
splunk>
Splunk
Cloud MSP
Blueprint

Table of Contents

Introduction 4

MSP requirements 5

- Data sovereignty and access control 5
- Scalability and repeatability 5
- Customer onboarding costs 5
- Ongoing administration costs 5
- Complexity 5

MSP considerations 6

- Single pane of glass for the MSP analysts 6
- Fast onboarding of new platform engineers/administrators 6
- Splunk Blogs and Splunkbase 6
- Attribution of workload to specific customers 6

Documentation exclusions 7

- Data collection 7
- Splunk professional services 7

Architecture overview 7

- Key Splunk technology and features 7
- Recommended third-party technologies 7
- Version control system 7
- Secrets management that can be integrated with the automation tool 7
- Automated deployment tool (CI/CD) 7
- Recommended architecture 8

Why separate customers into different Splunk Cloud Platform environments? 9

Design pattern 10

- Anatomy of a single customer 10
- Description 10
- Admin Config Services (ACS) 11
- Splunk search head REST configuration endpoints 12
- Federated search 13

MSP/Customer benefits 14

- Quantifiable attribution of workload 14
- Simple and efficient searches 14
- Data segregation and access control 14
- Ability to support customers that have different geographical and regulatory requirements 14
- Introduction of new use cases 14

A typical analyst's flow 15

Configuring the environment 16

Index configuration 16

- Optional automation 16
- Configure indexes in indexes.conf and then distribute them as private apps using Admin Config Services 16
- Configure indexes using the Admin Config Services endpoint 16

Data models and data model accelerations 16

- Optional automation 16

Federated search 17

- Optional automation 18

Role-based filtering and federated search 19

Access control 19

- SAML 19
- Local accounts 19

Optional automation 20

- SAML automation 20
- Local account automation 20

Knowledge object management 21

- Limiting knowledge object management 21
- Process control 21
- Technical control 21
- Developing knowledge objects 21
- Approach 1 22
- Approach 2 22
- Optional automation 22

Table of Contents

- Splunkbase apps 23
- Local changes to Splunkbase apps 23
- Private apps 23
- Macros 24

App permissions 24

- Limiting app access to specific roles 24

Addendum 25

Automation recommendations 25

- Automation workflows 25
- Automated testing and validation 25
- Naming conventions 26
- Code validation 26
- AppInspect 26
- Testing 26
- Push of feature branch 26
- Pull request with main branch 26
- Programmatically generated configuration 27
- Example of configuring federated search and federated datasets using programmatically generated configuration 27
- Securing your repository 28
- Branch protections 28
- Approvals 28

Managing a customer with multiple search heads/search head clusters in an environment 29

- Federated search 29
- Private app management via ACS 29
- Option 1 29
- Option 2 29

Splunk Enterprise Security 30

Splunk Enterprise and bring your own license 30

Hybrid model (Splunk Enterprise and Splunk Cloud Platform) 30

MSP feature limitation matrix 31

Multi-stack architecture vs. single-stack architecture 32

Third-party tools for automation 33

Functional examples 34

- Customer monitoring use case 34
- Customer component 34
- MSP component 34
- Simple alerting use case 34
- Customer component 34
- MSP component 34

More questions? 35

Introduction

Splunk Cloud Platform is a powerful and flexible tool that supports the needs of customers with complex configuration requirements.

This document is intended for the Managed Service Provider (MSP) and customers with similar complex customer requirements that align with the service that an MSP might supply. The blueprint describes how to architect the base platform to support their offerings on Splunk Cloud Platform, enabling the MSP to meet the complex needs of the customers.

This architecture blueprint leverages Splunk Cloud Platform features to provide a platform for MSPs and their customers that keeps the complex needs of the customer and MSP as key design considerations. Its objective is to provide the experience of a unified solution to the MSP user, even though it will be made up of distinct building blocks.

It is assumed that the audience has a basic understanding of Splunk Cloud Platform. If you have limited experience with Splunk Cloud Platform, then there are some great courses from Splunk Education, and we recommend "Transitioning to Splunk Cloud" as an ideal starting point.

We will also be discussing third-party tools that support the MSP journey, so a basic understanding of the following will also prove helpful:

- Git and the GitFlow and/or GitHub flow
- CI/CD and deployment automation

splunk>

MSP requirements

This architecture has been developed with the MSP and their customers in mind and is aimed to meet many of the requirements that the MSP will be looking for a platform to solve.

Data sovereignty and access control

Data sovereignty

Increasingly we are seeing regional restrictions around some aspects of customer data. This might result in the need to restrict where a customer's data is stored or accessed from.

MSPs that try to provide for many customers within a single environment often struggle to meet regulatory requirements for data locality. This means that a more traditional MSP offering doesn't necessarily meet the modern requirements of their customers.

Access control

MSPs are required to keep each customer's data segregated using appropriate access controls. This is an important concern, usually at the top of the list of customer requirements. The requirement comes under various names — data hygiene, data segregation, data separation or data leakage.

Essentially, a customer wants to have confidence that no unauthorized access of their data can be made by either the MSP or other customers of the MSP.

Scalability and repeatability

Customer onboarding costs

Repeatable onboarding and offboarding of customers is critical. A solution should make it as easy to onboard customer #2 as customer #50. Ideally, it should be cheaper to onboard the more customers there are.

Ongoing administration costs

The MSP objective is to have a solution that is scalable and cost effective with the ultimate aim being that the cost per customer for administration activities should reduce as more customers are added. The solution should not become cost prohibitive to the MSP at scale.

Complexity

It is accepted that the size of a platform will increase as new customers are added, but this should not fundamentally affect the complexity of the solution. A solution should have a linear increase in complexity as more customers are added.

MSP considerations

Single pane of glass for the MSP analysts

The MSP is likely to be performing detection and response activities as a primary part of their service offerings. It is essential in any solution that they can see the KPIs and alerts for all customers within a single view to perform part of their workflow.

Fast onboarding of new platform engineers/administrators

MSP solutions/architectures often become complicated. This increases the barrier of entry to new team members, often requiring a new skill set before they are effective within the team.

An effective MSP solution should enable new colleagues within a team to be effective earlier to help achieve faster return on investment.

Splunk Blogs and Splunkbase

Historically many MSPs have had difficulty leveraging out-of-the-box content available from Splunk Blogs and Splunkbase, such as:

- Splunk Enterprise Security Content Updates (Splunkbase)
- Splunk Security Essentials (Splunkbase)
- Splunk Blogs relating to industry security events and zero-day alerts (surge blogs)

This challenge has been simplified with this approach, and as a result this content will be easier for MSPs to leverage without the need for customization.

Attribution of workload to specific customers

MSPs' customers will have a varied workload. Some will have large volumes of data, while others might be much smaller. Different customers might also want different use cases implemented to meet their specific requirements.

An effective solution should ensure two or more customer workloads cannot affect each other.

To reduce the complexity of MSP billing, the solution should be able to easily and robustly attribute workloads to a customer, so the MSP will ensure an effective customer billing model.

Documentation exclusions

This document will **not** cover the following topics.

Data collection

Data collection is something that is covered well in the Splunk ecosystem, and we recommend starting with the following resources:

- Splunk Validated Architectures
- Splunk documentation around getting data in

Splunk professional services

Complex data onboarding requirements might go beyond the aforementioned documentation, in which case the Splunk Professional Services team can be engaged via your account team to support you.

Architecture overview

The architecture blueprint we are recommending leverages Splunk technologies at their core. It can be configured using only these technologies successfully; however, to ensure repeatability and scalability, we will be including several third-party technologies to support the MSP through configuration management and automation.

All third-party technologies included have free and commercial options available depending on the MSP's chosen tooling. We will refer to these technologies at a high level rather than suggesting specific vendors.

Key Splunk technology and features

This architecture will utilize the following Splunk features and technologies:

- Splunk Cloud Platform (Victoria Experience)
- Federated search (standard mode)
- Admin Config Services (ACS)
- Splunk Cloud REST API

Recommended third-party technologies

We recommend leveraging third-party tools to provide the following features. These features will support automation of changes to the deployment and will enable a scalable and robust platform for the MSP.

Version control system

The purpose of a version control system is to allow collaboration on changes to be made by developers in a controlled fashion. It will allow the MSP to know what the current deployed configuration is and understand what changed, why and when.

Secrets management that can be integrated with the automation tool

Secrets management helps us with two challenges in this solution. We do not want to put sensitive information like passwords in our version control system, and we should aim for the principle of least privilege, meaning we want to keep our developers and users away from this sensitive information if they do not need it to perform their roles.

Automated deployment tool (CI/CD)

A tool that can help you automate both testing and deployment of configuration means that we can have confidence in our configuration before it is deployed. Automating will also improve confidence in the platform and will allow developers to focus more on developing content rather than deploying it to the MSP's platform.

Some examples of the third-party tools we discuss in this document are listed in the Addendum toward the end of this document.

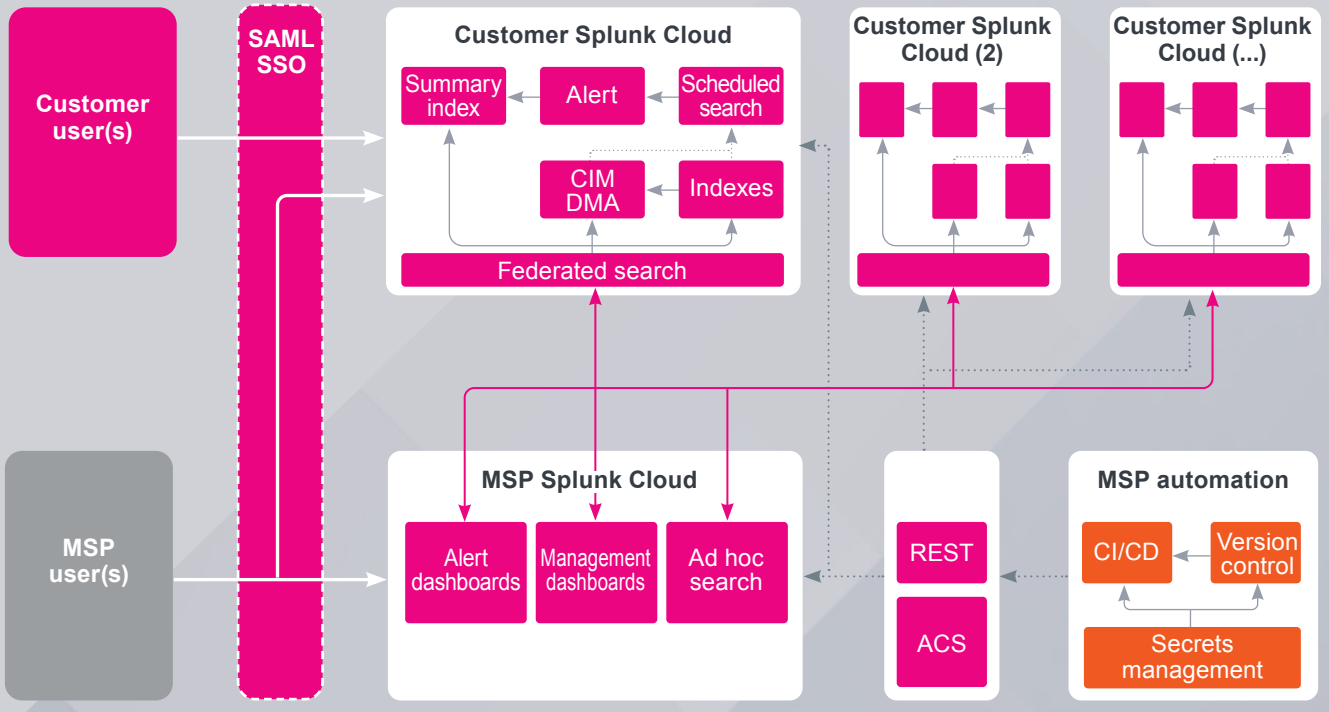
Recommended architecture

The diagram below is an example of an MSP architecture at a high level. Each customer has their own Splunk Cloud Platform environment, which ensures simple segregation of data, access control and attribution of workload.

The MSP Splunk Cloud Platform environment is connected to each customer environment using federated search, which allows for centralized status monitoring, aggregation of alerts and use case output. This MSP enviroment might also be used for limited use cases that require correlation of data across multiple customers at the same time, for example alert aggregation, environment monitoring or looking across all environments for an indicator of compromise (IOC) or behavior seen in a single customer.

