parallel pipelines on capable servers to take advantage of available resources</i> Parallelization features enable exploitation of available system resources that would otherwise sit idle. Note that I/O performance must be adequate before enabling ingest parallelization features.		✓	✓		
2	<i>Consider using SSDs for HOT/WARM volumes and Summaries</i> SSDs have reached economical prices and remove any possible IO limitations that are often the cause for unsatisfactory search performance.		✓			
3	<i>Keep indexing tier close (in network terms) to the search tier</i> Lowest possible network latency will have positive effect on user experience when searching.		✓			
4	<i>Use index replication when historical data / report HA is needed</i> Index replication ensures multiple copies of every event in the cluster to protect against search peer failure. Adjust the number of copies (replication factor) to match your SLAs.	✓				
5	<i>Ensure good data onboarding hygiene (e.g. line breaking, timestamp extraction, TZ, and source, source type, host are properly and explicitly defined for each data source) and establish ongoing monitoring using the Monitoring Console</i> Explicitly configuring data sources vs. relying on Splunk's auto-detection capabilities has been proven to have significant benefit to data ingest capacity and indexing latency, especially in high-volume deployments.		✓	✓		✓
6	<i>Consider configuring batch mode search parallelization setting on indexers with excess processing power</i> Exploiting search parallelization features can have a significant impact on search performance for certain types of searches and allows you to utilize system resources that may otherwise be unused		✓	✓		
7	<i>Monitor for balanced data distribution across indexer nodes (=search peers).</i> Even event/data distribution across the search peers is a critical contributing factor for search performance and proper data retention policy enforcement.		✓	✓		✓
8	<i>Disable web UI on indexers in distributed/clustered deployments.</i> There is no reasonable need to access the WebUI directly on indexers.		✓		✓	✓
9	<i>Consider Splunk pre-built Technology Add-Ons for well-known data sources</i> Rather than building your own configuration to ensure data onboarding hygiene for well understood data sources, Splunk-provided TAs can provide faster time to value and ensure optimal implementation.		✓			✓
10	<i>Monitor critical indexer metrics</i> Splunk provides you with a monitoring console that provides key performance metrics on how your indexing tier is performing. This includes CPU and memory utilization, as well as detailed metrics of internal Splunk components (processes, pipelines, queues, search).	✓	✓			

Collection Tier Recommendations

DESIGN PRINCIPLES / BEST PRACTICES (Your requirements will determine which practices apply to you)		SVA PILLARS				
		AVAILABILITY	PERFORMANCE	SCALABILITY	SECURITY	MANAGEABILITY
1	Use UF to forward data whenever possible. Use of the Heavy Forwarder should be limited to the use cases that require it. Built-in autoLB, restart capable, centrally configurable, small resource demand		✓			✓
2	Use at least 2x intermediary forwarding pipelines to indexers when funneling many UFs Multiplexing a large number of endpoint forwarders across a small number of intermediary forwarders impacts even event distribution across indexers, which affects search performance. Only deploy intermediary forwarders if absolutely necessary.	✓	✓			
3	Consider securing UF-IDX traffic using SSL/TLS Encrypting data in transit reduces the amount of data transmitted and improves the security of your deployment				✓	
4	Use native Splunk LB to spray data to indexing tier Network load-balancers are not currently supported <u>between forwarders and indexers.</u>	✓		✓		
5	Utilize Splunk Connect for Syslog (SC4S) containers for syslog collection as close to the data sources as possible. SC4S can be quickly deployed and configured to collect data from most popular data sources with minimal effort.	✓	✓	✓		✓
6	Use HEC for agent-less collection (instead of native TCP/UDP) The HTTP Event Collector (HEC) is a listening service that allows events to be posted via the HTTP[S] protocol. It can be enabled directly on indexers, or configured on a heavy forwarder tier; both served by a load balancer.	✓				✓

