MANAGEMENT AND CONFIGURATION

Deployment Type

One of Splunk's (many) strengths is the ability to scale from the desktop to large global deployments with the same binary.

To make data usable, accessible and valuable to everyone, as well as scaling massively, Splunk Enterprise can be split across multiple instances each specializing in one of these three functions. First it collects the data (**forwarder**), second it indexes the data (**indexer**) and third it runs searches on the data (**search head**). These functions map to four different phases: Input, Parsing, Indexing and Search.

This strength can be a source of confusion (and if not checked serious performance problems) as users grow and want to scale their Splunk implementations to index more and more data. Splunk in its current state offers very different administrator experiences when growing from a single indexer/search head instance to a multi-tiered data platform.

Single-server

- One instance of Splunk Enterprise installed, acting as both a search head and an indexer.
- Content easily installed from the GUI.
- The Splunk instance may collect the data locally, and/or a forwarder(s) may be deployed to collect the data.

Distributed

- Splunk Enterprise is tiered onto at least two servers—one is a search head and one is an indexer.
- The search head must be told to what indexer(s) to look on for its data.
- Content is not installed via a GUI but from a central place to the search head and the indexer like single-server.
- To accomplish the above, a configuration management tool is used or the installer must copy configuration files to each server.
- These deployments are usually much larger with multiple indexers and potentially multiple search heads.
- Predominantly have forwarders collecting the data.

Types of Splunk Content

This ability to scale from single-server to distributed environments with a multitude of topologies is extremely powerful, but it presents a challenge for developers of Splunk content. For example, there are many Splunk deployments that never go beyond-single server. Based on their use case and/or data amounts, one Splunk instance does the job. For larger deployments data collection, indexing and search are distributed across multiple servers. Splunk content containing the configuration files associated with those functions need to go on the appropriate tier. Splunk apps and add-ons are a way to group those configuration files and ease their installation on the various tiers. See: http://docs.splunk.com/Documentation/Splunk/latest/Admin/Configurationparametersandthedatapipeline

Apps

At a high level, you can think of an app as a workspace that solves a specific use case. An app can extend Splunk Enterprise with new navigable views that report on particular kinds of data, can provide tools for specific use cases and technology, and are often developed for a specialized user role. Apps are made up of knowledge objects and configuration, anything from custom UI to custom input scripts. One of the key differentiators of an App is that it contains a navigable view(s).

Add-ons

An add-on is a reusable Splunk component much like an app, but does not contain a navigable view. Add-ons can include any combination of custom configurations, scripts, data inputs, custom reports or views, and themes that can change the look and feel of Splunk Enterprise. A single add-on can be used in multiple apps, suites, or solutions. They can reside on either or all tiers (data collection, indexer, search).

Mapping to Splunk Phases

As mentioned above, the process of ingesting data into Splunk (also known as the data pipeline) consists of four phases: Input, Parsing, Indexing and Search. This must be kept in mind when developing content that is to be deployed in a distributed environment. To help with this, add-on grouping is employed:

Domain add-ons provide the dashboards and views for each domain within the app.

Supporting add-ons provide the underlying support modules and tools leveraged by domain add-ons (saved searches, macros, and so on).

Technology add-ons provide the feeds to get data from different sources, and search-time knowledge maps to normalize the data for use within the app. This includes modular inputs and scripted inputs.

While there is not a direct mapping of the add-on types above to the data pipeline phases, the grouping can be helpful. The target that we are trying to hit (and it is not always an easy one), is to make content installs simple for single-server as well as distributed deployments, the latter being the most mistakes prone. See:

To make things less confusing for the Splunk administrator's who presently or at sometime in the future will manage a distributed environment, content packages should be broken down into the following categories:

Input
Data mapping
Visualization.

Installs made easy: Packaging

When working with a single-server deployment, the standard Splunk package (file type .spl) works for both Windows and non-Windows platforms. When it comes to distributed Splunk deployments, some sort of configuration management tool is introduced. Many times it is the Splunk deployment server. When this is the case, administrators have to work at the file and directory level. It is here that Windows users need a zip file, since they may or may not have access to tools like WinRAR, due their company's software download policy. Splunk can make that easier by offering both .zip and .spl file packages for content that is platform agnostic.

Minimum requirement: Splunk downloads will be offered in zip and spl.

Installs made easy: Dependency declarations and checks

The Splunk developer ecosystem is all about extending Splunk. Many times that extending requires some additional software. At minimum dependencies should be documented. At a minimum, a CLEAR AND PRECISE list of dependencies is a requirement.

Minimum requirement: List all dependencies

Installs made easy: Storage implications

Indexes

Some apps create an index, while others leave it to the users to decide. The administrator needs to be informed of this upfront. will be added to the info about the app or add-on, along side of CIM compliant, Splunk version supported, etc.

When it comes to creation and management of Splunk indexes, developers need guidance and their users need guidance especially in the realm of a distributed Splunk implementation. A new index has two requirements-- access control and retention rates. Apart from that there is great freedom. Currently, there is no GUI that allows for administering an indexes.conf across a distributed system.

Minimum Requirement: Document any index creation

Summarization

Splunk summarization offerings are discussed in greater detail later on in this document. As is known, the data associated with summarization has to be stored somewhere. These are extra directories that will be added to the index directory structure (summary and datamodel_summary by default). If report acceleration or data model acceleration is in use, the administrator needs to know about it up front. Since these are the result of searches what can vary greatly, original index size estimations may be insufficient.

