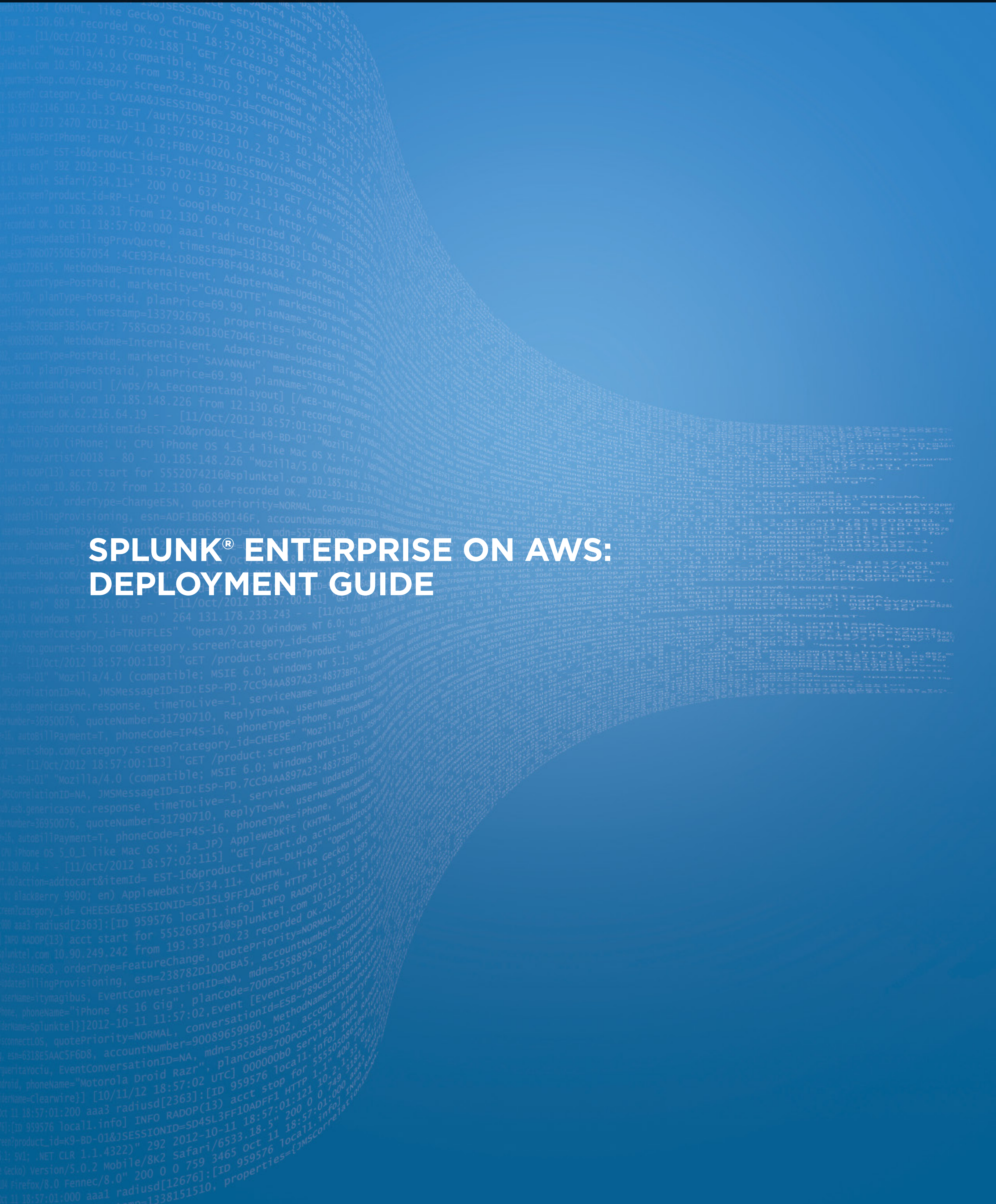
splunk>

WHITE PAPER



SPLUNK® ENTERPRISE ON AWS: DEPLOYMENT GUIDE

Introduction

This deployment guide discusses the steps and architectural considerations for deploying Splunk Enterprise on the Amazon Web Services (AWS) cloud. It also provides links for viewing and launching AWS CloudFormation templates that will help automate the deployment.

This guide is for IT infrastructure architects, administrators and DevOps professionals who are planning to implement or extend their Splunk Enterprise deployment on the AWS cloud. The reader should have familiarity with core AWS services and a general idea of cloud concepts. The reader should also be familiar with core Splunk concepts, including both **indexer clustering** and **search head clusters**.