MSPs can then optionally use the Admin Config Service (ACS) and the appropriate REST endpoints to manage all of the environments leveraging version control and third-party automation tools to ensure that multiple stacks can be managed as a single platform while also providing the benefits that configuration management tools provide, such as the ability to track “What changed?,” “When did it change?,” “Why did it change?,” and “Who changed it?”

This model also ensures customers can have different requirements, such as retention, data volume and workload, while ensuring a manageable and scalable solution for the MSP.



Why separate customers into different Splunk Cloud Platform environments?

The world is changing faster than ever. Compliance regulations are evolving, especially with respect to data sovereignty and retention. This, coupled with the bespoke and complex nature of your customers, means a single Splunk Cloud Platform environment does not give the MSP service the flexibility it needs to meet the customer's needs in a scalable way.

A single Splunk Cloud Platform stack looks simple on the surface, but access control, data segregation and even searches create additional complexity that grows with every customer that is added to the MSP service. This soon creates a platform that can be brittle and difficult to effectively manage.

The additional complexity of many customers in a single environment also translates into a less performant solution. This is due to the additional dimension of knowing which customer was involved in each search, which increases the amount of resources that the search will require to execute.

In order to meet the data governance requirements enforced within some regions, like the EU, a single multi-tenant architecture approach would mean a copy of the MSP multi-tenant architecture might be required when entering a region for the first time. For example, these governance requirements apply when data is not allowed to leave the borders of a country or region.

Separate Splunk Cloud Platform environments may seem more complex, but they actually make the environment simpler and more efficient for both MSPs and their customers. It allows MSPs to build uniform or unique environments to meet the customers' needs. Separate environments also ensure that access control is simple because MSPs do not have to handle the complexities that come with trying to ensure one customer cannot see another customer's data, or with having to add complex and expensive filtering to searches and use cases.

Design pattern

Anatomy of a single customer

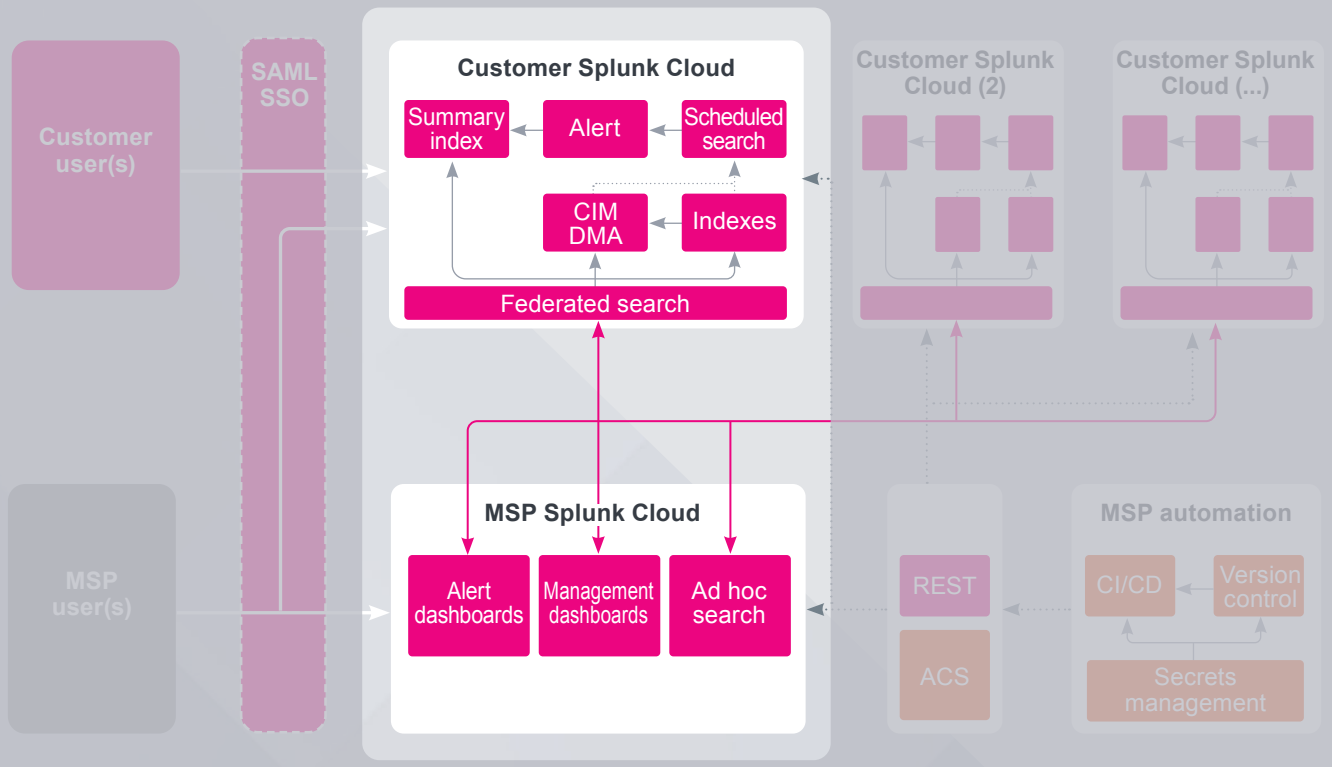
Description

Each customer will likely be provided with the same basic elements to ensure the service being offered by the MSP is sustainable. If MSPs treat every customer as completely bespoke, then they cannot achieve the lowest platform management overheads.

Use cases, where possible, should run within the customer environment rather than being initiated from the MSP Splunk Cloud Platform environment. This prevents unnecessary data transfer between the customer cloud environment and the MSP cloud environment and reduces Search Processing Language (SPL) complexity.

The use cases run for the customer will output a notification/alert that the MSP will consume and act upon. Summary indexes are an ideal target for these notifications to be stored. They can be consumed over federated search from the MSP hub environment for aggregation and reporting as appropriate.

MSPs can leverage all Splunk features within the customer environment and can have environments with or without premium apps like Enterprise Security (Enterprise Security will be covered specifically later in this document).

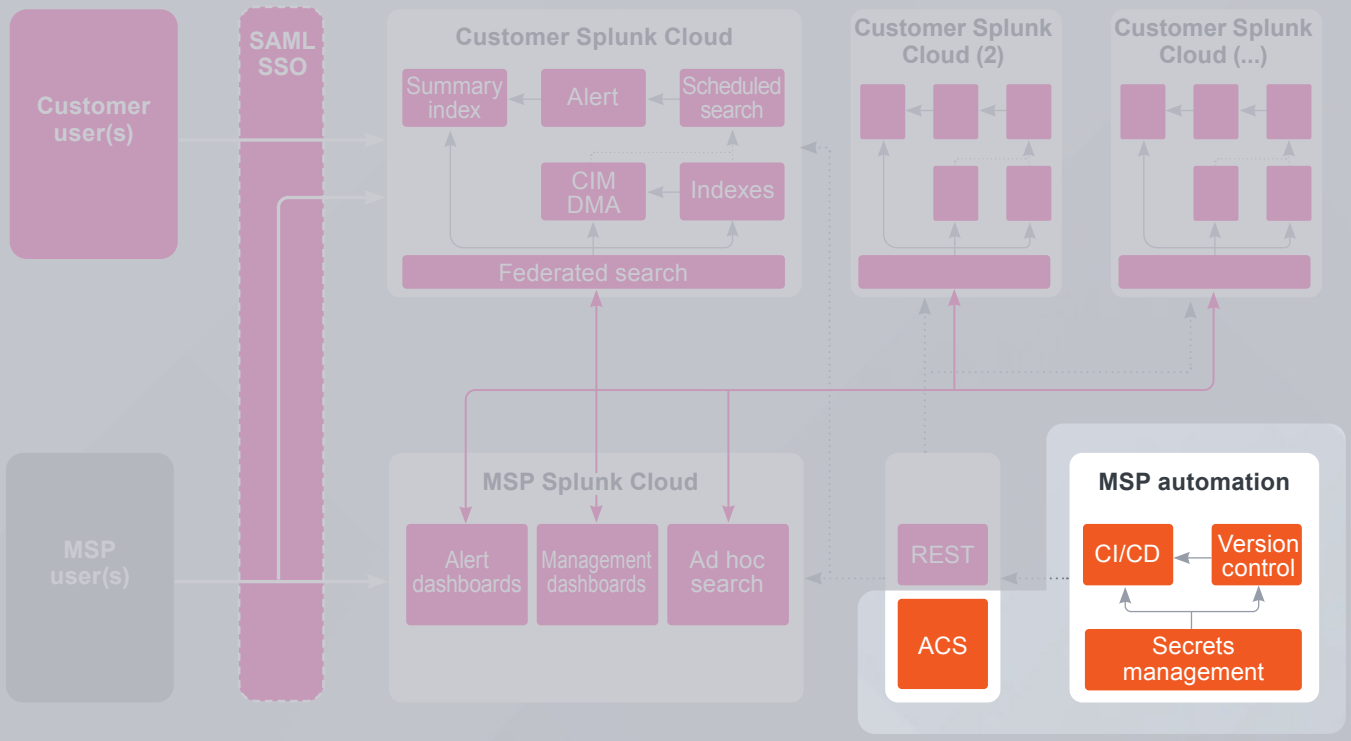


Admin Config Services (ACS)

Each customer's environment can be managed using third-party automation, and ACS will be a key part of that. Admin Config Services is a set of REST endpoints, similar to the Splunk REST API that MSPs may be familiar with, that allow for programmatic configuration of Splunk Cloud Platform.

There is also a command line interface (CLI) version of Admin Config Services available which some users may prefer.

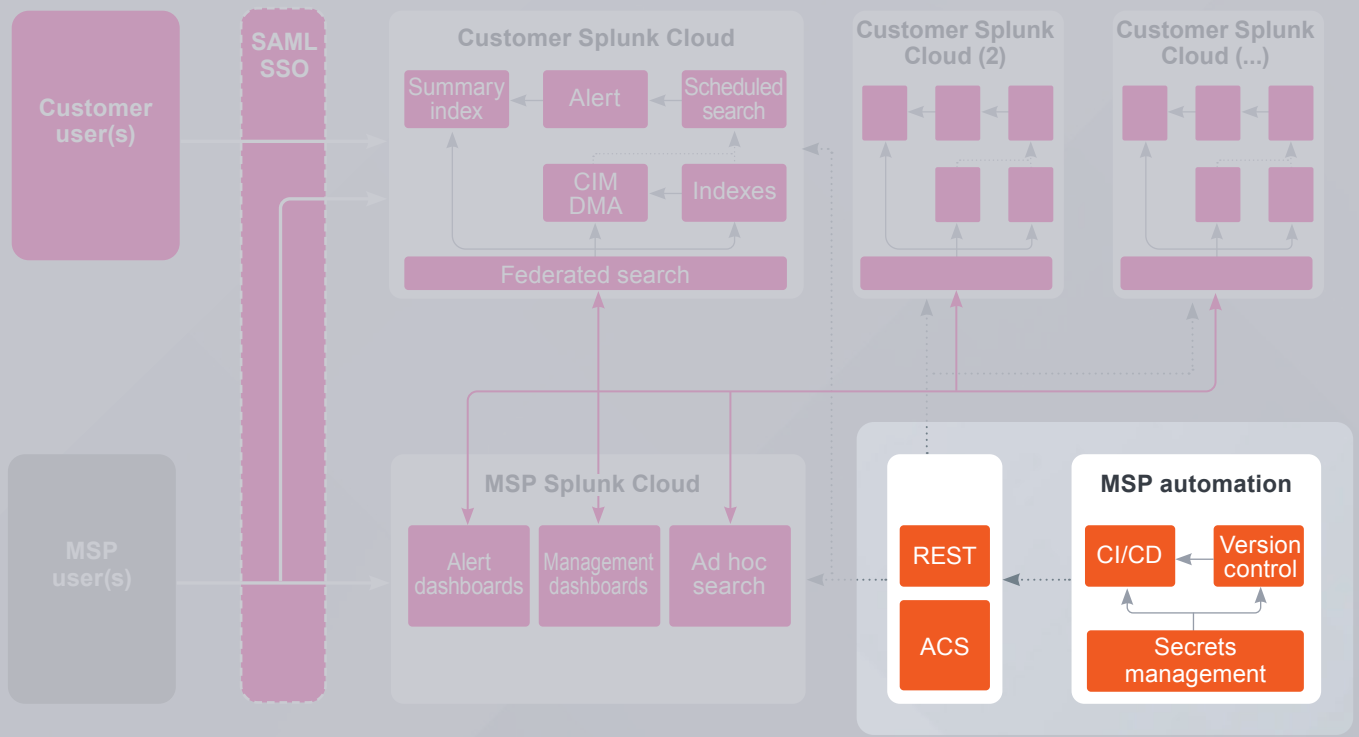
The current features of Admin Config Services can be located in Splunk's Admin Config Service Manual.



