SPLUNK FOR MANAGED SERVICE PROVIDERS
TECHNICAL ARCHITECTURE

How to deploy Splunk for Managed Service Providers or large organizations with multiple Splunk deployments.
# Table of Contents

Who should read this? ........................................................................................................... 3  
What are some of the MSP’s requirements? ........................................................................ 3  
  Lower the cost of delivery per customer ................................................................. 3  
  Repeated work ................................................................................................................. 3  
  Ongoing system administration costs ....................................................................... 3  
  Sharing resources ........................................................................................................... 3  
  Data segregation and leakage ................................................................................... 3  
  Performance and configuration impacts on other customers .................................. 4  
  Ways to manage multiple customer profiles or types ............................................. 4  
  High-Availability (HA) and Disaster Recovery (DR) ................................................. 4  
  Latency requirements ................................................................................................. 4  
  Alert triaging ............................................................................................................... 4  
  Reports for customers ............................................................................................... 5  
The Multi-Tenancy Discussion ................................................................................. 5  

Technical Architecture ................................................................................................. 6  
  Overview ....................................................................................................................... 6  
  Data Collection (Forwarding tier) ............................................................................ 7  
    Technique 1: Customer-premise Splunk HF with syslog server .......................... 8  
    Technique 2: MSP-premise Splunk HF with VPN from customer site .......... 8  
    Technique 3: Combination of MSP-premise Splunk HF + syslog server AND a Customer-premise syslog server ............................................. 9  
    Intermediate forwarders at MSP ........................................................................... 10  
      Dedicated Splunk Stacks ...................................................................................... 10  
      Shared Splunk Stacks ............................................................................................ 10  
Indexing Tier ................................................................................................................ 10  
  Option 1: Dedicated Full Stacks (Recommended) .................................................. 10  
  Option 2: Shared Splunk Stack ................................................................................ 11  
    Advantages .............................................................................................................. 11  
    Disadvantages ........................................................................................................ 11  
    Dedicated Indexes per customer ........................................................................ 12  
    Adopt a naming convention ................................................................................. 12  
    Name the indexes .................................................................................................. 12  
    Name the customer in each raw event ................................................................. 13  
    Name the customer in lookup data ..................................................................... 13  
    Create catch-all indexes ...................................................................................... 13  
Role Based Access Control (RBAC) ........................................................................ 13  
  Roles .......................................................................................................................... 13  
  Users ......................................................................................................................... 13  
Searching Tier ............................................................................................................. 14  
  Architecture ............................................................................................................... 14  
  Role Based Access Control .................................................................................... 14  
Deployment, Orchestration, and Management and Integration with Operations Center Workflow ................................................................. 14  
  Integration with Operations Center Workflow .................................................... 14  
  Central View Search Heads .................................................................................... 14  
  Customer Reporting ............................................................................................... 15  
Deployment, Orchestration, and Management ......................................................... 15
Who should read this?
This document is intended for Managed Service Providers (MSP) interested in designing Splunk environments for use as the backbone of a managed service that offers IT operations monitoring or the like. The environment envisioned here would include only Splunk Enterprise. If premium solutions like Splunk IT Service Intelligence (ITSI) are needed, that is certainly possible, but the deployment of those premium solutions is beyond the scope of this paper. The exception being Splunk Enterprise Security, which is covered in the companion whitepaper to this one - *Splunk for Managed Security Service Providers*. 

Aside from MSPs, this document is also relevant to anyone interested in multi-tenant-like Splunk environments. These could be large corporations, governmental entities or universities, all of whom are often interested in running multiple Splunk environments reporting up to a central entity, sometimes called “Splunk as a Service”. The final architecture for those scenarios may look a little different from what is described here, as the architecture here is designed around the requirements of MSPs. Many of the requirements will be similar, so we expect a lot of the architectural patterns described here to be relevant to these alternative use cases as well.

What are some of the MSP’s requirements?
As you can expect, this is not an exhaustive list, but here are a few things that most MSPs will care about. These are the requirements that guide the architectural design described further on.

Lower the cost of delivery per customer
When managing large numbers of customers (thousands in some cases), MSPs consider the cost of delivering their service to each customer to be very important. Costs can be of various types:

Repeated work
Avoiding manual and custom, non-repeatable work is key to a low-cost operation, so it is important to be able to spin up a new customer in an easy, automated fashion. This includes technical work like system installation and configuration, and also acquiring software licenses and transacting with the vendor (like Splunk).

Ongoing system administration costs
Lowering the internal costs of work performed to administer a customer under management is very important. The more economically the operations can be performed, the better the MSPs margins. This technical work falls into two buckets: system administration of the products used (configuration, upgrades, patching etc.), and actually managing the customer’s operations using these systems (monitoring, acting on alerts, investigations etc.). The costs extend across the customer lifecycle – onboarding a new one, managing them actively for the extent of the contract, and decommissioning them at the end of the lifecycle.

Sharing resources
This is another important aspect. Whether it is hardware, software, or hard-to-staff people, MSPs are interested in extracting the most out of every resource they have. Virtualized hardware starts to look appealing. So does sharing human output and reusing work. Anything that requires significant human effort to create - workflows, rules, or similar content, for example – is always a candidate for reuse and repurpose. The ideal situation is when a piece of content is created once and reused across a much larger set of customers.