Synopsis

Splunk is a platform that makes machine data accessible, usable and valuable to everyone. By monitoring and analyzing everything from customer clickstreams and transactions to security events and network activity, Splunk software helps customers gain valuable Operational Intelligence from their machine-generated data. With a full range of powerful search, analysis and visualization capabilities and pre-packaged content for use cases, any user can quickly discover and share insights.

This paper includes architectural considerations and configurations used to build a Splunk Enterprise deployment on the AWS cloud. It will also cover how to configure the necessary AWS services, such as Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Virtual Private Cloud (Amazon VPC).

Users may select from CloudFormation templates that can build either a single "all-in-one" instance or a distributed multi-indexer deployment. This document also provides guidance on how to build fault-tolerant deployments utilizing Splunk clustering technology. Available templates may be used as a starting point for deployments with specific requirements.

Architecture Overview

This section discusses some of the choices to make when deploying Splunk Enterprise and the corresponding AWS services used in the Splunk deployment.

It is important to make fundamental architecture decisions before deploying Splunk Enterprise. Will there be a single, all-in-one instance, or will the deployment need multiple instances across a distributed deployment? What sort of storage will meet deployment requirements? Is high availability a requirement? Should AWS Elastic Load Balancing (ELB) be a consideration for load balancing the deployment?

The AWS components used by this guide include the following AWS services:

- **Amazon VPC** – The Amazon Virtual Private Cloud (Amazon VPC) service provisions a private, isolated section within the AWS cloud. AWS services can be launched from within this virtual network. The VPC grants complete control over virtual networking environment including authority over IP addresses, subnets, routing tables and internet gateways, among others.
- **Amazon EC2** – The Amazon Elastic Compute Cloud (Amazon EC2) service is engineered to launch virtual machine instances with a variety of operating systems. Users may can choose from existing Amazon Machine Images (AMIs) or import their own images.
- **Amazon EBS** – Amazon Elastic Block Store (Amazon EBS) provides persistent block-level storage volumes to use with Amazon EC2 instances. Each EBS volume is replicated within its Availability Zone to mitigate any risk from component failure, offering both high availability and durability. With several volume types to choose from, EBS provides the flexibility to tailor a deployment to best suit a user's needs.

- **Amazon ELB** – Amazon Elastic Load Balancing (ELB) automatically distributes incoming application traffic across multiple Amazon EC2 instances. Amazon ELB helps customers achieve and maintain fault tolerance in applications, seamlessly providing the required amount of load balancing capacity needed to route application traffic.

Implementation Details

Single instance versus distributed deployment

The first decision to make when deploying a Splunk cluster is whether or not to deploy a single node deployment or a distributed environment.

- A **single node deployment** is one server providing all Splunk-related functionality; license server, indexer and search head. Typical use cases for a single-server deployment might be for small scale non-HA low volume production scenario, proof of concept or dev/test/QA cycle. We recommend a maximum of 300GB/day on a single node deployment.
- A **distributed deployment** consists of several instances working together to provide indexing and search head duties. These deployments can range anywhere from two to hundreds of instances.

Further information on the dimensions of a Splunk Enterprise deployment on AWS can be found in the Splunk Enterprise capacity planning manual.

Storage

In most circumstances, the deployment type and use cases will dictate the type of storage used.

When deploying a **non-clustered environment**, either single-server or distributed, we recommend utilizing EBS volumes and EBS-optimized instance types.

When architecting a **clustered solution**, there are two storage options available:

- **Instance storage** is temporary, often referred to as "ephemeral" storage. Once the instance is terminated or crashes, all data stored on the instance is lost.
- **An EBS volume** is persistent, even in the case of instance termination or crash. Each of these options come with their own feature set and price points. Exact storage selection will depend on a customer's desired requirements and price sensitivity.

Data Acquisition

After deciding on the back-end architecture, the next step is to determine how best to get the data into the Splunk platform.

- Using a **Splunk universal forwarder (UF)** installed on the machine where the data resides is recommended in most scenarios. A simple example of this would be installing the UF on a webserver to forward the webserver logs to the Splunk indexers.
- A **Splunk heavy forwarder**, also referred to as a heavy weight forwarder (HWF), adds additional capabilities at the forwarding tier. In more advanced scenarios, a HWF may provide the deployment with additional capabilities that the UF cannot.
- In the case where a forwarder can't be installed on the client that is generating data that needs to reside in the Splunk platform, sending that data directly to **Splunk's HTTP Event Collector**, or HEC, is an option. In such situations, placing Splunk HTTP event collector(s) behind an ELB, and configuring the client to send the events to the load balancer address is recommended. This makes future scaling easier and painless.