Splunk search head REST configuration endpoints

While most configuration will be managed via Admin Config Services, there might be specific functionality that automation would configure using available REST endpoints on the search heads. This might include:

- Authentication configuration that might not be achievable via ACS
- Enabling of certain content deployed via ACS (covered in “Managing a customer with multiple search heads/search head clusters in an environment” in the Addendum)
- Configuration of federated search



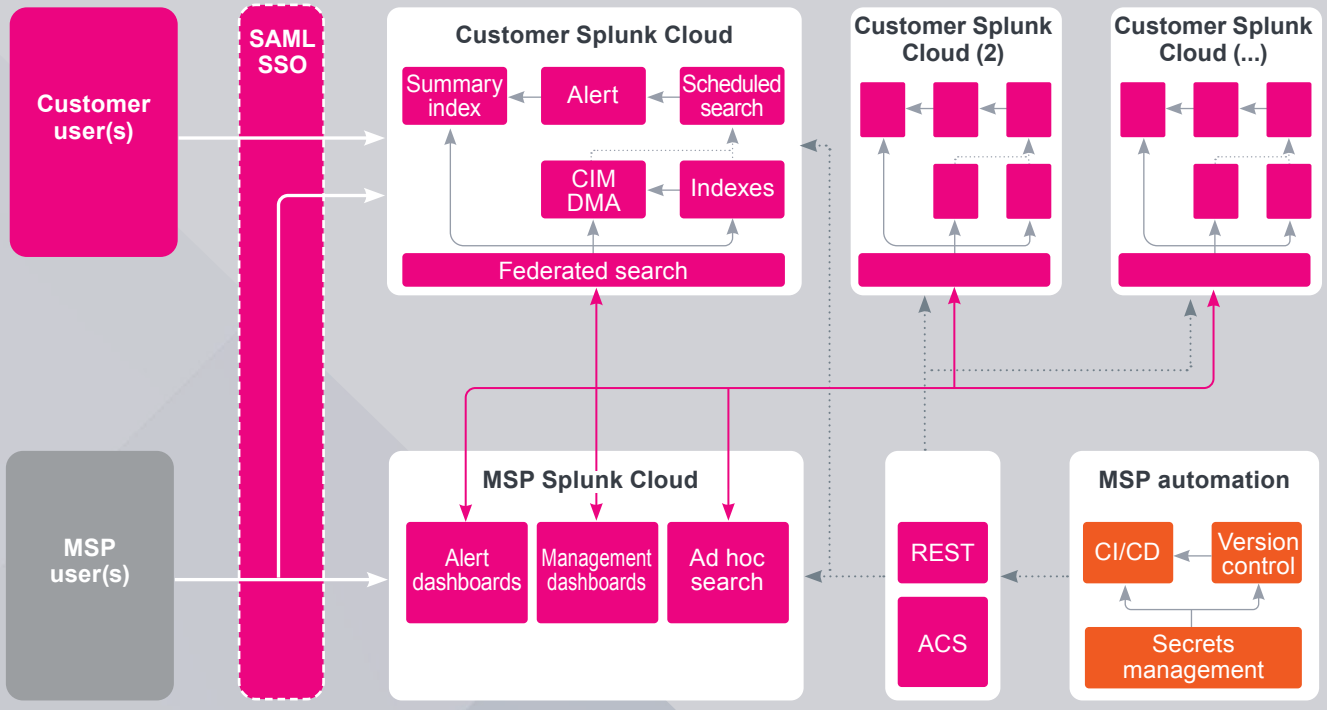
Federated search

As part of the day-to-day operations of the MSP, there will be a requirement to interact with the customer environments both individually and en masse, so there is also a need to ensure that searches can run across the environments selectively when needed. To this end, federated search in standard mode is the key component.

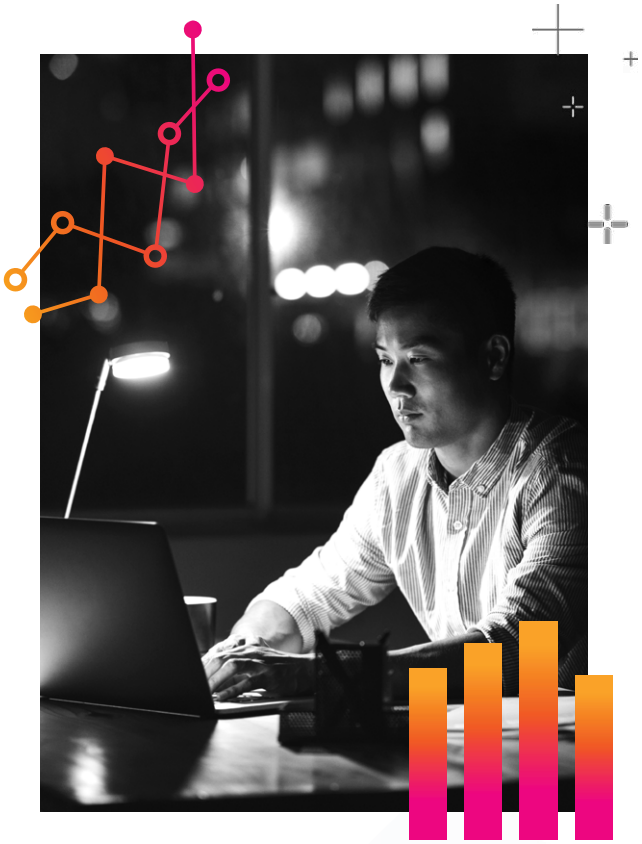
Standard mode in federated search allows multiple levels of access control, which will increase customer confidence around who can access their data. Also, as analysts specify where to target a search, they end up with searches that are only distributed to one or more customer environments from the MSP environment when the search explicitly needs to be targeted at that customer.

Note: Transparent mode for federated search is not appropriate for the MSP architecture pattern described in this document. This is because the MSP Splunk Cloud environment cannot selectively choose where a search is executed in this mode and the MSP Splunk Cloud Platform environment would always have access to all customer data within the customer Splunk Cloud Platform Stack.

Federated search will use one or more service accounts that are defined on each customer Splunk Cloud Platform stack. These service accounts will manage what the MSP Splunk Cloud Platform stack has the potential to see within the customer environment. These service accounts should be local accounts and can be configured on the relevant customer search head automatically using API calls for scalable creation and modification at onboarding and over time.



MSP/Customer benefits



Quantifiable attribution of workload

Separating the majority of the work that is being performed for a customer makes environment sizing and workload billing easier. One customer cannot skew the utilization of another customer nor the MSP environment as a whole.

Simple and efficient searches

As most workloads are being executed within a single customer environment, MSPs do not need to add complexity and dimensions to searches to ensure that a use case aggregates by customer.

Data segregation and access control

Separating customers ensures simple and robust control of customer data, so one customer can never access another customer's data even in the event of a misconfiguration.

This customer separation allows for access control strategies to be simple and scalable.

Ability to support customers that have different geographical and regulatory requirements

Your customers are likely to have requirements that mean they might not all be able to operate within the same Splunk Cloud Platform region. By separating customer environments MSPs can ensure customers are operating where they need to without the MSP having to deploy multiple versions of their offering across many regions.

Introduction of new use cases

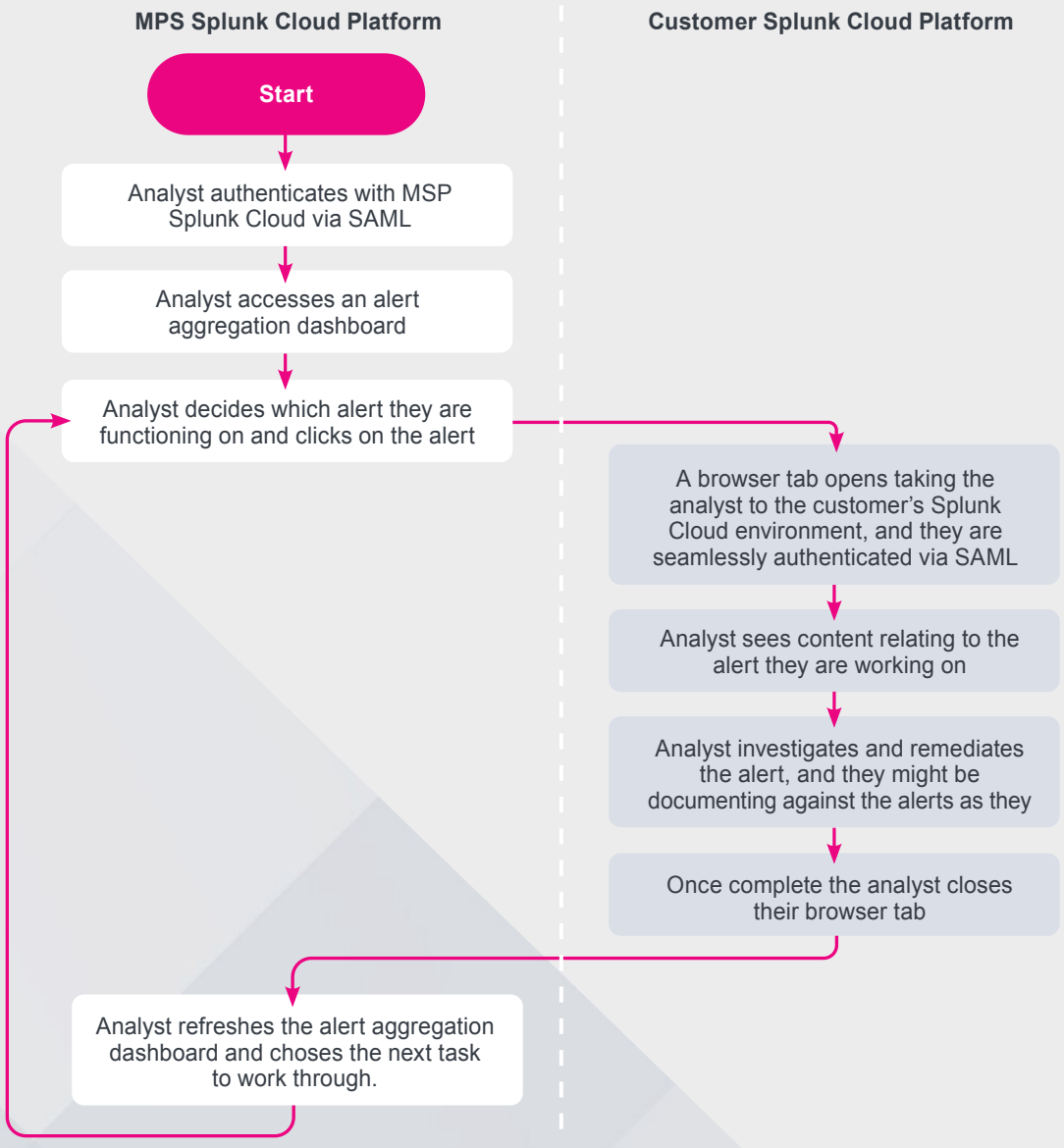
Because the customer can have simple access to their environment in this model, it gives the MSP the opportunity to start upselling customers with new use cases that extend beyond the original offering that was sold.

A typical analyst's flow

Once the Splunk Cloud Platform is provisioned and functioning for the customer, most of the work that will be performed will be reactive, in response to indicators/ notifications or alerts that have been triggered by the scheduled searches defined and active for the customer.

The analyst might have a central dashboard on the MSP Splunk Cloud Platform stack that allows them to see all alerts that have been triggered across all customers. While in some scenarios they might perform some initial triage following a defined workflow from the MSP environment, it is more likely that they will pivot to the customer's environment to continue with the investigation. They might click on the alert in this central triage dashboard, which would then pivot the analyst to that particular customer's environment, and the integration could leverage single sign-on (SSO) so that the analyst gets an experience of a single platform as they will be authenticated to the customer's environment seamlessly.

Once the analyst is authenticated with the customer environment, they would leverage workflows (dashboards and content) that have been defined for that use case. These would have been deployed to the customer's environment via automation and ACS. They could also use the search interface to perform parts of the investigation. Because all of the customer environments are configured in a similar way, the analyst's experience should feel very consistent across all the customer environments.



Configuring the environment

The features that will need to be considered when building out the environment are listed here. Each feature is described as if it were being performed manually, and then a subsection for each describes how the MSP could automate that functionality.