Data segregation and leakage
MSPs are required to keep each customer’s data separate from another. This is an important concern. The requirement comes under various names - data hygiene, data segregation or data leakage. The basic idea is the same, but the appropriate level of separation at the technical level depends on the legal and security requirements of the MSP and its end customer. Some customers demand entirely separate physical hardware, others are satisfied with VM-level segregation, while others are happy with filesystem- or application-level separation. This topic is covered in more depth in the Multi-Tenancy section.
Performance and configuration impacts on other customers

Similar to the data segregation problem, the workload of one customer should not impact the performance of another, nor should configuration changes to one customer upset another. Each customer has an individual agreement with the MSP and, regardless of what their Service Level Agreement (SLA) with the MSP entitles him or her to, the customer generally does not expect to be inconvenienced or have the service quality lowered on account of other customers. Understanding the impact of things that are “shared” among different customers, such as virtualized infrastructure, is important here. This is also covered more in the Multi-Tenancy section.

Ways to manage multiple customer profiles or types

Grouping customers into a limited number of types or buckets helps reduce complexity and cost. Using templates to quickly configure a customer falling into a particular bucket is an added plus. Various tools make this job easier - version control systems like git or Subversion for storing configurations and templates, and automation systems like Chef, Puppet, Ansible or Salt for deploying them. Systems like this let the MSP spin up appropriate hardware that is known to work for a certain set of requirements, and configure that environment with an appropriate set of content, including apps, searches, reports, dashboards and alerts. Different customers have different needs, and governmental customers often have unique concerns, so many different templates could be used in practice.

High-Availability (HA) and Disaster Recovery (DR)

High-Availability and Disaster Recovery are going to be a requirement for most MSPs, since the service must be running at all times. Multisite indexer clusters and search head clusters will be common. Note that the complete HA/DR solution for the MSP will extend beyond Splunk to the other systems that must remain running to provide the service in whole.

Latency requirements

Latency between sites is obviously relevant for HA/DR – it is desirable to recover operations quickly in case of outage. There is another aspect to it as well - the SLAs the MSP is subject to. To see if it has complied with its SLA to respond to an event, the MSP will be interested in comparing the time that an event occurred with when it was investigated. The times used for comparison here are an interesting point of discussion. All customer environments will not be synchronized to the same Stratum-1 time source, so correlating times between the customer and MSP is not straightforward. Even less so when trying to see if an event occurred for multiple customers. If the MSP uses the customer’s time as the frame of reference, and calculates intervals based on that, it would be left having to manage potentially thousands of different times (for thousands of customers), with much confusion. The easier solution is a single way to interpret times for all customers for SLA purposes. This is most conveniently done by considering the MSP’s time to be canonical. The clock can start at the time that the event made its way into the MSP’s environment, as shown at the MSP. Further defining this to the time that the event was indexed, rather than the time that it occurred, is one way to approach this. This is not necessarily a business concern for all MSPs, so this should not become a complex topic of discussion unless it is really necessary. It is worth bringing up when the topic of SLAs comes up, however.

Another concern with being able to respond to incidents in a timely fashion, is keeping the time to index data and thereby make it searchable within Splunk down to a minimum. This is directly relevant to discussions of how Splunk will perform on systems with different CPU, memory, network and storage resources.

Alert triaging

MSP analysts need to triage alerts efficiently for their customers and turn them into incidents when appropriate. They are usually subject to SLAs that mandate that incidents be escalated to the customer within a specified time frame. What constitutes an “incident” is a matter of interpretation, so the severity of the problem escalated to the customer can often vary. These SLAs are important because they are part of the contract the MSP undertakes with the customer, and SLA breaches can be grounds for
terminating a contract. This is an area well worth exploring, because all other things being equal, higher quality escalations take longer to generate. This results in a natural tradeoff. If, for example, an MSP escalates more alerts than they should out of fear of breaching an SLA, they are not really offloading work from their customer and performing the service they were hired for.

2. One customer’s environment cannot be allowed to impact another’s performance. Performance impacts reduce an MSP’s ability to service that customer effectively, and this can affect how well they are meeting their SLA.

3. Customers follow their own upgrade lifecycles, so their individual environments may look quite different. They need to be customized independently, which is most easily done when each customer’s environment is completely isolated.

4. Customers need to be protected against impacts from maintenance work being done on other customer environments. In a shared environment, many customers may need to be taken down to adjust something for only one of them.

For these reasons, our default architectural recommendation is that MSPs use separate Splunk installations per customer. This approach supports the requirements above quite well. If correlation across customers or any kind of multi-customer aggregation is needed, that can be accommodated by means of a central environment that pulls information out of the individual ones using Splunk’s comprehensive APIs. This is covered in more detail in the architecture section of this document. Visually, the recommended architecture looks like this:

![Figure 1 - Splunk Reference Architecture - default recommendation](image)

An alternative architecture that does not involve separate Splunk instances is also possible in certain circumstances. There are tradeoffs between this approach and the multi-instance architecture, detailed in the Technical Architecture.
Technical Architecture

Overview
The general problem space of Splunk as a Service is not limited to MSPs. Many different types of organizations find themselves needing to do this, so the guidelines here can be considered relevant for them as well, though they are tailored more to the concerns of MSPs. For example,

- Large corporations may have multiple business units or geographical sites, all subordinate in some way to a central entity.
- Universities may have multiple departments.
- Governmental agencies may have component agencies, or there may be shared services organizations whose purpose is to provide a shared service (like Splunk) to different departments.