Splunk provides an add-on that will automate ingest data from several AWS services, including Config and Config Rules, CloudTrail, CloudWatch, Inspector, CloudFront and more. Installing and configuring the add-on will allow data collection from any of the services a user selects directly to the user's Splunk Enterprise deployment.

Provisioning Splunk in AWS

Sizing

Once the general deployment architecture has been decided, the next step will be to determine the instances and specific storage types on which to deploy Splunk solutions. Decisions at the compute layer depend on the specific use cases of the deployment, but there are a few generalizations that can be used as a starting point.

Instance Selection

As a general rule, deployments without Splunk premium solutions, single node and non-clustered deployments should use c4 instance types. Clustered deployments may opt to use d2 instance types. Which specific type of instances to deploy depends on how much workload will be delivered to the instance. In Splunk's case, the workload is defined as both indexing and search. In this document, use cases typically considered average will drive the recommendation, but it's important to understand that heavy searching can impact performance as much—or more than—heavy indexing. The assumptions below assume EBS gp2 volumes attached to the c4 instance types. The d2 instance types come with instance storage, so EBS is not required. In all situations, deploying on dedicated hosts to avoid potentially noisy neighbor situations is recommended.

Indexers

Instance Type	Daily Indexing Volume (GB)
(c4 or d2).2xlarge	<100
(c4 or d2).4xlarge	100-200
(c4 or d2).8xlarge	200-300+

Search Heads

Instance Type	Concurrent Users
r4.4xlarge	Up to 8
r4.8xlarge	Up to 16

When using Splunk premium solutions, such as Splunk® Enterprise Security (ES) or Splunk® IT Service Intelligence (ITSI), we recommend indexer instance types with a larger memory footprint. The following assumptions also assume EBS gp2 volumes attached.

Indexers (Splunk Premium Solutions)

Instance Type	Daily Indexing Volume (GB)
r3.8xlarge	100
m4.10xlarge	100-150

Search Heads (Splunk Premium Solutions)

Instance Type	Concurrent Users
r3.4xlarge	Up to 8
r3.8xlarge	Up to 16

HA/DR Considerations

For deployments requiring either high availability (HA) or resilient disaster recovery (DR) capabilities, Splunk Enterprise has built-in clustering technology to safely replicate data. There are two separate clustering options, search head clustering and indexer clustering, that allow for considerable flexibility when architecting a deployment.

Indexers

Indexer Replication

The importance of ensuring resiliency and recoverability of data in the event of a node failure or other disaster cannot be overstated. One way of doing that at the indexer tier is to use Splunk clustering and index replication.

The three primary considerations when deciding how to provision index clustering:

1. Is a secondary site necessary? Does the deployment require a secondary physical location to mitigate losses in the case of natural disaster or otherwise catastrophic damage to the primary location? This is called multi-site clustering.
2. How many copies of data should the cluster replicate? Essentially, how many failed nodes should the deployment be able to tolerate before data loss? This is called replication factor. The number of concurrently tolerated failed nodes is replication factor -1. For example, if the replication factor is set to 3, then two nodes can fail without data loss.
3. How many immediately searchable copies of data should the cluster retain? This is called search factor. It is possible to have a replication factor higher than a search factor if data resiliency is the primary requirement, as opposed to high availability search capability.

In the case of multi-site clustering, each distinct location is referred to as a ‘site’ with the replication factor (RF) and search factor (SF) configured on a per-site basis. For example, a primary and DR configuration might configure the total RF and SF to 4. The primary site could maintain three replicated and searchable copies while the backup/ DR site maintained one. A cluster cannot have a search factor greater than the replication factor.

In AWS deployments, a Splunk Enterprise site will typically be an Availability Zone or a region, depending on requirements. Each AWS AZ is a physically unique facility, and many customers will find multi-site deployments across multiple AZ in a single region sufficient for HA. If the requirements are for a physically separate region, then treating an entire region as a site will fulfill the requirement.

The following diagram illustrates an indexer cluster where both SF and RF equal 3.