Index configuration

Splunk recommends using a common naming convention for indexes across customers. This allows for an MSP analyst to interact with every customer environment uniformly, without needing knowledge specific to the customer environment, which would add complexity. It also allows for searches to be configured in a uniform way.

At a minimum, an index will be needed for alerts/notifications to be written to when a scheduled search outputs results (very similar to how Enterprise Security (ES) or IT Service Intelligence (ITSI) create “notables”). This will allow for retention of an audit record of the alerts/notifications that the customer would be able to view. It will also be the point of reference for the aggregation and alerting dashboards on the MSP environment.

Optional automation

There are two ways that indexes could be managed via automation.

Configure indexes in indexes.conf and then distribute them as private apps using Admin Config Services

Create Read Update Delete (CRUD) functionality would be possible using automation.

Configuration would be performed by creating an app in version control and then pushing it to the relevant customers as a private app using Admin Config Services. As this is a private app, it will have to be vetted using AppInspect.

The pipeline would likely push a new version of the private app when a change occurs within the app.

Configure indexes using the Admin Config Services endpoint

CRUD functionality would also be possible via this approach. Configuration would be stored in version control and then submitted to the Admin Config Services index management endpoint. The service behind the endpoint would then perform the appropriate CRUD operations that have been submitted.

The pipeline could perform the relevant CRUD operations and can gather the current state of the environment, so when the pipeline runs it might validate the current state of the stack and apply any changes necessary so that the customer and MSP environments meet the current state defined within the version control system.

This is the recommended approach.

Data models and data model accelerations

Data models such as the Splunk Common Information Model are useful for standardizing events so that the interesting elements have standard naming conventions for a specific purpose. These can be leveraged with this solution.

Acceleration of data models can be leveraged to speed up the execution of each detection that is running and to reduce overall load on the customer environment, subject to the limits defined in the Splunk Cloud Platform Service Description.

A common set of data models is the Splunk Common Information Model, which is an app available from Splunkbase and certified for Splunk Cloud Platform use.

Data model accelerations would be generated within a customer environment and accessed locally using the customer Splunk Cloud environment or via federated search from the MSP Splunk Cloud Platform instance if needed.

Optional automation

The Splunk app for the Common Information Model can be deployed using the ACS Splunkbase apps endpoint.

Configuring the accelerations and macros as defined in the app’s documentation would be performed using the “Local changes to Splunkbase apps” section (later in this document) or via the REST endpoint.

If the environment has multiple search head groups, which is common in Splunk Cloud Platform environments with premium apps installed, then we will want to enable data model accelerations and report accelerations selectively using REST to ensure that we do not have multiple versions of the same acceleration enabled in the environment.

Federated search

This is the key functionality required for this architecture. It will need configuring for every customer onboarded to the solution.

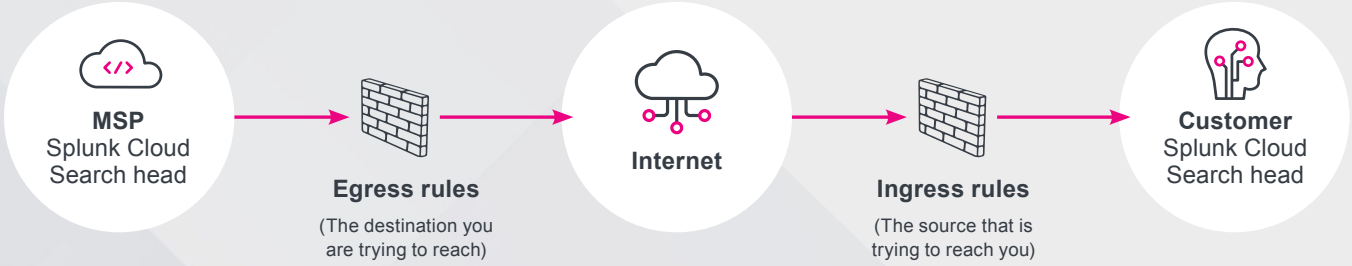
If the customer has Enterprise Security, then a decision will be made by the MSP as to whether to integrate with the Enterprise Security search head or the ad hoc search head.

1. Federated search will require a service account to be created on the customer environment (remote search head) with the relevant permissions to appropriately control access within that customer environment.
2. When configuring federated search, it is necessary to configure ingress and egress for each environment for the Splunk Management port to enable federated search connectivity as represented in the following diagram. Ingress and egress can be configured via the user interface (UI), using Admin Config Services CLI or via a support ticket.

   To keep the number of platform changes to a minimum Splunk recommends:

   - Configure ingress rules on the customer environment (remote search head) to the IP ranges relevant to the AWS region where your MSP Splunk Cloud Platform environment is located
   - Configure egress rules on the MSP Splunk Cloud Platform environment to the IP ranges relevant to the regions where the customer Splunk Cloud Platform environments are hosted

   Splunk Cloud Victoria Experience is hosted within AWS enabling the configuration of rules to allow connectivity from the IP ranges of the various AWS regions. More restrictive rules could be put in place but would need to be monitored regularly and updated as the Splunk Cloud Platform infrastructure changes over time to ensure connectivity is not impacted.
3. The connection from the MSP Splunk Cloud Platform search head can now be configured. First the MSP should define a federated provider that allows for connectivity to the customer stack. This should be configured in standard mode using the service account created in Step 1.
4. Now it is necessary to update the federated indexes so that the MSP stack can access datasets on the customer environment. MSPs should ensure they use naming conventions that easily allow the analysts to work, for example, `<cust_identifier>_<index>`. (As a minimum for initial setup, configure internal indexes to allow for monitoring and the summary index created for alert artifacts that was created earlier).

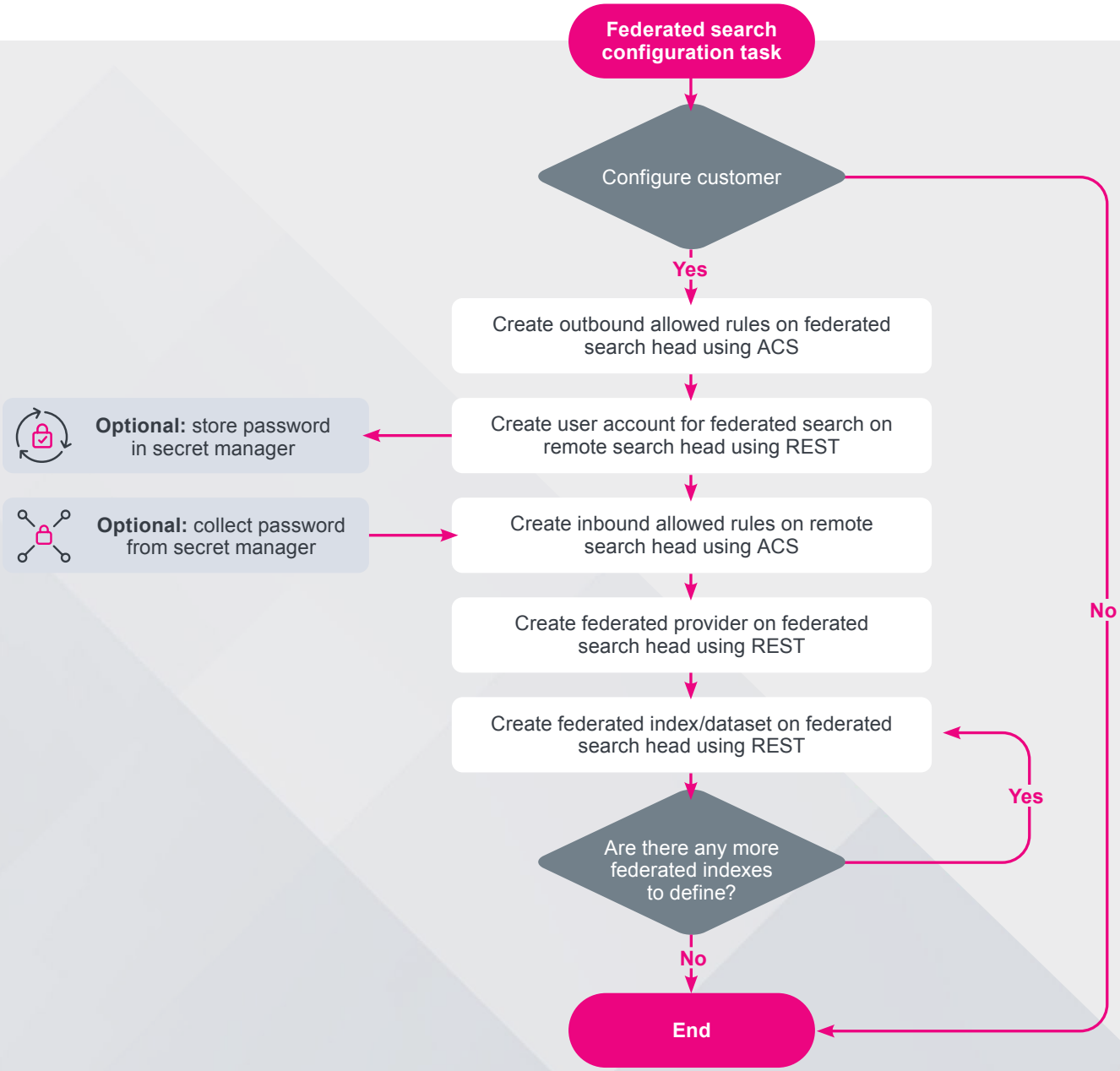


Optional automation

Automating the setup of federated search between the MSP Splunk Cloud Platform instance and the many customer environments will utilize both Splunk Admin Config Services and the Splunk REST API. Automation could perform programmatic creation of the relevant configuration so it is repeatable and all federated datasets will be configured to the same naming standard, which will make usability easier for the MSP admins and analysts.

Note: MSPs will need to ensure that the environments allow access on the management port from the automation system via the IP Allow list functionality before performing this automation.

This automation would be looped through for each customer and could follow this flow.



Role-based filtering and federated search

Role-based filtering is a new feature introduced to provide an extra layer of control around who can see sensitive information. It allows MSPs to apply role-based access to elements of an event so members within a specific role can see the unobfuscated event, while people not in the role can only see the obfuscated version of the event. This feature could be applied to the service account on the customer's Splunk Cloud Platform environment that federated search is using to prevent the MSP environment seeing certain personally identifiable information (PII) when they are not directly connected to the environment.

Example
Sample event:

```
Audit:[timestamp=11-29-2022 09:19:16.986,
user=darrend, action=search, info=granted
REST: /search/jobs/1669713550.68148/timeline]
```

It may be necessary to prevent the MSP Splunk Cloud Platform environment from being able to see this username in the raw event. Settings can be added to the role associated with the service account being used for federated search on the customer environment:

```
authorize.conf:
[role_msp-fs-svc]
fieldFilter-_raw = s/user=[^ ]+
user=REDACTED,/g
```

As a result of this filtering when accessing the event over federated search from the MSP Splunk Cloud Platform environment, we see:

```
Audit:[timestamp=11-29-2022 09:19:16.986,
user=REDACTED, action=search, info=granted
REST: /search/jobs/1669713550.68148/timeline
```

Documentation on this feature can be found here.

Note: The initial enablement of this feature currently needs to be performed via a support ticket; once enabled it can be managed using private apps and Admin Config Services.

Access control

There are several authentication mechanisms available within Splunk Cloud Platform; the most common are Security Assertion Markup Language (SAML) and local authentication.

The contractual agreement between the customer and the MSP might vary, and access control might vary depending upon their agreed roles and responsibilities, so we will cover this topic discussing both SAML and local accounts

SAML

SAML offers the greatest flexibility to the MSP and their customers.

For MSP users it helps to provide SSO which will support the experience of a single look and feel for the many Splunk Cloud Platform stacks making up the MSP solution.

Currently Splunk Cloud Platform only supports a single SAML identity provider (IdP). This can create challenges when the customer and the MSP both want to use their own SAML provider on the same customer search head. Many existing Splunk MSPs have solved this challenge using SAML federation, which most SAML providers support. This means the MSP or the customer could use their SAML provider and federate the other's access.

There is also a Splunk Ideas entry for multiple IdP support that is currently in the "planned" state, which should mitigate the need for federation between providers in the future.

Local accounts

If the customer wants to use their SAML provider and it is not possible to federate the MSP's accounts via that provider, then the MSP might chose to use local accounts. MSP users will lose some of the unified look and feel of the platform because they will not get access to SSO for the MSP analyst as they move between the MSP stack and the subordinate customer stacks.