And so on. Each of these will have its own specific use cases, legal constraints, data protection and segregation requirements, scale of customer base (number and size of each), deployment considerations, ongoing system administration and management concerns, and operations center workflow concerns. The final architecture chosen depends a great deal on these requirements. What follows can be considered a good starting point that balances requirements for most MSPs. We highly recommend consulting with Splunk architectural experts and professional services teams to build out the final infrastructure.

The architecture outlined below makes no recommendations on where, physically, to deploy Splunk – MSP premise or customer premise. That said, it is expected that Splunk will be deployed within the MSP’s own premises, whether that be a datacenter or private cloud. (While similar patterns may be followed if they choose to deploy within Splunk Cloud, the architectural decisions there will be made by the Splunk Cloud Operations team, and MSPs will generally not be involved in the specifics, as is typical in SaaS situations). Deploying at their own premises is in line with typical practices at MSPs, as they centralize operations within one or more operations centers and often prefer to have as little as possible deployed at the end-customer’s premises. This offers the advantages of proximity for maintenance and monitoring, and avoids the costs of truck rolls. This is entirely a business decision on the MSP’s part, however. If they choose to deploy Splunk in the manner below at their customer’s premises, that is perfectly acceptable.

The default recommendation for handling multiple customers is to deploy multiple instances of Splunk, each dedicated to a customer. Each deployment includes all the software and hardware needed to run Splunk Enterprise and any other apps needed. (Note again that premium solutions like ITSI and Enterprise Security are beyond the scope of this paper). Conceptually, it looks like this:

![Figure 2 - Splunk Reference Architecture – default recommendation](image)

From the MSP’s perspective, this is still a multi-tenant business offering, i.e. their environment hosts multiple customers. Behind the scenes, we recommend that they deploy separate environments for each customer to maximize their chances of achieving all of the following goals:

1. Data segregation between customers.
2. No performance impacts from one customer on another.
3. Ability to grow, mature and configure each environment independently without fear of impacting another.
4. Ability to maintain and temporarily bring down each customer if necessary, without fear of impacting another.

If a single view of the different environments is needed, this can be done either via single central view Search Head that sees across all environments, or via a custom portal that pulls in information using the Splunk API. If central configuration of the different environments is required, this can be done by...
deployment and management processes and systems at the MSP level. Many of the pieces that deploy the whole of the environment needed to use Splunk effectively (since a functioning MSP environment could often extend beyond Splunk to other work systems like ticketing systems) will be custom-written by the MSP. This can also be done by Splunk Professional Services or Splunk partners. Technically, this may leverage well-known IT orchestration systems like Chef, Ansible or Puppet. This same infrastructure is useful for non-MSP, non-security specific use cases, as well. For example, a large multinational may want or need each sub-division, legal sub-entity or business unit to have their own Splunk deployment.

Each customer’s environment can come in one of two flavors.

1. **Dedicated Splunk stacks per customer**
   
   This includes forwarders, indexers, and search heads, and anything else that would exist in any single distributed implementation. This is the default recommendation, suitable for most situations, and was shown in figure 2 above.

2. **Shared infrastructure**
   
   This shares resources at the intermediate forwarder, indexer and search head tiers, with all forwarders dedicated to specific customers as in any single system. This is a suitable approach for some situations and is shown in figure 4 below.

The different architectural considerations for these two scenarios will be covered below. In either case, these systems are managed solely by the MSP via a deployment system that relies on much automation, not the MSP’s end customer themselves.

The architecture will be described below in order of data flow:

1. Data collection from within the MSP’s customer’s environment
2. Indexing and searching
3. MSP operations center view

![MSP Architecture Overview](image)

**Figure 4 - Multi-customer architecture based on default recommendation – detailed view**

**Data Collection (Forwarding tier)**

Gathering data from the MSP’s end customer is essentially the same as gathering data from endpoints in any Splunk environment. For those platforms supported by the Splunk Universal Forwarder (UF), such as Linux and Windows servers and workstations, simply deploy the UF to those systems. Where a UF is not possible, alternate methods are used, such as a syslog server with a Splunk Heavy Forwarder (HF - a full Splunk installation used in a forwarding-only mode). In some cases, custom inputs are required to pull data from platforms incapable of utilizing the syslog protocol and unsupported by the Splunk...
packages, such as mainframes or ICS devices. These are the same data collection mechanisms used in any Splunk deployment.

Regardless of the collection method, one key requirement is that the data be sent to the MSP via secure transport. Although there are many ways to implement this, there are three methods discussed here:

1. A customer-premise system containing a Splunk HF and syslog server, sending data to the MSP.
2. An MSP-premise Splunk HF using a VPN between the customer and MSP sites.
3. A combination: MSP-premise Splunk HF and syslog server plus a customer-premise syslog server, with the customer-premise syslog server sending directly to the MSP-premise one.

The particular implementation chosen, being one of the above, a combination, or something else entirely, matters only in how well it integrates into the MSP and customer operational models. This is important to understand. As long as all relevant and needed data is sent to the MSP securely, the specific choice does not matter.