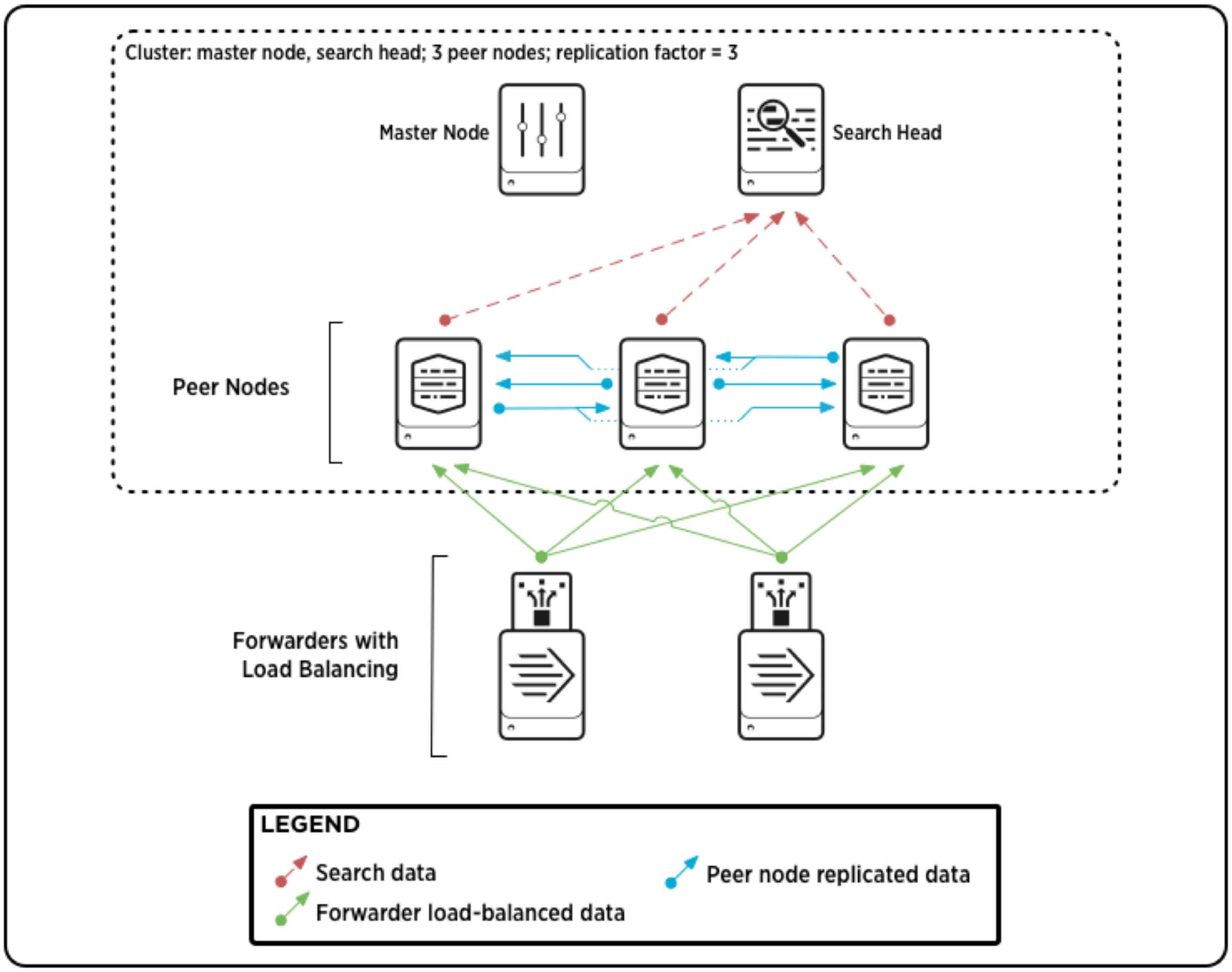

Cluster: master node, search head; 3 peer nodes; replication factor = 3
Master Node
Search Head
Peer Nodes
Forwarders with Load Balancing
LEGEND
Search data
Forwarder load-balanced data
Peer node replicated data

Figure 1 - source: Splunk multi-site cluster architecture documentation.

When deciding how to properly provision a clustered deployment, it’s important to consider the differences in requirements of a replicated bucket versus a replicated and searchable bucket. For a replicated-only bucket, it estimated that roughly 15% of the original size of indexed data will be copied to the replicated destination(s). For a replicated and searchable bucket, the estimate increases to about 50% of the original indexed data. Lastly, note that a Splunk cluster will try to always maintain the appropriate number of replicated buckets. When a node failure occurs, all buckets on that failed node will try to be re-replicated across the cluster to maintain the configured replication count.

