Building Large Scale Splunk Environments? Automation and DevOps are a must.

by Subir Grewal | May 7, 2019

Introduction

Splunk is a log aggregation, analysis, and automation platform used by small and large enterprises to provide visibility into computing operations and as a Security Incident and Event Monitoring platform. It is a very mature product, with deep penetration in financial services firms. The Elastic-Logstash-Kibana (ELK) stack is an open source suite with a similar feature set.

For a large organization, log aggregation platforms can easily end up ingesting terabytes of data. Both Splunk and ELK require incoming data to be indexed and rely on distributed storage to provide scale. In practice, this means dozens, or even hundreds of individual nodes operating in a cluster. Once an organization moves past a 2-3 node log-aggregation cluster, a strong automation framework is essential for testing and deploying all components of the log aggregation platform.

Risk Focus has built large analytics clusters for a number of clients in recent years, including those focused on financial and healthcare data analysis. Several engagements have involved building large Splunk clusters. The automation challenge for log aggregation platforms are similar in many respects to other analytics platforms. In almost all cases the three major challenges are:

- Managing data ingestion/archiving
- Configuring and scaling compute clusters required for analysis
- Maintaining consistent configuration across entire cluster and robust testing

Infrastructure automation

Based on our experience, Splunk clusters containing more than a handful of nodes should not be built or configured manually. Automation delivers configuration consistency and prevents configuration drift. It also improves efficiency in resource deployment and utilization. This is true for both the base system configuration as well as cluster deployment and configuration tasks.

Organizations using current generation technology can deploy standardized base operating system images to virtual machines, speeding up initial infrastructure deployment significantly. An automation and configuration management tool (such as Salt or Ansible) can then be utilized to deploy software and customized configuration onto each node within the network. Cloud orchestration technology (e.g. Terraform or AWS Cloudformation) may be utilized alongside configuration management tools in particularly large environments. As an aside, AWS has a managed Elasticsearch/ELK offering. For organizations considering Cloud deployments, there is no need to re-invent the wheel, the AWS offering does virtually everything an organization might want in terms of infrastructure automation, including multi-AZ deployments for High-Availability.

Utilizing such automation frameworks makes it extremely easy to scale the environment up (and in certain cases down). It also simplifies other common management tasks critical for operational stability, including:

- **Adding search-heads or indexers**: This is easier with a well-understood automated deployment process not subject to manual error.
- **Disaster Recovery**: is easier and less costly to accommodate when compute resources are stood-up quickly and confidently with automated procedures.
- **Resource Efficiency**: When search, ingestion and reporting follow a specific pattern during the day, or different business units require additional capacity, infrastructure automation enables re-scaling of components and re-directing resources towards other tasks/nodes.
- **Testing/Upgrading/Patching/Re-Configuring software**: A consistent and modern DevOps practice is necessary to make modifications to a large cluster reliably and with minimal downtime.
- **Security and Auditability**: Financial services firms, health-care providers, and utilities face high regulatory burdens. Auditors and regulators have an interest in the computational operations of these clients. Splunk/ELK is a good source of data for audits and an indicator of mature operational management and monitoring practices within the information technology organization. As an organization begins to rely more on Splunk for both security monitoring and operational visibility, it can expect auditors and regulators to treat Splunk as critical infrastructure and take a greater interest in the Splunk environment itself. Employing automation and consistent DevOps practices to build, deploy and manage the Splunk environment goes a long way towards allaying regulatory concerns. But more importantly, to ensure operational stability, large Splunk clusters (or any other large-scale analytical/compute environment) should use automation for initial deployments and to manage configuration drift across the fleet.

Splunk Administration Tools

Splunk has a rich toolset of GUI and CLI tools to manage configurations for indexers, forwarders, license managers and other components in a Splunk cluster. Most firms are likely to have some form of automation/DevOps standard across the organization, in aspiration if not yet in full practice. A key part of planning for a large Splunk environment is defining where the boundary between standardized, cross-platform configuration management tools end and where the Splunk toolset exercises control.

There is no perfect answer to where this boundary lies. Risk Focus works with clients who have limited the use of Splunk management tools to managing licenses and performed virtually every other task with scripted automation frameworks. There are other clients, who chose to go in the opposite direction and rely largely on Splunk's tools to manage and configure the cluster.

Making the right decision on tooling is dependent on the practices and skillset of the team expected to support Splunk. Organizations that rely on external service providers (including Splunk PS) to maintain their environment need to consider whether these providers are familiar with their preferred automation or configuration management toolkits. Organizations that prioritize mobility among technology staff will want to place more emphasis on common tooling across all applications, rather than rely on Splunk-specific management tools.

Application Life Cycle

Testing and Quality Assurance
An important and often underappreciated part of Splunk environment management is how to deal with the testing and release of software updates and patches, reports, dashboards, and applications.

Organizations using Splunk for critical business activity should treat Splunk like any other business critical system and follow best practice for software delivery and maintenance. That means establishing processes to test software releases. A release can impact user activity, reports, dashboards, ingestion configurations, and system performance. Building Splunk test environments and efficiently rolling changes through a development/test/deployment lifecycle requires automation. Absent such automation, test cycles become expensive and will not be executed consistently. Mature organizations using a Continuous Integration/Continuous Deployment (CI/CD) lifecycle will find that the effort expended to integrate Splunk into their CI/CD pipeline delivers enormous rewards over time.

Release Management
In managing releases and updates to Splunk configuration, it is useful to view Splunk in a broader context as a data analytics tool. Most organizations have some experience with data analytics tools such as SAP, SaaS, Informatica, or Business Intelligence. For these systems,

organizations have often established fine-grained control over reports and dashboards used in critical business activities. This includes limiting users' ability to change them in production. Splunk is no different. We advise customers to clearly establish permission and ownership boundaries in their production Splunk environment. These should be balanced so that they do not constrain the Splunk users' ability to analyze data.
One way to balance the tension between user freedom and organizational oversight is to create a dividing line between reports/dashboards used in daily operations and ad-hoc analyses. We find that the following best practices are quite useful:

- Critical reports and dashboards should be tightly controlled via Splunk's permission tools
- Changes to critical dashboards, however minor, should be tested
- Software upgrades should involve user acceptance testing and automated testing of all such dashboards
- Data from test environments should be ingested continuously into a Splunk test environment. Application/infrastructure changes which might impact Splunk should be tested here.
- Critical monitoring dashboards should be validated as part of application testing to ensure any changes to software/log format do not impact these dashboards.

Example

Splunk scales well to meet the needs of large enterprises, but the topology can get complex with increased scale. An example of a deployment framework for a multi-tenant Splunk cluster is below. Depending on your organization's need, automated or semi-automated scaling can be built into the framework.

Conclusion

The current generation of automation tools and DevOps best practices can deliver significant benefits to an organization seeking to manage and maintain a large Splunk cluster. Every organization should carefully consider the benefits of using such tools to manage their environment. Organizations relying on Splunk for critical operational management should treat it as such and build a robust testing framework for their environment.