Technique 1: Customer-premise Splunk HF with syslog server

In many cases, the preferred approach is to place an MSP-controlled system at the edge of the customer site. This is used to aggregate all data from within the customer site, then send it to the MSP Operations Center. This is a fairly standard configuration for a Splunk HF install that uses syslog-ng, rsyslog, or something similar to collect syslog data from the various syslog data sources and any other universal forwarders deployed. The advantage of encrypting the data at the aggregation point is that it allows data to be collected from the internal systems in its original unencrypted format, without sending it in the clear across the WAN to the MSP. It also allows granular control over setting sources, sourcetypes, and other Splunk metadata using Splunk configurations and syslog filters.

In some cases, there may be multiple layers of syslog coming from largely isolated devices and networks within the customer site, requiring many local syslog servers. These servers would all send their data to the MSP-controlled customer-edge system. This approach takes advantage of existing centralized collection points within the MSP customer site to collect data, rather than implementing an entirely new way to get data to the MSP.

Aside from forwarders and syslog servers, the collection infrastructure within the customer premises may also require heavy forwarders, such as when a technical add-on is required to normalize data to CIM format, or when pulling data from a database using dbconnect. All of this data will be forwarded to the MSP-controlled HF-based central collection point before progressing beyond the customer perimeter.

Once data reaches the MSP-controlled Splunk HF system on the customer site, Splunk forwards the data to a Splunk intermediate forwarder (discussed below) in the MSP operations center DMZ. The Splunk-to-Splunk communication is over TLS, thus meeting encryption in transit requirements.

This system has a notable advantage in the case of WAN outages between customer and MSP sites. It provides a large local buffer that can collect data while the outage is in place, which can be sent on to the MSP once the outage is resolved. Testing latency, outage scenarios and corresponding buffering needs is highly recommended. For many implementations, the default buffer-related settings should suffice.

Technique 2: MSP-premise Splunk HF with VPN from customer site

An alternative to having an MSP-controlled system at the customer edge to aggregate and encrypt
the data is to eliminate that system, host an HF at the MSP instead, then establish a VPN between the MSP operations center and the customer across which all data is sent over syslog. The difference from the previous architecture is that all data from the customer site, whether from syslog or Splunk forwarders, routes directly via the VPN to a system hosted by the MSP, rather than first routing internally to a customer-site edge system provided by the MSP. This adds the complexity of maintaining a site-to-site VPN, but it removes the need for an MSP to manage a system at the customer site. Truck rolls are expensive, and this approach eliminates those in favor of taking on VPN maintenance instead.

Due to a higher potential for data loss, this is not recommended. Should the VPN or network layer become unavailable, thus stopping all traffic from the customer site to the MSP, the data intended for the MSP would have to be buffered at the customer edge until connectivity returns. With the removal of the forwarder at the edge, this buffer no longer exists. Where the original data is coming from endpoints or servers with universal/heavy forwarders installed, those forwarders can do some buffering at that level. Where the original data is sent over syslog, on the other hand (for example no UF installed), there is a problem; syslog generally is not designed as anything other than fire and forget. The buffers for sending syslog via TCP are minimal at best. The result is that some data loss will likely occur if the WAN link fails for more than a few seconds. The VPN connections themselves may prove less stable than the underlying WAN transport, so care must be taken for this particular approach.

Technique 3: Combination of MSP-premise Splunk HF + syslog server AND a Customer-premise syslog server

The third option is a hybrid design that involves two things:

1. A customer-premise syslog server at the customer site.

Under this scheme, different types of data are sent to the MSP in different ways.

- All Splunk UF/HF-supporting devices send to the MSP-premise Splunk HF directly across the WAN, using encrypted Splunk-to-Splunk communication.
- All syslog-supporting devices send first to an on-premise syslog server at the customer site, which forwards on to another syslog server hosted at the MSP. The primary advantage to this arrangement is that the traffic between the two syslog servers can use TCP and TLS for secure transmission of data, even when the bulk of the original syslog sources are only capable of sending syslog in clear-text via UDP.

This reduces some level of management overhead and complexity for the MSP by freeing them from having to manage a customer-premise Splunk HF, while gaining the encryption and buffering advantages of a customer-premise syslog server. Syslog servers are known technology, so customers may be able to maintain this themselves if needed.
Intermediate forwarders at MSP

Once data is collected onto the MSP’s customer-specific Splunk HF system, whether that is hosted by the MSP, on premise at the customer site, or a hybrid system as per the three options above, the data must be sent from an Internet facing host to the Splunk indexers that reside within the MSP’s boundary. The first landing point for data received by the MSP across the WAN is an intermediate forwarder, which is generally hosted in the MSP’s DMZ. This can be a universal or heavy forwarder depending on routing requirements.

These intermediate forwarders can be either dedicated to single customers or shared among multiple, depending on the indexer tier implementation. It is recommended that the architecture here match that of the indexing tier (described shortly).

The number of systems deployed as intermediate forwarders depends specifically on the ingest volume of the customer. All calculations should be based on the current recommendations for the version of Splunk used.

Dedicated Splunk Stacks

In an architecture with dedicated full stacks per customer, each customer should have at least two intermediate forwarders for high availability and redundancy, as well as to ensure even data distribution from the forwarders spraying data to them. This reduces network- or system load-related bottlenecks.