To protect against filling drives in the event of a failure, reserving - at minimum - one indexer’s worth of storage across the cluster is recommended, to allow for such failures without filling up the storage volumes. For example, if a cluster has ten indexers, each with 10TB of disk space, reserving a minimum of 10TB in total across the cluster (approximately 1TB per indexer) would be the suggested course of action. It is recommended to keep a cluster around 80% capacity to allow for both failures and unexpected spikes in traffic.

Search Heads

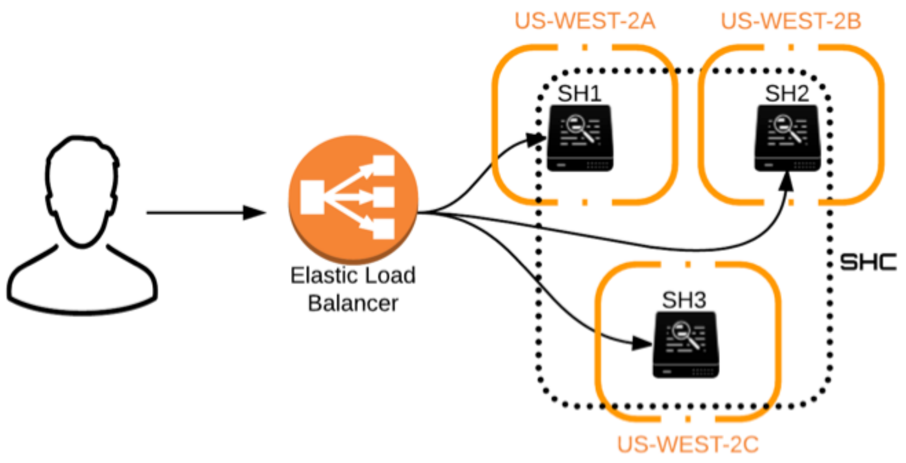
Search Head Clustering

Similar to indexers, search head clusters will allow for replication of configuration, job scheduling and saved search results. The replication factor works the same way as the indexers; a search head cluster can tolerate simultaneous failures from (replication factor -1) nodes. Again, similar disk space considerations should be implemented to accommodate for artifacts being re-distributed upon node failure.

ELB

The AWS Elastic Load Balancing (ELB) service can provide a Splunk Enterprise deployment with an additional layer of fault tolerance in a few different scenarios. Deploying in front of a search head cluster is one such scenario.

When deploying a Splunk search head cluster (SHC), placing all of the search heads behind an ELB is highly recommended to help ensure the highest availability. By placing the SHC instances in different Availability Zones, and the ELB directing traffic across only healthy search heads, any outage in a single Availability Zone will automatically be routed around by the ELB.



Configuring Splunk in AWS

This introduction has covered topics on architecture and provisioning. It will now proceed to outline how to configure Splunk software in several of these scenarios.

Storage

Data is central to everything Splunk does. As mentioned earlier, indexer data is extremely important and this section will contain examples of how to configure indexes and their retention policies.

Indexes

As explained in Splunk documentation, a Splunk index is stored in directories called buckets. These buckets go through a four-stage lifecycle: hot, warm, cold and frozen. Those four stages break down like this:

Bucket Stage	Description	Searchable?
Hot	Contains newly indexed data. Open for writing. One or more hot buckets for each index.	Yes
Warm	Data rolled from hot. There are many warm buckets. Warm buckets are read-only.	Yes
Cold	Data rolled from warm. There are many cold buckets. Cold buckets are read-only.	Yes
Frozen	Data rolled from cold. The indexer deletes frozen data by default, but you can also archive it. Archived data can later be thawed.	No

As data ages, it will go from hot to warm, warm to cold and from cold to frozen. Under typical usage patterns, hot and warm buckets will be searched far more frequently than a cold bucket. Consequently, the underlying storage requirements for each bucket type is likely to be different. More frequently accessed data should be on higher performing storage than rarely accessed data.

There are two EBS storage volume types that we generally recommend; gp2 for hot/warm and st1 for cold. Due to the way AWS handles EBS volume I/O credits, gp2 volume types operate best at 1TB or larger. Similarly, st1 volume types operate best at 12TB and larger.

While segregating volume types for the different bucket types is not necessary, doing so can allow for considerable cost savings. As of this writing, an st1 volume costs 45% of a gp2 volume.

The most common method to control how the data is moved through each lifecycle stage is by time. Splunk refers to the migration of buckets through stages as “rolling” buckets. The various options are covered extensively in index storage configuration docs, but this example will focus on managing retention by time.