It is possible to automate CRUD operations of local accounts across many customer environments, which would make this easier to manage, as explained subsequently.

Optional automation

SAML automation

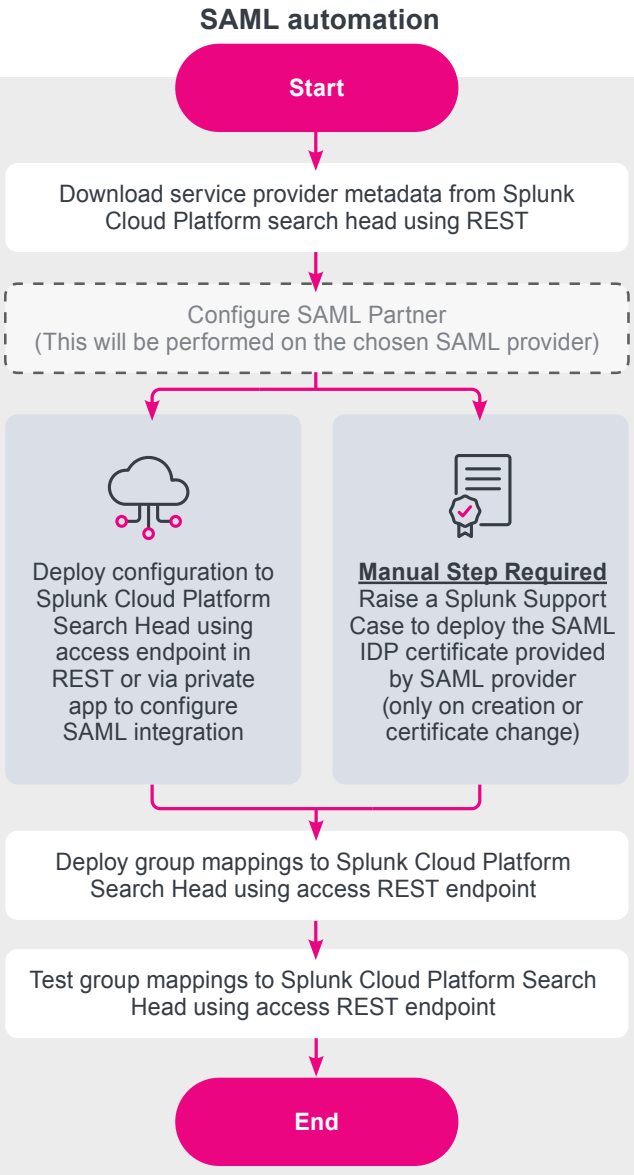
The automation of the creation and ongoing maintenance of SAML configuration will use the Access REST Endpoint within Splunk Cloud Platform. There is currently a manual step required in the initial creation (or certificate update), and investigation is underway on removing this manual step, which will be covered in future releases of this document.

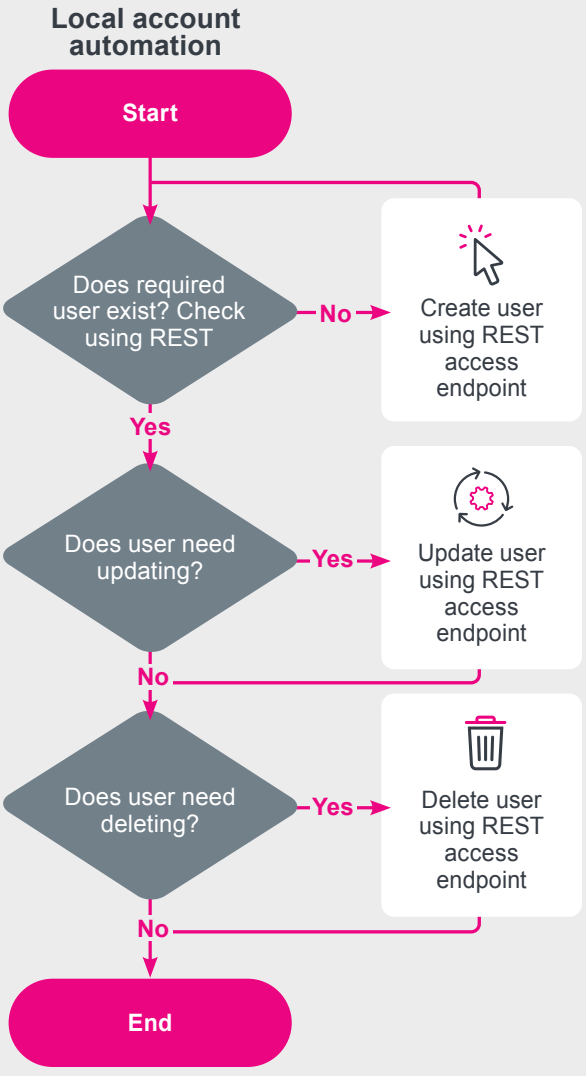
Automation of SAML configuration also relies on supporting automation mechanisms being available from the chosen SAML provider as there are initial configuration steps that need to be executed on both Splunk and the SAML provider, such as uploading the MSP metadata and downloading the relevant configuration to populate in Splunk via REST.

SAML configuration (with the current exception of the IdP certificate) could also be deployed using a private app via REST; however, this would have lower priority under the Splunk order of precedence and as a result Splunk would recommend REST for this task so that it is written to `$SPLUNK_HOME/etc/system/local/authentication.conf`, which has the highest priority.

Local account automation

Splunk Cloud Platform supports CRUD operations on local accounts using the Access REST Endpoint, which makes automation of user account activities relatively easy. In the event that SAML cannot be leveraged one might choose to use local accounts.





Knowledge object management

When working with multiple environments MSPs are unlikely to want to manage knowledge objects via the UI on each stack as they would suffer very quickly from configuration drift. They might also want different configurations to be deployed to different environments, which further complicates the challenges of managing knowledge objects. For example, one might want customer and MSP environments to run with a different set of knowledge objects.

To mitigate this challenge, MSPs will want to place our configuration into discrete groups of apps and addons. They would likely want to use standardized naming conventions so that they know what is in these addons.

A common approach to standardizing naming is used in Enterprise Security and is described in About the ES solution architecture.

As a result of putting knowledge objects into apps and addons MSPs are more likely to be working with configuration files. They will need to work out a process of determining which apps/addons should be deployed to each environment.

Creation of apps and addons is well documented and not covered further in this document, but a useful reference on this topic can be found here: Lifecycle of a Splunk app for Splunk Cloud Platform or Splunk Enterprise.

Once we have our apps/addons created, we can install them to the relevant MSP and customer environments.

Limiting knowledge object management

This model limits the configuration of knowledge objects using the UI to make the platform scalable and easier to manage. As a result MSPs do not want most of the user base to share objects at the app or global level. There are typically two ways to achieve this, with a process control or with a technical control.

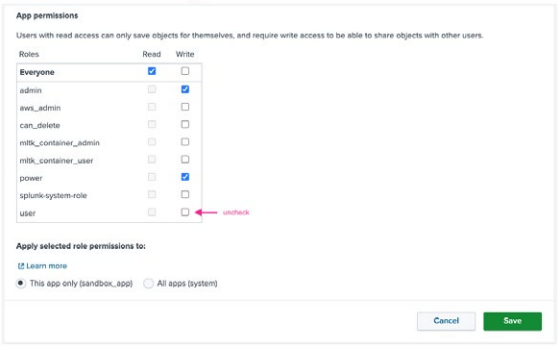
The easier option is to implement a process control.

Process control

Create a business process that states users should not change any knowledge objects permissions or edit existing knowledge objects shared at the app or global level in place via the UI.

Technical control

Change the permissions of all deployed apps and addons to remove the write ability for the relevant roles. This will allow users to create private knowledge objects, but not to promote them to “app” or “global.”



Developing knowledge objects

MSPs still want users to be empowered to work on and develop content on the platform. However, they want to have control over the knowledge objects that are in use. There are two typical approaches that could be adopted.

Approach 1

This method relies on a separate environment for development, which might be a sandbox or temporary environment running in a development environment or developer's laptop and will be running Splunk Enterprise.

This approach is effective for creating and updating content, but thought may need to be taken on how adequate testing might be performed. This approach also minimizes the potential for accidental changes as we will not be altering knowledge objects directly in production.

As users will likely have access to the underlying filesystem, it will also make it easier to gather knowledge objects in preparation for deployment to the MSP and customer environments.

Thought will need to be taken on how to provide representative test data to these separate environments, and in some cases search development might occur in production environments using the controls already discussed to ensure representative data for prototyping.

Generally this approach offers a lower skill level for new MSP admins and as such once created is more robust.

Approach 2

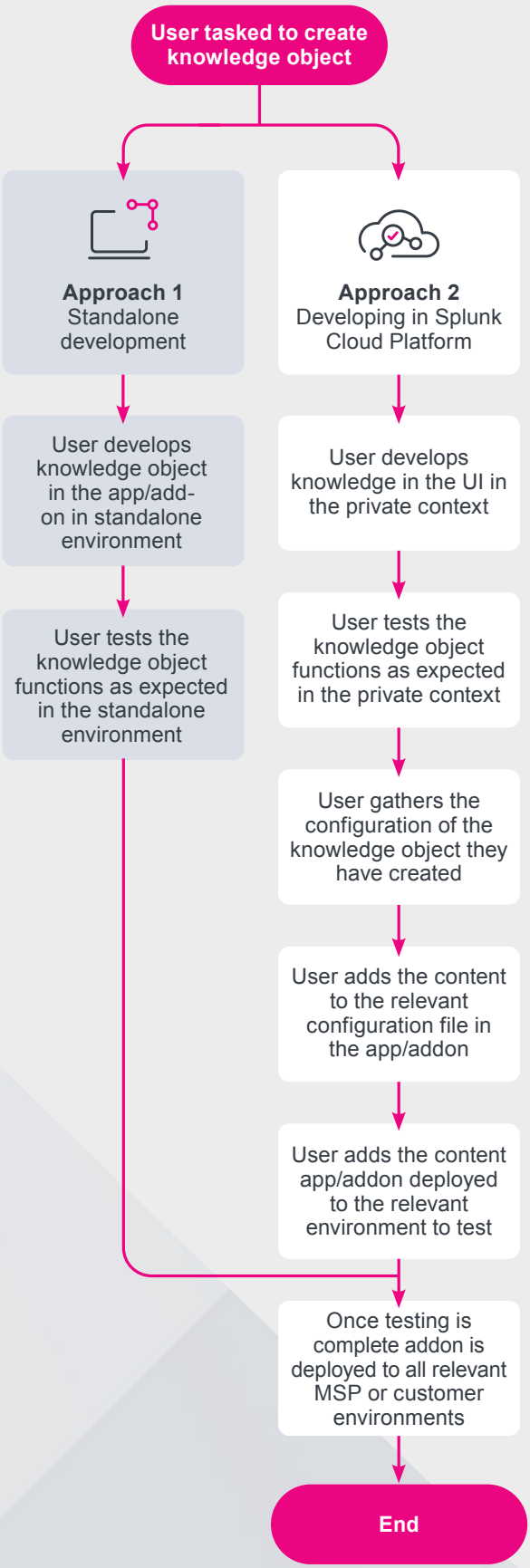
This method allows creation of knowledge objects directly in the production environments. It allows users to create knowledge objects and dashboards in their private user context directly in the environments but limits their ability to share those objects at the app or global level (as discussed in the preceding content on controls). MSPs can then take those changes and add them to their apps/addons to be deployed to the relevant customers.

Optional automation

Managing knowledge objects across the multiple Splunk Cloud Platform environments in a robust, scalable and repeatable way is the perfect use case for automation.

Knowledge objects will be grouped into apps or addons and will either be deployed from Splunkbase or as a custom app.

The deployment of these apps or addons is performed using ACS.



Splunkbase apps

MSPs can automate the CRUD operations of Splunkbase apps easily using ACS. They simply make REST calls telling ACS which cloud vetted apps to install, and they will be installed onto the Splunk Cloud Platform environment within a couple of minutes (Admin Config Service Manual).

Local changes to Splunkbase apps

In an on-premise deployment of Splunk Enterprise, we would historically have solved this challenge using modifications to the addon that we deploy to the local folder of the same app or addon. This is typically performed when tuning a Splunkbase app or addon to enable or disable specific functionality.

ACS does not currently support local changes being made to a Splunkbase app in this way. MSPs are still able to make these changes in a supportable way by leveraging the Splunk order of precedence of apps (How app directory names affect precedence).