Shared Splunk Stacks

In an architecture with a shared Splunk stack, the MSP can either dedicate per-customer intermediate forwarders or share intermediate forwarders across customers. The recommendation is to match intermediate forwarder architecture to the indexer tier architecture that it feeds; if the indexer tier is shared, then the intermediate forwarder tier ought to be shared as well to reduce complexity and maximize hardware utilization. This also provides for easier calculations for matching intermediate forwarders to indexers in the recommended ratios (which depend on the Splunk version used) to ensure even data distribution.

In both cases - the customer-premise Splunk HF systems sending to the MSP's intermediate forwarders, and the subsequent step of the intermediate forwarders sending to the MSP’s indexers - the forwarder's outputs.conf should be set up to spray data evenly across the receiving systems. This has the added benefit of reducing the impact of network connectivity or other related outages on one or more of the intermediate forwarders. To force a load balancing setup:

- set forceTimebasedAutoLB to true
- ensure autoLB is set to the default of true
- ensure autoLBFrequency is set to an acceptable value, such as the suggested 5 seconds. This minimizes the time any data stream spends sending to a single intermediate forwarder, thus maximizing the even spread of data while minimizing the possible effects of an unavailable intermediate forwarder.

Indexing Tier

There are two methods for implementing the indexing tier. As discussed above, either each customer has their own dedicated full Splunk stack, or the indexing tier (and likely the intermediate forwarder per the recommendation above) is shared among multiple customers. This is an important decision to get right, and depends heavily on the business needs of the MSP. In general, we recommend dedicated full stacks per customer as a starting point for all implementations. See the Multi-Tenancy and Architecture Overview sections for why.

Option 1: Dedicated Full Stacks (Recommended)

This is easy to understand – each customer gets their own completely separate Splunk installation. Each customer’s data lands in separate Splunk indexers. This architecture also involves separate Search Heads or SHCs, and other ancillary components such as Deployment Servers, Monitoring Consoles (formerly known as Distributed Management Consoles) etc. The systems can be logically or physically separated. Logically separate systems would be separate Splunk software installations on the same hardware. Physically separated systems would be separate
Splunk VMs or physical machines. Either case dedicates each Splunk installation to a particular customer; it will contain only their data, and will be used only to handle their operations. Since each is a dedicated environment, it can be operated largely as you would any independent Splunk installation. We say “largely,” because, depending on the number of these environments and amount of central orchestration desired, they could each be controlled mostly by the central deployment and management systems used by the MSP, and consumed by the end customer not directly via the Splunk user interface but via external portals that pull Splunk data into them. These are decisions to be made by the MSP; there is no hard requirement for either in this architecture.

Even with dedicated full stacks, it is recommended that MSPs follow the customer-specific naming conventions described below in the Shared Splunk stack. This makes it easy to identify and back up data at the filesystem layer if needed, and easier to migrate to the shared Splunk stack if needed.

**Option 2: Shared Splunk Stack**

The shared Splunk stack is an alternative to the dedicated stacks architecture. In a nutshell, it involves logically segmenting each customer’s data into *indexes* (the logical unit comprised of bucket files on disk and available in Splunk for granular RBAC access settings) not *indexERS* (the system including operating system and Splunk software designated to ingest data). This option can be explored by MSPs with a number of small customers who fall below Splunk reference architecture sizes, where multiple customers can conceptually fit inside a single piece of reference hardware. There are advantages and disadvantages to this approach.

**Advantages**

If you are sizing your deployment in units of Splunk reference hardware, this approach allows better use of the excess capacity within the hardware. For example, a single reference indexer can ingest up to 250 GB/day, per Splunk 6.5. If the data volumes for a customer are much lower, a reference indexer could actually support more than one customer. Placing multiple customers’ data within the same Splunk indexer allows you to use the available capacity.

Another advantage is avoiding the performance overhead of many virtual machines, each running a separate Splunk indexer, which would be another way to squeeze multiple customers into a single reference indexer. These savings gained by eliminating VMs can matter in some cases, such as when margins are low enough that the hardware must be utilized to the maximum extent possible.

The use of a single Splunk indexer also cuts down on management overhead, since you are not administering many different clusters or, stepping outside Splunk, virtual machines.

**Disadvantages**

Since data is segmented into indexes, not indexERS, this approach does segment the data physically – indexes are separate directories. It does not provide the deep data segmentation that may be legally required in some circumstances, or which might be necessary due to extreme sensitivity of the data or client. When these situations arise, they call machine-level segmentation, virtual or physical.

Another disadvantage of this approach is that it does not protect one customer environment from the performance impacts of another’s workloads. Indexers do a lot more than indexing; they also do the bulk of the work in searching, so search loads from one customer can impact another. These search loads include such things as datamodel acceleration searches, not just user-generated searches.

A similar point can be made about maintenance windows. When one customer’s environment needs maintenance for any reason, taking an indexer offline temporarily could impact other customers. SLAs should take this into account.

Another factor that can influence index- vs indexer-level separation is the reality that customers have independent lifecycles at the MSP. They are acquired, upgraded to higher tiers, and EOL’ed separately, and each customer may be configured differently. While the degree to which environments can be customized is certainly part of the MSP’s business model – they may choose to have all environments be essentially identical – the acquisition/upsell/upgrade
aspects force other environments within the shared environment to make room for the new or growing environments. This means that the MSP must have a good understanding of how these environments may evolve, and create procedures to make sure that any impact on other customers is minimized. The dedicated stacks approach has the advantage of isolating customers much better from each other’s growth cycles.