In this example, assume the following:

- 1TB/day indexing
- The deployment needs roughly 15 days of hot/warm retention and an additional 45 days of cold retention (60 days total)
- The gp2 volume is mounted as /data/hot
- The st1 volume is mounted as /data/cold

To achieve this, the pertinent configuration would include settings similar to the following:

```
[volume:hot1]
path = /data/hot
maxVolumeDataSizeMB = 7500000

[volume:cold1]
path = /data/cold
maxVolumeDataSizeMB = 22500000

[my_index]
homePath = volume:hot1/my_index
coldPath = volume:cold1/my_index
thawedPath = $SPLUNK_DB/my_index /thaweddb
```

These settings are detailed in the index storage docs noted earlier, in addition to indexes.conf documentation.

Backups

AWS makes dealing with backups of EBS volumes incredibly simple via snapshots. There is a caveat to this simplicity, however, in that AWS charges five cents per GB per month for a snapshot. For comparison, it's possible to manually backup to Amazon S3 and use what they call the infrequent access (IA) storage type for 1.25 cents per GB per month, and an additional 1 cent per GB upon retrieval. Included are examples of both methods.

Snapshots
In AWS, an EBS snapshot is a point-in-time snapshot of the selected volume. A snapshot allows the user to recover all data on the volume at the time the snapshot was taken. To back up via snapshots, simply script the snapshot for volumes at the appropriate interval that is acceptable for the deployment. In some cases, a daily backup may suffice, while others will require multiple backups per day.

Recommended steps for backing up index volumes are:

1. Roll hot buckets to warm
2. Determine EBS volume id(s) where the hot/warm buckets reside
3. Create snapshot of appropriate EBS volume(s)

Note: The following steps require installing the AWS CLI tool.

Before backing up, rolling the hot bucket(s) to warm is strongly recommended. Wait to roll the buckets until just before the snapshot is executed. The goal is to snapshot the volume while there is as little new data in the hot bucket as possible at the time of snapshot. The following command should be used for the rolling of hot to warm, and more information about index backups can be found in Splunk documentation.

```
splunk _internal call /data/indexes/
<index_name>/roll-hot-buckets–auth
<admin_username>:<admin_password>
```

Next, find the EBS volume id(s) where the index is writing to. A sample shell script is provided for users to get started, but this is not officially supported. This script will look up the instance-id of the instance it's running on, check for mounted EBS volumes and then reference those with the existing mount points where the warm and cold directories are configured. Access sample shell scripts at:

https://github.com/splunk/splunk-aws-cloudformation/blob/master/scripts/get_ebs_volumes.sh

Once the volume id(s) have been determined, a simple CLI command is used to create the snapshot with an optional description. See the detailed explanation and other settings for the snapshot in the AWS create-snapshot documentation. Using an example variable name from the previous script, the scripted snapshot command might look something like this:

```
aws ec2 create-snapshot –volume-id $warm_
volume –description "splunk indexer backup
on `date +'%m-%d-%y`"
```

Backups to Amazon S3
The following guidance is intended for non-clustered deployments. Most multi-site clustered deployments do not do traditional backup because the data is already replicated across multiple sites. If a clustered deployment requires an additional level of redundancy, reaching out to Splunk Professional Services for guidance is recommended due to the complicated nature of a clustered backup.

Schedule incremental backups of the warm buckets. As shown in the earlier chart explaining bucket types, warm buckets are read-only. This means they will not change from the time they are rolled from hot, through warm and then to cold. From an incremental backup perspective, it's as simple as scheduling a sync of the warm Splunk buckets to S3.

Setting up the backups should be relatively straightforward. First, create an S3 bucket where the backed-up buckets should reside. While S3 provides several storage classes, reserving Infrequent Access for backups that will not be used or accessed often is recommended.

Once the S3 bucket has been created, scheduling a simple script like the one below should be all that's required for sending warm buckets to S3. The example script is only backing up a single index, but it's simple to expand to additional indexes. Again, this requires installing the AWS CLI tool. Lastly, this is not a supported script, but it should help in getting started.

https://github.com/splunk/splunk-aws-cloudformation/blob/master/scripts/warm_bucket_backup.sh

While using the "sync" option, any existing warm buckets already archived to S3 will not be re-uploaded. There are a few things to keep in mind with this approach:

1. A script written as per this example would result in the very first upload being as large as all warm buckets combined. Assuming the earlier example of 10GB/day, and the backups started after 15 days of indexing, this initial upload would be in excess of 100GB.
2. Make sure the frequency that the scheduled backup runs at is less than the time it takes to cycle through all of the warm buckets. For example, if an indexer is storing about ten days of warm buckets, make sure the scheduled backup runs less than every ten days.
3. If the backup policy for the deployment is something other than saving all backups indefinitely, we strongly encourage configuring S3 lifecycle configuration to manage the backup S3 bucket. For example, if the deployment requires 180 days of backups, then set the lifecycle configuration to remove objects older than 181 days.

    a. There is an additional consideration if the backups are not required to be "online" and available for immediate recovery. Amazon provides another, even less expensive, storage service called Glacier. Glacier is unique in that retrieval must be queued and will take several hours—at minimum—to recover from. The trade-off for slow recovery is cost; only .07 cents per GB per month, or $0.007/GB/month. Automatic migration to Glacier from other S3 storage classes is available via similar S3 lifecycle configuration methods mentioned earlier.