Changes should be deployed to apps with a similar name that are more important in the order of precedence. This way customizations will be applied rather than the original settings in the Splunkbase app or addon.

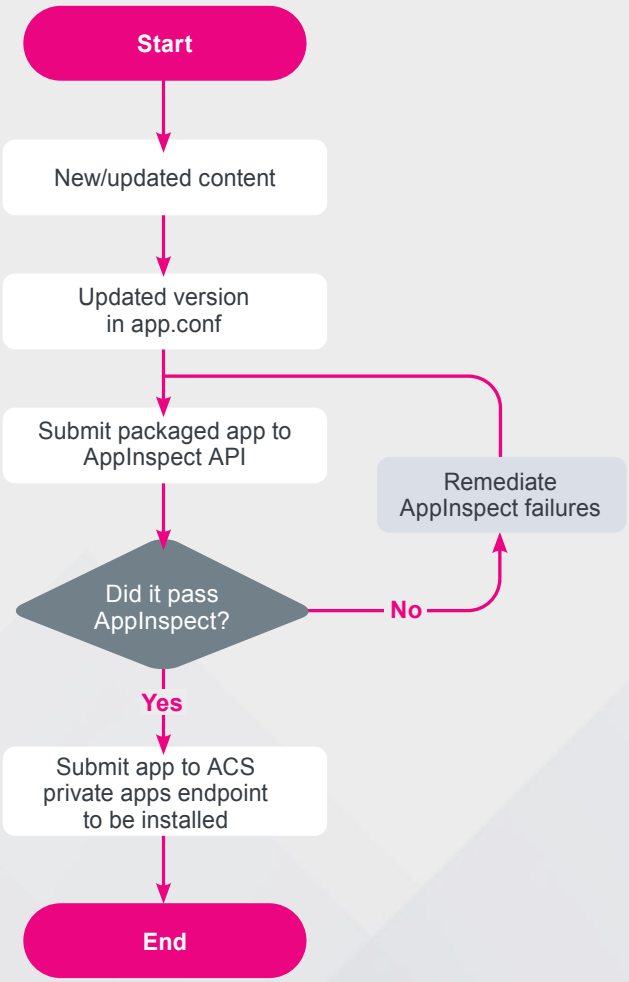
The app containing the customizations could then be deployed to the environment using the ACS private app management endpoint.

Private apps

Private apps is the term used when talking about any app that is not downloadable and ready to be used directly from Splunkbase. These would typically be:

- Content developed by or on behalf of the customer
- Apps/Addons obtained from a source other than Splunkbase, such as an app/addon that was supplied directly by a another vendor
- A Splunkbase app/addon that was not cloud vetted that needed significant customization to be allowed in Splunk Cloud Platform
- An app containing the customized contents of a local directory as a default directory when a customer has migrated from on-premises to Splunk Cloud Platform.

These apps/addons should go through a process called app vetting and can then be installed on Splunk Cloud Platform via ACS (Vet apps and add-ons for Splunk Cloud Platform).



When automating AppInspect into your workflows, we recommend running AppInspect against any private apps that have changed in that branch before the merge request that will attempt to deploy the changes is run. This will ensure that AppInspect issues are caught early and will minimize any potential issues just before deployment, even though it is possible that you might still push the apps through AppInspect again during the deployment.

Macros

In this configuration the goal is for analysts to have a uniform experience when interacting with the platform. When accessing datasets across many customers from the MSP environment, users should not have to type in every customer environment; equally wildcards should not be used excessively. MSPs also don't want to increase the amount of customer specific knowledge expected of analysts before they can support the platform.

A great solution to this challenge is macros (Knowledge Manager Manual).

Administrators can abstract the customer datasets that they might want to access en masse into macros so that the analysts and users utilize them in their searches, rather than having to type complex queries.

As an example, rather than a user having to write a search similar to:

```
index=federated:cust1_internal OR
index=federated:cust2_internal OR … |stats count
```

They could simply run:

```
`cust_internal_indexes`|stats count
```

The macro would ensure the user didn't have to know information about the name of every customer, they just need to know that the macro `cust_internal_indexes` provides them access to the `_internal` index of every customer in parallel.

Optional automation

These macros can easily be programmatically created/modified by automation every time you onboard a new customer.

In the automation MSPs will know which federated datasets exist and which customers they have. They can use this information to programmatically build a new private app that contains the relevant macro definitions. This app would then be deployed via ACS and automatically updated as new customers are added/removed or when new datasets need this level of abstraction.

App permissions

In Splunk we often just leave app permissions at their default settings and use the global sharing option to allow content to be merged across multiple apps. This works well for most scenarios.

However, there are scenarios which need to use more granular access control on apps to meet specific scenarios.

Limiting app access to specific roles

RBAC can not only be applied to indexes and capabilities in Splunk, it can also be applied to apps. This might be to limit the access of an MSP specific dashboard to only colleagues from the MSP, or limit access to an app specific to web application logs to the web application team.

Details on configuring more granular permissions in your apps are covered here: Set permissions using RBAC in Splunk Cloud Platform or Splunk Enterprise.

Addendum

Automation recommendations

Using third-party automation tools will significantly improve the quality and ease of management. It will also allow you to identify what changed, why, when and the user that made the change, which are useful things to track from a compliance perspective.

We recommend validating the current state of an environment and only applying necessary changes to match your configuration in version control. We do not recommend attempting to apply every configuration in your version control system every time you perform a release.

When working with private apps, we recommend only deploying apps that have new changes in the feature branch that is being merged and/or installing apps that are not currently installed in the target environment.

We do not recommend attempting to install every app on every release as some apps might trigger a restart of the platform and we want to manage the deployment of those apps.

Automation workflows

While GitHub flow and GitFlow provide approaches to managing your configuration, you might also have a flow within your organization that you would prefer to work with.

For the remainder of this section we will use GitHub flow as our workflow, which might look like the diagram below.

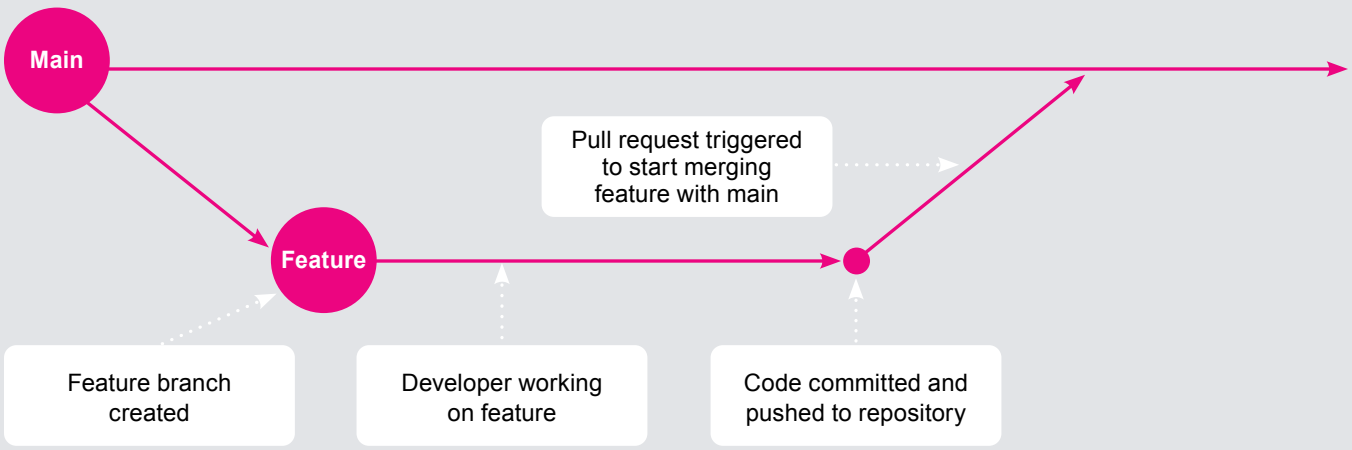
Try to keep your automation workflows relatively efficient. Generally, if the workflow takes longer than 5 minutes to complete, then it might hinder rather than help your content developers.

Automated testing and validation

Automation allows you to enforce your defined standards easily and also to perform syntax checking proper to testing.

Some of the automation that you develop as an MSP might validate the syntax and naming standards of the changes that have been made before they are tested and eventually deployed, while other parts of your testing might perform more functional testing.

When developing validation and testing, we would not recommend “boiling the ocean.” There are some tests that are sensible to start with, but others might be based on challenges you have encountered. When you have an issue in the future a good practice might be to learn from it and build a test so that it cannot happen again. It is also sensible to review failures to ensure a test is still valid and tune it appropriately over time.



Naming conventions

Naming conventions are important at scale. However often developers don't follow them quite as strictly as they should. This can lead to confusion in your user base since they are relying on those conventions. With automation we can add checks and fail changes that don't adhere to your documented standards.

We recommend enforcing naming conventions early. This could be performed as early as when the developer pushes their feature branch before any further testing is performed.

Code validation

One of the useful features about adding automation to your testing and release processes is that we can check for syntax errors. This could be as simple as validating any JSON you are using in your repository is actually JSON and doesn't have syntax errors before proceeding.

This is another check that you could perform prior to any functional tests being performed.

AppInspect

When making changes or creating a new private app, we can leverage AppInspect to get a view of changes that might need to be made prior to testing and deployment. We can identify changes to private apps that have occurred in our branches and push them through AppInspect early. Upon failure of AppInspect, we can parse the output and provide feedback to the developer on what they need to change to be successful.

Testing

Some kind of testing will probably be part of your process before you decide to release changes to your MSP stack and your customers.

There are two points when we are likely to perform testing. You will need to decide the appropriate point(s) for the test(s) you wish to perform.

Push of feature branch

This is likely where you will be performing some of the checks already like initial AppInspect, naming standards and code validation. This is a useful place for performing isolated testing like using an instance of Splunk Enterprise (maybe the docker container) to check for runtime errors or similar.

Pull request with main branch

When we are ready to commence final testing and deployment, we will create a pull request. This is where we are starting the process of merging our feature branch into production.

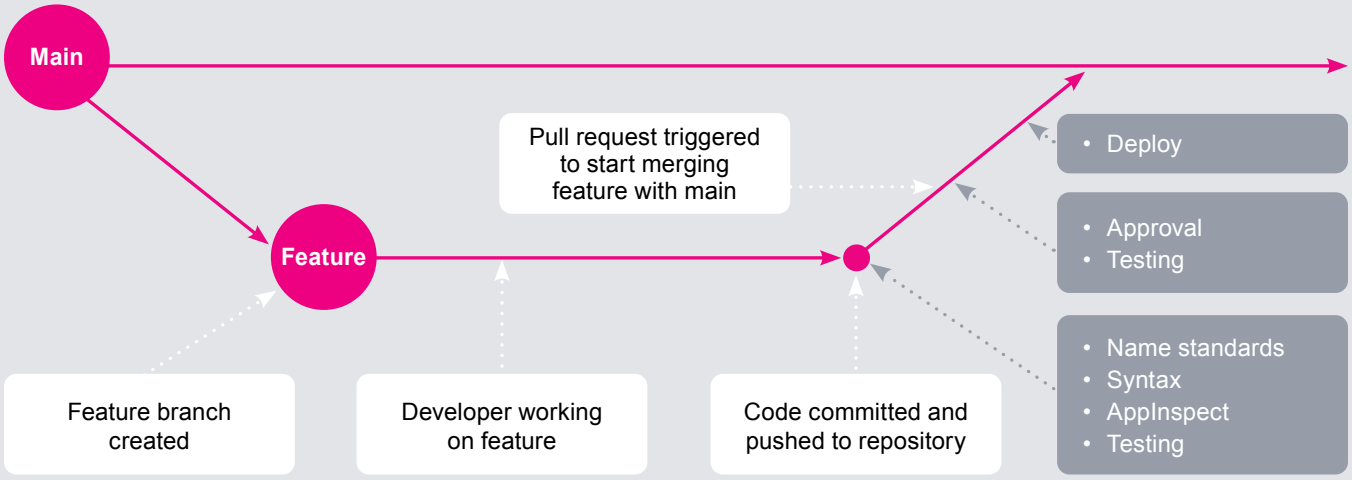
This is also where we can perform automated testing and approval steps for code review, change management approval or similar.

Your automated deployment of configuration is likely to form part of this pull request testing, so we might want to perform some checks at this point too. Checks that might be appropriate to perform are:

- Check that the customer environments are available and healthy
- Check you are able to authenticate with ACS, Splunkbase and, in the case of using REST, the search heads to which you will be connecting

Once our testing has successfully completed and all approvals have been achieved, we can merge our pull request. At this point we will be deploying our change to production.