A final factor to be aware of with the shared indexer option is that it requires a higher level of Splunk skill to implement than the dedicated stacks approach.

There are three main aspects to this:

1. Each customer’s data must be uniquely **identified** as belonging to that customer, as there can often be overlaps with hostnames, IP addresses, and other such things.

2. Each customer’s data must be **routed** correctly to indexes created and reserved solely for that customer. Data cannot be mingled, as reversing such mistakes is non-trivial. Access to these indexes must granted only to those who need it. This applies to the dedicated stacks approach as well, but that option allows you to wall off access to the entire Splunk system at the network or OS level, while the shared Splunk stack requires careful application of Splunk’s inbuilt RBAC.

3. The system must continue to **perform** well. Splunk can scale to support extremely large data volumes on the order of petabytes per day and the patterns for doing so are well-understood, but it does require more knowledge and experience than administering a small-scale system does. The assumption is that this shared environment would be fairly large - larger than any one dedicated environment - so extra engineering effort should be expected.

The dedicated stacks approach can be suitable for handling many small customer environments as well if the tradeoffs with the shared Splunk stack above are ruled unacceptable. This can be done by downsizing the hardware below reference hardware specification. Since all Splunk testing is done on reference hardware, Splunk does not provide official recommendations on how small an environment can be made. It is strongly recommended that MSPs work with Splunk directly if this the way they want to go.

Some specific guidelines for the shared Splunk stack follow.

**Dedicated Indexes per customer**

The same indexer cluster will be used for all customers. Each customer’s data should land in one or more **dedicated indexes** in the shared indexer cluster. This approach ensures no data cross contamination or leakage between customer indexes. The index-level separation is also necessary because it allows the use of tstats searches on accelerated data models. If a single index were used for multiple customers and we wanted to search only a particular customer’s data by applying search-time filters to the role we created to access that customer’s data, it would not work; tstats does not respect search filters. The index-level separation means that even without specifying filters like index=blah in the role, the tstats searches would only be allowed to access the particular indexes accessible by that role. Note that custom tstats searches on indexed data, or non-tstats searches, could still be restricted at the role level by setting a search filter like `index=CUST4893_*` for all searches conducted by that role.

**Adopt a naming convention**

For ease of data management (say if files are backed up manually) and easy visual identification, a naming convention should be adopted to uniquely identify each customer’s data. For example, there might be a customer ID or customer billing code, such as `CUST4893` or `CS4AB2Z`. If the MSP has no such identifiers already, it must create a unique nomenclature for use within the Splunk environment. The naming convention has other advantages - it allows the MSP to manage its business operations by obtaining easy statistics about customers (say trending over time, spotting the need for additional services etc.), or by comparing customers with each other.

**Name the indexes**

Every index’s name should clearly reflect which customer’s data it contains. This results in indexes with names such as `CUST4893_linux`, `CUST4893_windows`, or `CUST4893_cisco`.
Name the customer in each raw event
Each raw event should reflect which customer it belongs to in a way that cannot be missed by the analyst at search time. This makes the analyst’s work easier. The recommendation is to add an index-time field called customer to every event, whose value is the customer ID for that customer. This will be added alongside the usual index-time Splunk metadata – source, sourcetype and host. Adding one field does not add much to the storage requirements. Besides making it easy to identify whose event it is, this field also allows reports and dashboards to be filtered down to a particular customer or group of customers. While this can be done with search-time fields, those can be easily modified at search time and are not recommended here.

Name the customer in lookup data
Raw data can be enriched with data looked up from other databases to make the analyst’s task easier, and also offer some unique analytical abilities. This is done by means of lookups. Like the raw events, the lookup data should also name the customer it belongs to. This can be done for asset data lookup tables, for example, by making use of field like owner, bunit, and category and setting those to reflect the customer ID. This makes it easy to create custom dashboards with only that customer’s data in it.

Create catch-all indexes
Create a per-customer catch-all index that serves as a test case for flagging any data that is not correctly and explicitly routed to the customer-specific indexes. When everything is working well, this should never contain any data. If anything appears in here, it indicates either misconfiguration at the forwarder/ingestion level, or a missing/malfunctioning customer-specific index.

Role Based Access Control (RBAC)
MSPs may have restrictions on which operations center analysts can view a customer’s data. Analysts could be restricted to only accessing certain customers, for example. Customers may require analysts to hold certain security clearances, possess citizenship of a certain country, or be located in a particular region. Splunk’s RBAC facilities can accommodate these requirements.

Roles
Each indexer in either a dedicated or shared environment should have a customer-specific role created, intended to control access to that particular customer’s data. It is recommended that roles be named with a unique customer identifier, such as the customer ID mentioned earlier.

In the dedicated full stacks approach, the indexers are dedicated to a customer, and all indexes within contain only that customer’s data. The customer role can therefore be allowed to access any index, though it is good practice to restrict access to the internal indexes used by Splunk itself. Access to those should be reserved for administrators.