Indexer Cluster Configuration

Master Node

In Splunk indexer clusters, the node that coordinates all the activities of the peer indexers is called the master node. The master node does not store or replicate any indexing data, and thus can be configured on a much smaller instance than an indexer. Splunk's recommendation for a cluster master node is c3.xlarge for small and medium-sized deployments of less than 1TB per day, and a c3.2xlarge for larger deployments.

This guide will discuss how to configure the master node via the Splunk web interface, but the master node documentation goes into further detail on how to configure via server.conf and/or the Splunk CLI.

From the "Index Clustering" section (settings -> distributed environment -> Indexer Clustering), enable indexer clustering and select "master node." After clicking "next," there are several options to consider. Replication factor and search factor were covered in the earlier architecture section.

- Replication Factor
- Search Factor
- Security Key – This is a shared key that validates communication between the cluster members. The key needs to be identical across all nodes.
- Cluster Label – As straightforward as it sounds, this is simply a cluster label.   If a deployment has multiple clusters, this will help identify it in the DMC (Distributed Management Console). For more information on cluster labels, please see the related documentation.

Once those have been entered, enable the master node and then go to server controls to restart the cluster.

An important note: Indexing will be blocked until the requisite number of indexers to meet the defined replication factor have been restarted. Do not restart the master node while it waits for them to connect or all the peers will need to be restarted a second time.

For additional information on how to change these configurations at a later time, look over the master configuration overview docs.

Indexers

After the master node has been configured, the deployment must have peer indexers configured to become part of the cluster. As with the master node, it's possible to configure peer indexers via the CLI or directly editing server.conf. In this example, configuration via the Splunk web interface is outlined.

From the "Indexer Clustering" section (settings -> distributed environment) select "Enable indexer clustering" and then select "Peer node." Again, just like the master node configuration, there are a few fields to fill out.

- Master URI – This is the URI that the peer nodes will use to connect to the master node. As mentioned previously, we recommend using a DNS record for this.
- Peer replication port – The port where all replicated data between peer indexers is transferred. Any available port will suffice, however, it must be different from the management and receiving ports.
- Security Key – This is the key that authenticates communication between the master and the peers and search heads. This must be the same key used in the master node configuration.

Enable the peer node, go to server controls and restart the peer. Repeat as necessary for all peers in the cluster. Once the number of configured and restarted peers reaches the replication factor number, the indexers can start indexing.

Search Head Cluster Configuration

A search head cluster (SHC) is simply a group of search heads that share configurations, job scheduling and search artifacts. One member of the cluster assumes the role of captain, and the captain coordinates job and replication activities among the cluster members. The captain role is dynamic, and will switch among different cluster members over time so that one node failing doesn't compromise the cluster.

Along with the cluster itself, a functional SHC also requires a Deployer Node and typically has a load balancer to handle inbound traffic to the SHC. In the diagram below, the load balancer would most likely be an ELB for an AWS deployment.