The preceding steps could be represented on our GitHub flow diagram below.



Programmatically generated configuration

Some configuration might be the same no matter the stack you are trying to deploy to and might be programmatically generated and applied if needed during the release.

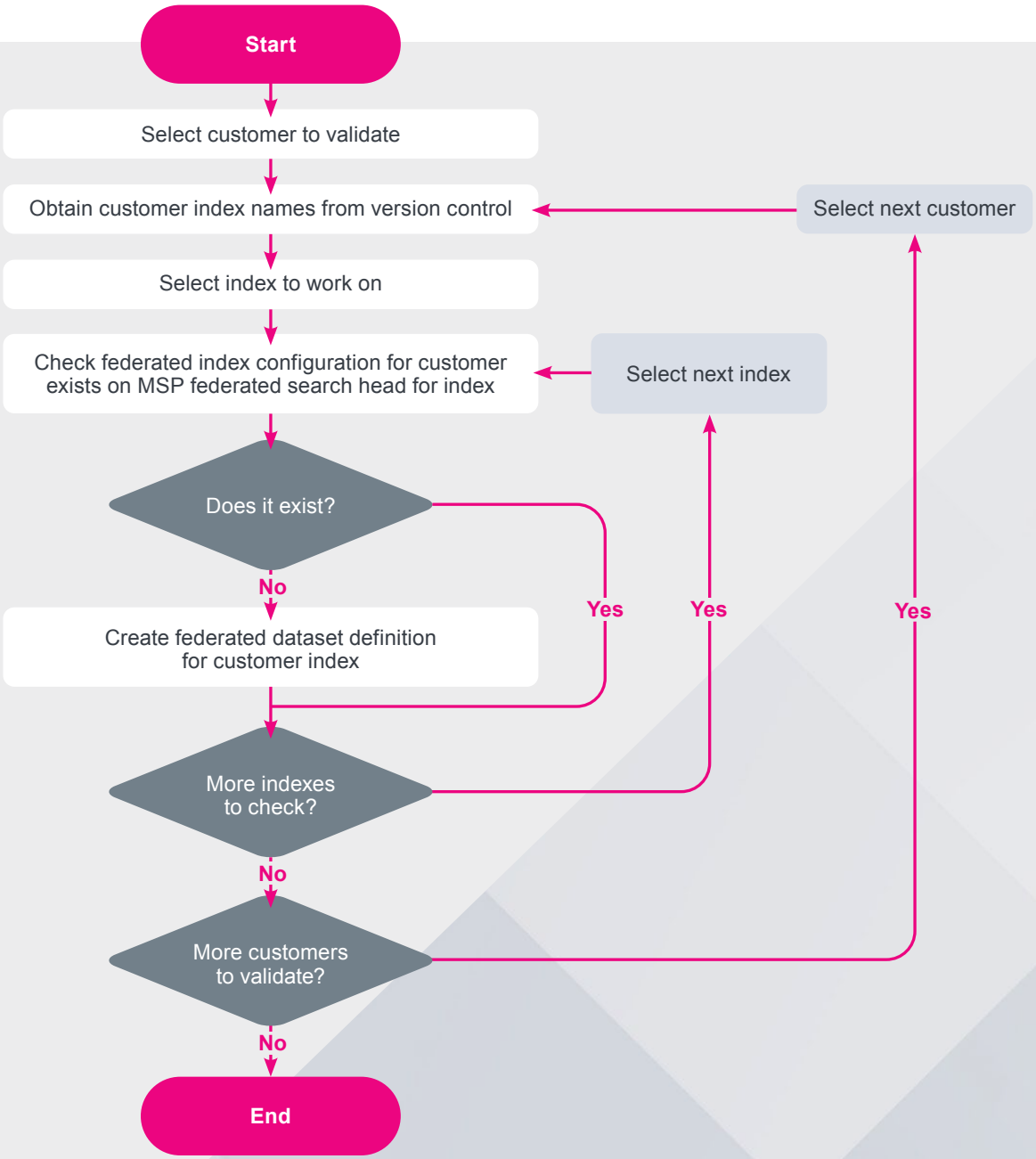
An example that might be appropriate for this is the configuration of federated search.

Example of configuring federated search and federated datasets using programmatically generated configuration

You are likely to define indexes for your customer environments in your version control. These might be the same for every customer, or they might be different. We might even only want to configure access to certain indexes. For this example we will keep it simple: every index configured for the customer in version control will be accessible from the MSP stack over federated search.

Since we will be defining the index names in our automation, we can use that not only to push the configuration to the customer environment but also to iterate through and configure the MSP's federated datasets.

In your pipeline you can create tasks to perform the following steps for each customer:



In an optimal automation scenario this means that when a developer adds a new index to version control they should only have to add that to a single place. When the automation is triggered, it should create all relevant configuration related to that index:

- Create index on each relevant customer via ACS.
- Update role permissions to include the index if you are restricting index access to the federated search service account.
- Create a federated search dataset for each customer for the new index so it can be used by the MSP analysts.

Overlaying this to our GitHub flow diagram would resemble the chart below.

Securing your repository

The repository is our single source of truth to how a customer is configured. It is almost the most important intellectual property you have about your MSP offering, and it needs to be protected appropriately. You need to ensure that change cannot be introduced without going through your appropriate processes. There are technologies and processes that help us.

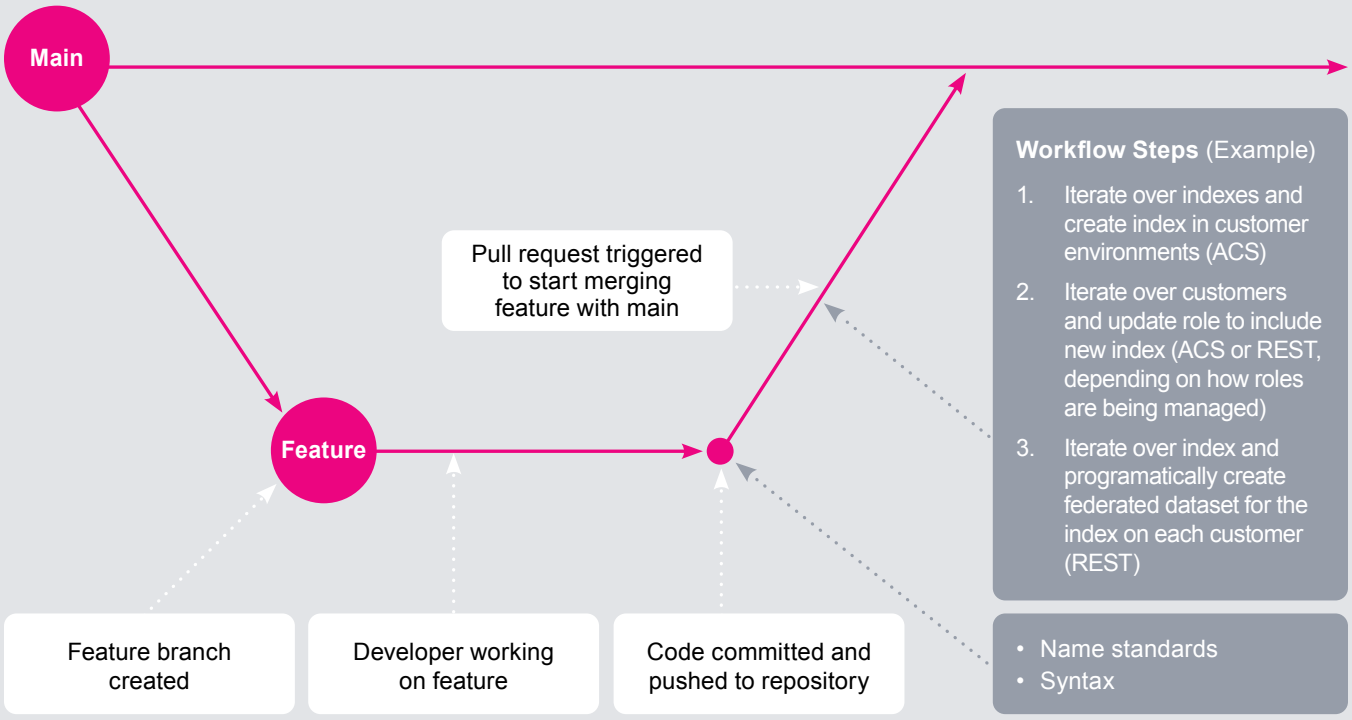
Branch protections

Branch protections are strongly recommended to prevent changes being directly made to your main branch. Without branch protections, an actor might be able to circumvent your processes and apply changes directly to the branch that is in production use. Protecting your main branch should be the minimum objective with this feature.

Approvals

We should have some level of control around approving changes to your platform, this could be a simple "four-eyes process," where when you create the merge request another member of your team has to review and approve to make sure it's appropriate before it is deployed. Alternatively, it might be more in-depth integrating with your organization's change approval processes.

It is common practice that at least two people are involved in a change, one developing and one approving, to reduce the risk of rogue or inappropriate change being deployed.



Managing a customer with multiple search heads/search head clusters in an environment

A standard Splunk Cloud Platform environment will be provisioned with at least one search head or search head cluster. Some customers may choose to utilize one of Splunk's premium products like Enterprise Security or ITSI.

When a customer has one of these premium solutions, this will be provisioned on a separate search head or search head cluster.

In environments with multiple groups or search heads within a single customer environment, we have some additional complexity we need to manage.

Federated search

Typically we will interact with either the ad hoc search head (or search head cluster) or the premium app equivalent in a customer environment, depending on the intended use case you are delivering for the customer. You will need to make this choice when you configure the environment. You could change this later if the type of service you are offering has evolved. The search head you choose to connect to will be defined by the service that you are offering to your customers.

Private app management via ACS

When you deploy apps to Splunk Cloud Platform using ACS, you may not choose where they are deployed. This decision is made for you by the SaaS. This means that deploying apps that contain enabled saved searches might cause those saved searches to execute on both the ad hoc and premium search heads.

To mitigate this, we should handle our saved searches slightly differently in our automation in this scenario.

Option 1

1. Use a naming standard for private apps that helps to identify where their enabled saved searches should be deployed, for example es_my_private_app.
2. Create content as normal in this app within version control.

In the automation, we will embed logic to accommodate the challenge:

1. Disable all of the saved searches in savedsearches.conf (but keep track of what they were).
2. Package the app.
3. Pass it through AppInspect.
4. Deploy the validated app via ACS.
5. Use REST to selectively enable the saved searches within that app that need to be enabled on the relevant SH (or customer).

Option 2

This option is similar to Option 1; however, rather than using a single savedsearches.conf, we will maintain two savedsearches.conf, one of which is the default to be deployed, and a separate savedsearches.conf.es that contains the Enterprise Security specific state of the savedsearches.conf.

This option has the advantage that we are not disabling every saved search in the app when we deploy it.

- Create content as normal in this app within version control. Saved searches in <app>/default/savedsearches.conf should be disabled if they are only needed on one of the search head groups, or enabled if they should run on both. We should create a separate file called savedsearches.conf.es.

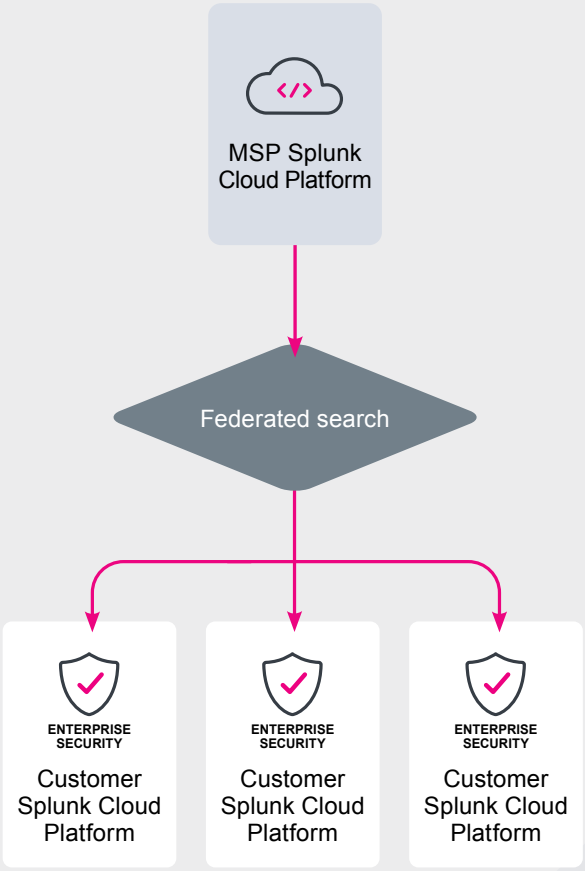
In the automation, we will need logic to accommodate the challenge:

1. Package the app.
2. Pass it through AppInspect.
3. Deploy the validated app via ACS.
4. Use REST to selectively enable the saved searches within that app that need to be enabled on the relevant SH using the extra conf file we added.