Management / Utility Tier Recommendations

DESIGN PRINCIPLES / BEST PRACTICES (Your requirements will determine which practices apply to you)		SVA PILLARS				
		AVAILABILITY	PERFORMANCE	SCALABILITY	SECURITY	MANAGEABILITY
1	<i>Consider consolidating LM, CM, SHC-D and MC on a single instance for small environments</i> These server roles have very little resource demands and are good candidates for colocation. In larger indexer clusters, the CM may require a dedicated server to efficiently manage the cluster.					✓
2	<i>Consider a separate instance for DS for medium to large deployments</i> Once a significant number of forwarders are managed via the Deployment Server, the resource needs will increase to where a dedicated server is required to maintain the service.					✓
3	<i>Consider multiple DSs behind LB for deployments with a very large number of managed systems</i> Note: This may require help from Splunk professional services to be set up and configured properly	✓		✓		
4	<i>Determine whether DS phoneHomeIntervalInSecs can be backed off the 60 second default</i> A longer phone home interval will have positive effect on DS scalability			✓		
5	<i>Use dedicated/secured DS to avoid client exploitation via app deployment</i> Anyone with access to the Deployment Server can modify Splunk configuration managed by that DS, including potentially deploying malicious applications to forwarder endpoints. Securing this role appropriately is prudent.				✓	
6	<i>Use the Monitoring Console (MC) to monitor the health of your deployment and alert on health issues.</i> The monitoring console provides a pre-built, splunk- specific set of monitoring solutions and health checks, and contains extensible platform alerts that can notify you about degrading health of your environment.	✓	✓			✓

Summary & Next Steps

This whitepaper has provided a general introduction to Splunk Validated Architectures. A Validated Architecture ensures that your organization's requirements are being met in the most cost-effective, manageable and scalable way possible. SVAs offer best practices and design principles built upon the following foundational pillars:

- Availability
- Performance
- Scalability
- Security
- Manageability

This whitepaper has also covered the 3-step Splunk Validated Architectures selection process:

- Definition of requirements,
- Choosing a topology,
- Applying design principles and best practices,
- and covered both search and indexing tiers as well as the data collection tier.

Now that you are familiar with the multiple benefits of Splunk Validated Architectures, we hope you are ready to move forward with the process of choosing a suitable deployment topology for your organization.

Next Steps

So, what comes after choosing a Validated Architecture? The next steps on your journey to a working environment include:

- **Customizations**
Consider any necessary customizations your chosen topology may need to meet specific requirements
- **Deployment Model**
Decide on deployment model (bare metal, virtual, cloud)
- **System**
Select your technology (servers, storage, operating systems) according to Splunk system requirements
- **Sizing**
Gather all the relevant data you will need to size your deployment (data ingest, expected search volume, data retention needs, replication, etc.) and consult with a Splunk expert to size your environment
- **Staffing**
Evaluate your staffing needs to implement and manage your deployment. This is an essential part of building out a Splunk Center of Excellence

We are here to assist you throughout the Validated Architectures process and with next steps. Please feel free to engage your Splunk Account Team with any questions you might have. Your Account Team will have access to the full suite of technical and architecture resources within Splunk and will be happy to provide you with further information.

Happy Splunking!

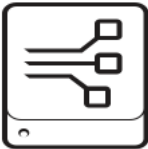
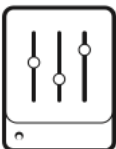
Appendix

This section contains additional reference information used in the SVAs.