Also, tscollect is still in the product and still puts tsidx files on the search head only. For distributed high performance analytic stores, accelerated data models are the way to go.

Minimum Requirement: List summarization techniques and acceleration if applicable

DATA AND KNOWLEDGE MANAGEMENT

Common Information Model support

A large part of Splunk's business and associated content is in the security and operations space. Splunk has introduced the Common Information Model or CIM. CIM is a set of field names and tags which are expected to define the least common denominator of a domain of interest. It is implemented as documentation on the Splunk docs website and JSON data model files in this add-on. Use the CIM add-on when modeling data or building apps to ensure compatibility between apps, or to just take advantage of these data models to pivot and report.

Recommended: List any sourcetypes and CIM mapping (if applicable). If not applicable to CIM, then state "N/A".

Sample Data: Splunk Eventgen

Some how, some way we have to have sample data. The downside of using samples that are merely copied and pasted from a source is two fold. One downside is security. A developer can mistakenly include IP addresses, hostnames and the like that never were meant for public consumption. Another is the time stamp. It's frozen at the moment the data was copied. If there is an interest in testing and add-on out with another add-on from another developer, we need a consistent way of handling time.

Fortunately Splunk has an open source eventgen on github, located here:
https://github.com/splunk/eventgen

The eventgen runs on the premise that the developer has a piece of raw data. The raw data is edited, and strings replace key words. For example, everywhere a username is used, the string "AAAAAAAAA" will replace it. The eventgen will then run and replace the anonymized string with entries from a sample file that can be anything such as Disney characters.

Another potential future use is performance testing if we decide to go there.

Minimum Requirement: Sample data to be provided with app (will not be distributed)
Recommended: Splunk Eventgen support.

RECOMMENDATIONS AND GUIDELINES

Post processing for dashboards

You can save search resources by creating a dashboard that feeds all downstream panels with one single search. Splunk recommends the proper use of this to enhance the user experience for content that includes dashboards, since it can work to minimize the number of searches required. However, do not use post processing if the parent search is non-reporting, since this can lead to incomplete results. For more information, please see: http://docs.splunk.com/Documentation/Splunk/latest/AdvancedDev/PostProcess

Summarization and acceleration techniques

Splunk Enterprise is capable of generating reports on massive amounts of data. However, the amount of time it takes to compute such reports is directly proportional to the number of events they summarize. Plainly put, it can take a lot of time to report on very large data sets.

Splunk Enterprise provides three data summary creation methods:

- Report acceleration – Uses automatically created summaries to speed up completion times for certain kinds of reports.
- Data model acceleration – Uses automatically created summaries to speed up completion times for pivots.
- Summary indexing – Enables acceleration of searches and reports through the manual creation of separate summary indexes that exist separately from your main indexes.

ASIDE: Another benefit of the CIM is that it offers a development model. As we know, Splunk revels in unstructured data. Humans do not and need structure. The CIM offers a structure that is not only functional for IT and Security use cases, but is also a great model for structuring the unstructured that could benefit other use cases as well.

Summary indexes are a separate index that has to be maintained separately, requiring separate retention and access controls, which fit some use cases. They are also more efficient for manually sharing summary data across many reports. By default, summary indexes reside on the search head, but can be forwarded to the indexers in a distributed environment, which is useful volume is large. For more information, please see: http://docs.splunk.com/Documentation/Splunk/latest/Knowledge/Aboutsummaryindexing

Dashboard and user interface

Splunk realizes that there is a lot of nuance here, but we can offer guidance based on many years of experience in helping other Splunk developers. Dashboards with an overload of charts and reports (say, 20+) should be discouraged. We do encourage the use of forms when appropriate to allow for more robust reports. For more information, please see: http://docs.splunk.com/Documentation/Splunk/latest/Viz/Buildandeditforms

Search efficiency

SPL™, the search processing language, can serve a variety of users ranging from those who just search key words to those who mining their data with powerful statistics. With an arsenal of commands at the user's disposal, there is room for less than proper use. Here are a few examples:

- Careful and considered use of commands, e.g. sort, eventstats, join, sub searches
- Tricks like using TERM() for IP address when possible.
- Lookups should be reasonable. Splunk will encourage the proper use of file based, script based and key/value store lookups.
- More?

Checklist Summary

Things to check for:

No .pyc or .pyo files
No local.meta
No "local" directory or empty "local" directory
Installs as documented
Python scripts must OS safe paths (Windows paths for Windows, Unix paths for Unix)
No emails in savedsearches.conf
Remove hidden files, version control remnants like dot-files, tilde-files, etc.

Declare the following if applicable:

Creates an index (indexes.conf)
Changes Splunk limits (limits.conf)
Implements summary index
Implements tscollect
Implements report acceleration
Implements data model
Implements data model acceleration
Contains inputcsv/outputcsv
Any outbound communications (outputs.conf, perhaps more)
Verify list Open Source components
Declare and document custom commands
Is Adobe Flash required for visualizations?
Access to files outside the "app" directory

Recommendations:

Post-processing in dashboards (where applicable)
- see: http://docs.splunk.com/Documentation/Splunk/latest/Viz/Savedsearches#Post-process_searches