Splunk Enterprise Security

In this model, we can easily accommodate Splunk Enterprise Security (ES) within any of the customer Splunk Cloud Platform environments. You can interact with each Enterprise Security instance using federated search. You can aggregate alerts on the MSP environment that are generated by each customer environment's Enterprise Security instance.

We do not currently support Splunk Enterprise Security on a federated search head, and as such you would not be able to use it on the MSP search head to investigate your customer environments.



This will mean that you will be able to aggregate notables into a dashboard on the MSP environment. Analysts, however, will need to navigate to the customer's Enterprise Security instance to perform investigations.

Splunk Enterprise and bring your own license

This blueprint has been focused on Splunk Cloud Platform, but it is transferable to Splunk Enterprise if you are running an MSP offering in a public cloud instead.

The only changes that need to be made are regarding Admin Configuration Services (ACS). This is a capability specific to Splunk Cloud Platform, providing you access to backend services within the Splunk Cloud Platform service. You will need to map the functionality that ACS is providing to the features available in your chosen platform (REST, file access, cloud service provider APIs, etc.).

Hybrid model (Splunk Enterprise and Splunk Cloud Platform)

A hybrid model where you have some customers using Splunk Cloud and others using Splunk Enterprise might be required. This is completely possible as federated search supports this.

However, this model will increase the complexity of your configuration management because the deployment methods to push configuration will be different depending on which variant of Splunk the customer is utilizing.

MSP feature limitation matrix

Feature	Common requirement	Current functionality	Evolution for the MSP
Multiple authentication method support	MSPs often want to have access to the customer's Splunk Cloud Platform instance as well as allowing the customer access. Sometimes the two parties want to use different SAML IdPs that do not easily federate with each other. Splunk currently supports only a single SAML provider per Splunk Cloud Platform environment.	1. One of the parties federates access for the other. This is common when both parties are using the same IdP. 2. One party leverages an alternate method of user integration such as local accounts. Local accounts can be managed programmatically using REST to perform CRUD operations. REST API Reference Manual	Feature changes requested and marked as planned for multiple IdP support.

Multi-stack architecture vs. single-stack architecture

In this architecture we are promoting using multiple Splunk Cloud Platform environments to provide a single look and feel to the platform being delivered. This aims to meet the needs of the MSP and their customers. Often there is an expectation of a single stack architecture serving multiple customers. The following matrix of key features along with a comparison of challenges created or solved by the two approaches should hopefully highlight why we recommend this model.

Requirement	Single-stack	Multi-stack
Data sovereignty • Support for the increasing number of data governance requirements in different countries	A completely new stack would need to be deployed every time you sell to a customer with different data sovereignty requirements. This eventually ends up with multiple MSP environments and still a need to bring them together with a single umbrella instance. That is, in the end, this architecture resembles the blueprint anyway.	The customer stack can be deployed to whichever Splunk Cloud Platform region meets the customer's needs while the MSP stack can be anywhere.
Data and access separation	Complex to manage and doesn't scale well: • Risk of running into service limits the more customers are onboarded • Very difficult to ensure all customer artifacts are segregated	Each customer is in their own stack, so they can never access another customer's data.
Search	Searches all must have an extra dimension added to them to filter by the customer. • This can slow the searches because it increases search complexity and can also increase compute usage significantly. • It increases the required skill level of the MSP's developers.	Searches are run within the confines of the customer's Splunk Cloud Platform environment. • Searches are simpler and as a result will use less compute. • A lower skill level is required before MSP developer can be effective. • Out of box content such as ES content updates or Splunk Security Essentials will require less customization before it can be leveraged.
Attribution of workload (How much of the workload should be billed to each customer?)	Customers have a high probability of influencing each other's workload. • This makes it difficult to work out how much compute to bill to each customer beyond the basics of you ingested x GB.	Each customer's workload is isolated and run within their own environment, so it is easy to calculate how much a customer should be billed. • No customer has the ability to affect or skew another's workload.

Third-party tools for automation

You are likely to have a chosen set of tools to automate the processes that have been talked about in this documentation. If you do not currently have automation tools in your organization to meet these needs, then you can investigate the following options.

Version control	Secrets management	Automation CI/CD	Notes
GitHub	GitHub Secrets	GitHub Actions	The Secrets management is basic, but you can perform all three of the required activities using this too. You might want to leverage a dedicated method of secrets management alongside the basic functionality provided by the tool.
GitLab		GitLab	Common version control and CI/CD tool, often leverages external tooling for secrets management
	HashiCorp Vault		Dedicated secrets and sensitive data management that can be integrated with a wide variety of tools
		Jenkins	Common CI/CD tool
Azure Repos	Azure Key Vault	Azure Pipelines	A tool suite available in Azure with many advanced features
Bitbucket		Bitbucket Pipelines	A tool suite available from Atlassian. Secrets managements plugins are available including integrations to HashiCorp Vault.

Functional examples

Customer monitoring use case

There will be times when you want to monitor or perform a specific use case from the MSP stack. We could run this as a search from the MSP environment where we search all stacks using a traditional SPL search.

However, we might want to keep as much of the processing as possible on the customer's environment and then only retrieve the results to the MSP environment.

This use case example shows how you could keep as much of the processing as possible in the customer environment.

In this example, we have deployed the use case in two parts: a saved search is deployed to the customer Splunk Cloud Platform environment and a corresponding search leverages that on the MSP Splunk Cloud Platform instance via federated search.

Customer component

savedsearches.conf:

```
[last_60m_ingest]
dispatch.earliest_time = -60m@m
dispatch.latest_time = -0m@m
search = | tstats p99(PREFIX(average_kbps=))
AS p99 first(host) AS host where index=_
internal sourcetype=splunkd TERM(group=thruput)
TERM(name=index_thruput) TERM(average_kbps=*)
by _time span=1m | rex field=host "[^\.]+\.
(?<stack>[^\.]+)\."|fields - host|table _
time,stack,p99
```

MSP component

Search in dashboard:

```
| union [
  from federated:cust1_last_60m_ingest
  ]
  [
  from federated:cust2_last_60m_ingest
  ]
| xyseries _time,stack,p99
```

Simple alerting use case

Much of the work you will likely perform for your customers will be through the use of saved searches to generate alerts that will be actioned by your internal resources.

In these scenarios we will ensure the workload is being executed in the remote environment and recorded there so that you can show your customers the alerts being generated. We will then access those alerts over federated search from your MSP Splunk Cloud Platform environment where you can aggregate all the customer alerts to perform further actions or to provide the start of a workflow for your analysts.

Customer component

The customer environment portion of the search will output information about the alert to the msp_alerts index as a summary indexing task.

savedsearches.conf:

```
[failed_logins_and_a_success]
action.summary_index = 1
action.summary_index._name = msp_alerts
action.summary_index._type = event
action.webhook.enable_allowlist = 1
alert.track = 0
cron_schedule = 14 * * * *
dispatch.earliest_time = -65m@m
dispatch.latest_time = -5m@m
enableSched = 1
search = index=_audit sourcetype=audittrail
user=* action="login attempt" info=*|
stats first(splunk_server) AS splunk_
server,count(eval(info="failed")) AS login_failed
count(eval(info="succeeded")) AS login_success
values(sourcetype) AS sourcetype,values(index)
AS index by user | where login_failed>=3 AND
login_success>0 |addinfo| rex field=splunk_
server "^[^\.]+\.(?<stack>[^\.]+)\." |rename
info_min_time AS earliest_time,info_max_
time AS latest_time,info_search_time AS
_time,host AS orig_host,sourcetype AS
orig_sourcetype,index AS orig_index | eval
savedsearch_name="failed_logins_and_a_success"|
table _time,stack,savedsearch_name,earliest_
time,latest_time,orig*
```

MSP component

The MSP component can aggregate all of the alerts generated from the customer environments.

Search example:

```
index IN (federated:cust1_msp_alerts
federated:cust2_msp_alerts federated:cust3_
msp_alerts)| table _time,stack,savedsearch_
name,count
```

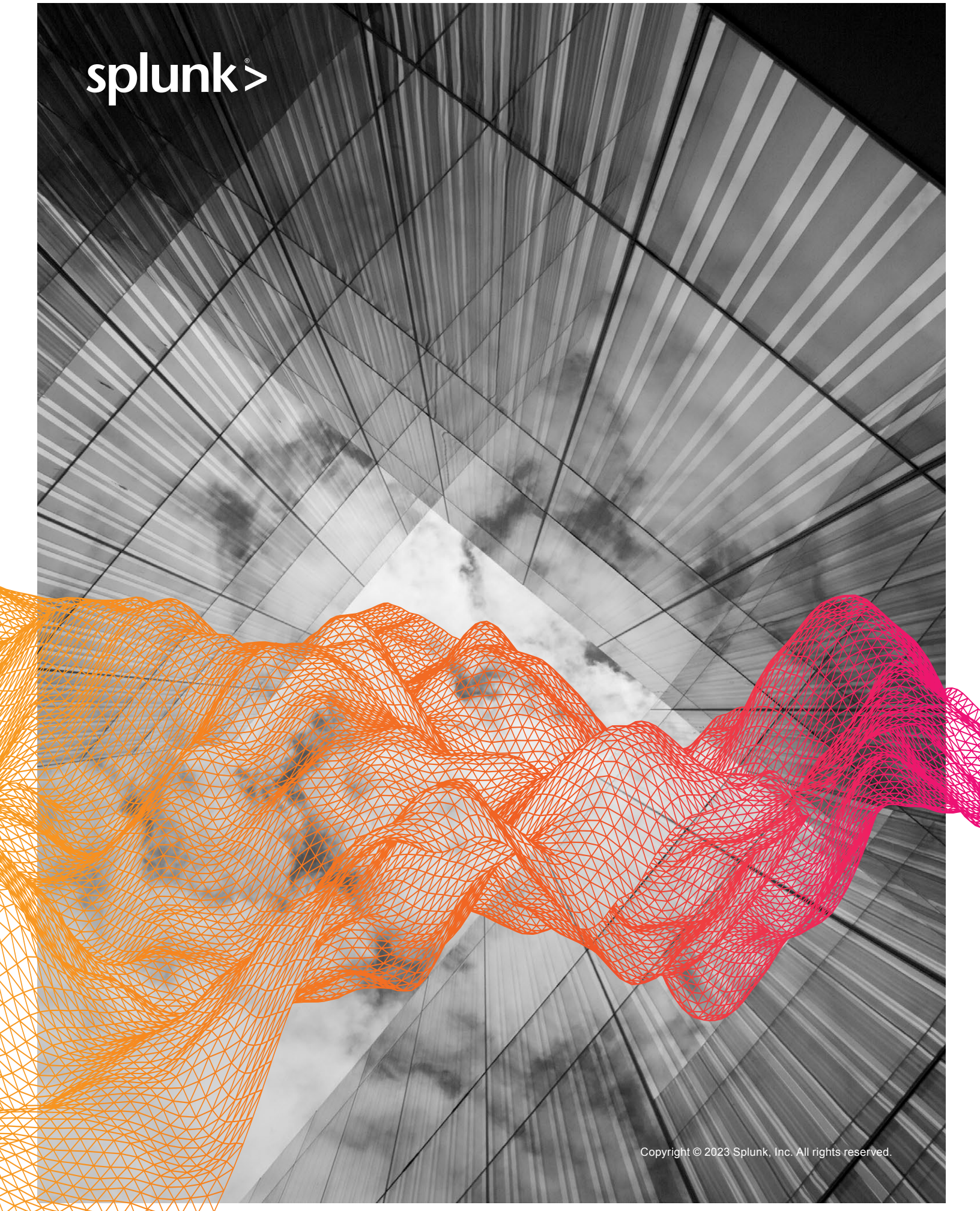
More questions?

For questions, reach out to Splunk at msp-partner@splunk.com or **+1.866.GET.SPLUNK** (1.866.438.7758)

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.

Learn More

splunk>


splunk>
Copyright © 2023 Splunk, Inc. All rights reserved.