In a shared Splunk stack environment, the indexers are not dedicated to a customer, and each customer’s data is segmented into a set of indexes that are reserved for them. Each customer-specific role must be restricted to being able to search only the customer’s indexes. For example, the role cust4893 would be restricted to searching the indexes CUST4893_linux, CUST4893_windows, CUST4893_cisco, and any other index whose name begins with CUST4893.

Note that if you are using data models, you do not need to set any data model constraints in addition to the restrictions on indexes. There is no need for separate data models per customer, each with a constraint like index=CUST4893_*+. A single data model can extend across all customer-specific indexes, and when a search is run against it, it will respect the permissions granted to the user that is running the search. You do not need separate data model acceleration stores either.

Users
In addition to the role, there should be a corresponding customer-specific user with only the single customer-specific role, per customer. This is used anytime the operations center staff or any automated solution needs to safely interact with a particular customer’s data. Using clear naming conventions for the user accounts and roles is crucial for proper management in a shared environment and for clarity in any automation code in systems outside Splunk.
Searching Tier

Architecture
Search heads can be dedicated in the dedicated full stacks approach or shared in the shared Splunk stack approach. The recommendation is to implement SHC to provide high availability of the search tier, as losing access to the sole Search Head used for operations monitoring would leave the MSP with no visibility into the customer. The cluster can be spread across multiple sites for added redundancy. However, for fairly small MSP customers with relatively less stringent SLAs, the added complexity may be more overhead than needed. Each MSP should determine their target customer market's normal sizing to decide whether a single SH or SHC is better for their needs. There must be no more than 300ms latency between any two SHC members (as of Splunk 6.5). This likely is a concern only in certain multi-site situations.

Role Based Access Control
The fundamentals of RBAC needed are described in the earlier section on roles and users at the indexing tier. The same customer roles and users used to restrict access to indexes must be configured on each customer's dedicated Search Head/Search Head Cluster. In the case of shared indexers across customers, this is imperative to prevent cross-contamination and data leakage between customers.

Deployment, and Management and Integration with Operations Center Workflow

Integration with Operations Center Workflow
A key step to success is knitting the various customer-specific Splunk instances into the operations center analysts' operational workflow. Analysts can log into each customer's instance, and keep an eye on them via simple means native to all browsers – multiple windows or split screens. This is fine for a few instances; it is possible to keep an eye on multiple environments and conduct cross-customer investigations this way. Beyond a few instances, looking at different windows is no longer practical, so the recommended approach is to create a custom multi-customer view to show exactly what the MSP needs. These requirements will differ from MSP to MSP, so a single layout is unlikely to satisfy everyone here. In addition to Splunk, operations center staff will need access to other tools, such as ticketing tools or other means of handling investigation/escalation workflows, operations center-only and customer-facing portals, external visualization tools used, and perhaps custom intellectual property - solutions and value added services that the MSP brings to the table. A custom web application providing a view that meets all these needs is often the right solution here. Splunk offers excellent API and SDK support to facilitate this, so it can be developed in any modern web toolkit that the MSP prefers. Splunk professional services can also partner with the MSP on this effort.

An external multi-instance view such as this may choose to pull in events from all customers, or events that meet certain thresholds. The choice depends on the type of work the analysts intend to do in the overall view. It may not be the primary interface for all types of investigations. Sometimes the goal is to gain an overall view of all customers' operational status, whether there are any red flags overall, in other words. Other times it is cross-customer correlation - seeing if a problem is happening on other customers. Aside from the differing motivations behind these overall views, the follow-up actions can differ as well once something is seen. Does the analyst dive into the customer-specific instance to make full use of Splunk to investigate, or do they simply file a ticket for another customer-specific analyst to follow up on later, and return to monitoring the overall view? The range of possibilities here make it a good idea to closely collaborate with Splunk on designing the overall view so it integrates smoothly into the operations center's workflow. This will ensure that Splunk is able to bring its experience and best practices to bear on every requirement in play.

Central View Search Heads
If a custom web portal to look at multiple instances is not desired - integrated with the rest of their workflow or not - a single Search Head can also be used to can search all other Splunk instances and provide that centralized view. This is a regular Splunk search head specifically configured for this purpose, rather than a separate component or software
product from Splunk. This central view search head should be given access to all the customer indexers (or indexes, if the shared model is used). It can then be used to produce custom reports tailored to whatever overall-view requirements exist, or to conduct ad-hoc search investigations. Since all data is now available in a single view, it becomes even more important for analyst to easily be able to tell whose data they are looking at in the moment, regardless of whether the indexer tier is shared amongst customers or not. Things like the index-time customer field make the analyst’s life much easier here. Custom dashboards can also be built to display each customer’s data separately, and these can also key off the customer field.

This central view Search Head will search all other indexers, so expect it to increase the search load on them, overall.

Conceptually, the central view Search Head approach looks like this:

![Diagram](image)

**Figure 8 - Central view search head to correlate across multiple customers**

**Customer Reporting**

MSPs generate regular reports for customers, usually offered via a web portal. Splunk can used as the portal itself, providing customer-specific views that look considerably different from what the analyst sees. A separate reporting search head can be used for this, with far more limited access to the data, carefully curated content, and custom Splunk apps per customer, each containing only that customer’s reports. If the MSP does not wish to use Splunk as the reporting portal, such as when they already have a portal of their own, the external portal can query Splunk to build the dashboards the customer will see. This can include things like interesting events investigated, number and status of escalations, data ingest volumes, system health etc. Splunk’s APIs and SDKs make this task easy.