Appendix "A": SVA Pillars Explained

Pillar	Description	Primary Goals / Design Principles
Availability	The ability to be continuously operational and able to recover from planned and unplanned outages or disruptions.	1. Eliminate single points of failure / Add redundancy 2. Detect planned and unplanned failures/outages 3. Tolerate planned/unplanned outages, ideally automatically 4. Plan for rolling upgrades
Performance	The ability to effectively use available resources to maintain optimal level of service under varying usage patterns.	1. Add hardware to improve performance; compute, storage, memory. 2. Eliminate bottlenecks 'from the bottom up' 3. Exploit all means of concurrent processing 4. Exploit locality (i.e. minimize distribution of components) 5. Optimize for the common case (80/20 rule) 6. Avoid unnecessary generality 7. Time shift computation (pre-compute, lazily compute, share/batch compute) 8. Trade certainty and accuracy for time (randomization, sampling)
Scalability	The ability to ensure that the system is designed to scale on all tiers and handle increased workloads effectively.	1. Scale vertically and horizontally 2. Separate functional components that need to be scaled individually 3. Minimize dependencies between components 4. Design for known future growth as early as possible 5. Introduce hierarchy in the overall system design
Security	The ability to ensure that the system is designed to protect data as well as configurations/assets while continuing to deliver value.	1. Design for a secure system from the start 2. Employ state-of-the art protocols for all communications 3. Allow for broad-level and granular access to event data 4. Employ centralized authentication 5. Implement auditing procedures 6. Reduce attack or malicious use surface area
Manageability	The ability to ensure the system is designed to be centrally operable and manageable across all tiers.	1. Provide a centralized management function 2. Manage configuration object lifecycle (source control) 3. Measure and monitor/profile application (Splunk) usage 4. Measure and monitor system health

Appendix "B": Topology Components

Tier	Component	Icon	Description	Notes
Management	Deployment Server (DS)		The deployment server manages configuration of forwarder configuration.	Should be deployed on a dedicated instance. It can be virtualized for easy failure recovery.
	License Master (LM)		The license master is required by other Splunk components to enable licensed features and track daily data ingest volume.	The license master role has minimal capacity and availability requirements and can be colocated with other management functions. It can be virtualized for easy failure recovery.
	Monitoring Console (MC)		The monitoring console provides dashboards for usage and health monitoring of your environment. It also contains a number of pre-packaged platform alerts that can be customized to provide notifications for operational issues.	In clustered environments, the MC can be colocated with the Master Node, in addition to the License Master and Deployment server function in non-clustered deployments. It can be virtualized for easy failure recovery.
	Cluster Manager (CM)		The cluster manager is the required coordinator for all activity in a clustered deployment.	In clusters with a large number of index buckets (high data volume/retention), the cluster manager will likely require a dedicated server to run on. It can be virtualized for easy failure recovery.
	Search Head Cluster Deployer (SHC-D)		The search head cluster deployer is needed to bootstrap a SHC and manage Splunk configuration deployed to the cluster.	The SHC-D is not a runtime component and has minimal system requirements. It can be colocated with other management roles. <u>Note:</u> Each SHC requires its own SHC-deployer function. It can be virtualized for easy failure recovery.
Search	Search Head (SH)		The search head provides the UI for Splunk users and coordinates scheduled search activity.	Search heads are dedicated Splunk instances in distributed deployments. Search heads can be virtualized for easy failure recovery, provided they are deployed with appropriate CPU and memory resources.
	Search Head Cluster (SHC)		A search head cluster is a pool of at least three clustered Search Heads. It provides horizontal scalability for the search head tier and transparent user failover in case of outages.	Search head clusters require dedicated servers of ideally identical system specifications. Search head cluster members can be virtualized for easy failure recovery, provided they are deployed with appropriate CPU and memory resources.

Tier	Component	Icon	Description	Notes
Indexing	Indexer		Indexers are the heart and soul of Splunk. They process and index incoming data and also serve as search peers to fulfill search requests initiated on the search tier.	Indexers must always be on dedicated servers in distributed or clustered deployments. In a single-server deployment, the indexer will also provide the search UI and license master functions. Indexers perform best on bare metal servers or in dedicated, high-performance virtual machines, if adequate resources can be guaranteed.
Data Collection	Forwarders and other data collection components		General icon for any component involved in data collection.	This includes universal and heavy forwarders, network data inputs and other forms of data collection (HEC, Kafka, etc.)
	Splunk Connect for Syslog (SC4S)		SC4S is the current best practice approach for SYSLOG data collection.	We created a dedicated icon for SC4S to reflect a fundamentally different, containerized deployment model for this data collection tier component.