**Deployment, Orchestration, and Management**

Since multiple instances (or indexes) will exist to service the different customers, the best way to deploy them smoothly is to use an automated deployment and orchestration system that can control the whole process. This is likely the most sophisticated piece for the MSP to design and implement. The orchestration tools themselves are well-known within the IT world. The ideal deployment involves not only installing Splunk, but tying it closely to the operations center’s overall workflow. This involves integrating it with other operational systems like ticketing systems, and adding other things unique to each MSP – proprietary intelligence and enrichment, for example. All of this can be done manually, but a proper orchestration system makes it much easier to do it consistently across customers, faster, and with a much lower risk of human error. This approach achieves many of the advantages an MSP seeks – lower cost of delivery, faster time to revenue, and greater customer satisfaction as the standard parts of onboarding, upgrading and decommissioning (for customers exiting the service) are finished quickly.

The deployment and orchestration solution can be anything that provides the capability to:

- Store configuration data in a version control system.
- Deploy changes to multiple systems, via either server push or client pull mechanisms.
- Greatly customize and automate things.

There are myriad commercial and open source (with commercial support) solutions on the market that support these goals. The most commonly cited solutions are generally Ansible, CFEngine, Chef, Puppet, and Salt in alphabetical order. These have
varying levels of capability for configuration control vs. direct system orchestration, but any of them should be sufficient. Many of them have been used successfully to deploy Splunk in the past, even at non-MSP environments. Splunk's ability to be configured via text-based configuration files lends itself very nicely to this.

Splunk itself has many central management and monitoring components - the Monitoring Console (previously known as the Distributed Management Console) to look at the overall health of the distributed system, the Deployment Server to push out configurations to different components, the Deployer to configure Search Head Clusters etc. All of these operate within the context of a running Splunk environment, and can only function after Splunk is installed and running on a system. A fully automated deployment takes care of the preceding initial steps as well - spinning up the virtual machines (if used), installing Splunk on them, starting it up, and then invoking the built-in Splunk tools above to configure the cluster fully. The orchestration system can also control what content is pushed out across the cluster by feeding the Deployment Server - for example, standard content from central VCS repositories. Note that the overall goal here is to augment Splunk's built-in tools, not replace them entirely. The built-in tools understand the underlying mechanisms of a Splunk deployment much better, and handle all the little details that an external tool would never know about. This is an important point to remember.

The basic components required in such a deployment and orchestration system are:

1. A version control system (VCS) to store configuration files and other artifacts, such as git or Subversion.
2. A deployment and orchestration infrastructure, such as Chef, Ansible, Puppet or Salt.
3. Integration into the MSP operational tools and portal - this will vary for each MSP. The integration will make extensive use of Splunk’s integration capabilities - the APIs, SDKs and text configuration files.

To start the process, the MSP must design a gold standard customer environment, then copy all the configuration files and other artifacts needed for it, such as logos and other custom art, into the VCS. Then a mechanism must be devised to automate the creation of branches, copies, or other means for providing multiple variations and customizations for different customers within the VCS solution. All customer deployment configurations and operational management changes should be stored within the VCS. This allows for rapidly recreating them for disaster recovery and testing purposes. The simplest way to reduce errors and understand exactly what went into a customer environment is to push all changes from the VCS and not make any changes directly on the individual Splunk systems or environments. Changing individual systems puts the changes in two places - the VCS and the hand-edited files, which can be tricky to keep track of.

Creating a new customer environment requires forking, branching or some other mechanism in the VCS from the gold standard defaults for customers, followed by any changes to those files to accommodate the new customer. For example, the customer ID used for naming roles, accounts, and indexes, as well as possibly systems, must be changed from the generic defaults. The automated changes must also accommodate some level of scaling for the number of systems required to handle the data ingest and search volume. They must also handle license allocation for indexing volume needs.

The whole operation then must be integrated into the MSP's custom workflows and systems so the final system can be used by the operations center efficiently. The operations center must have the new customer environment information added to ticket systems, workflows, analysts' displays, and any other systems developed to maintain operations.

This list of systems to integrate Splunk with will likely will include a reporting portal of the type described earlier. Integration with this portal or other portals used by the operations center team may require a gateway of some sort. This is depicted in the diagram below on the architecture. It is important to understand that the gateway is not a Splunk component or even required. It’s more of a conceptual idea that some kind of authentication or pass-through may be required before the portals can make API calls to Splunk.
Summing up, spinning up new customer environments can include (not limited to):

- Creating a server, either virtual or physical
- Installing the operating system
- Installing the deployment and orchestration packages
- Installing the Splunk packages
- Using the deployment and orchestration tool to install the configurations from the VCS store specific to the customer.
- Integrating this Splunk environment into the operations center’s existing workflows and systems.

As with any engineering effort, it is highly recommended to create a pre-production or development environment to test out the entire process. This should mimic the final production environment as far as possible, including realistic data ingest from customers.

As with any engineering effort, it is highly recommended to create a pre-production or development environment to test out the entire process. This should mimic the final production environment as far as possible, including realistic data ingest from customers.

Figure 9 - Deployment and Management Process

Download Splunk for free or get started with the free cloud trial. Whether cloud, on-premises, or for large or small teams, Splunk has a deployment model that will fit your needs.