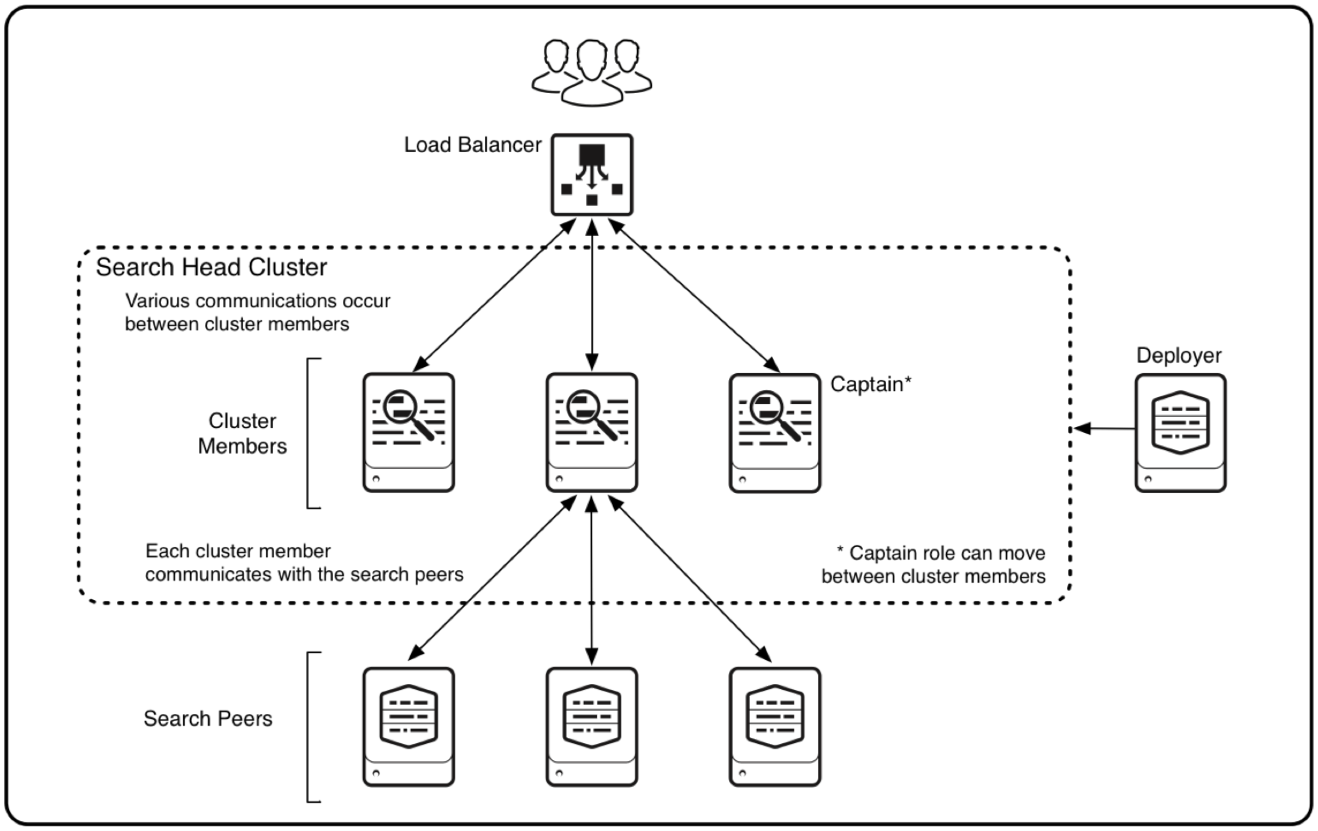

Load Balancer
Search Head Cluster
Various communications occur between cluster members
Cluster Members
Captain*
Deployer
Each cluster member communicates with the search peers
* Captain role can move between cluster members
Search Peers

Figure 2 source: Splunk search head clustering architecture documentation

The deployer is not a member of the cluster itself, but distributes apps and other configurations to the cluster members.

The primary steps required to configure the SHC:

1. Determine requirements – This will include, among others, how many search heads are to be in the cluster and what the replication factor of a cluster should be. Much like index clustering, an SHC will replicate data to prevent any single node failure from causing a loss of data.
2. Setup Deployer – A c3.xlarge instance type is recommended here. This node will not be in the cluster itself. Configure the deployer's security key, which is used to authenticate with the cluster members. While this is an optional step, configuring it is highly recommended. To set the security key, use the pass4SymmKey attribute in the server.conf under either the [general] or [shclustering] stanza. An example:

```
[shclustering]
pass4SymmKey = yoursecuritykey
```

   Lastly, set the cluster label. This is optional to help identify the cluster in DMC, but if it is set, it must be set the same on all cluster members and the deployer. This configuration goes under the [shclustering] stanza in server.conf:

```
[shclustering]
shcluster_label = yourclustername
```

3. Install Search Heads – Earlier, this paper recommended specific instance suggestions for search heads, and it is recommended to use those instance types. Install Splunk software on those instances, and be sure to change the default admin password. The CLI commands used to configure the cluster will not operate with the default password.
4. Initialize cluster members – For each instance that is intended for inclusion in the cluster, run the Splunk init shcluster-config command and restart the instance: (do not run this on the deployer)

```
splunk init shcluster-config -auth
<username>:<password> -mgmt_uri
<URI>:<management_port> -replication_port
<replication_port> -replication_factor <n>
-conf_deploy_fetch_url <URL>:<management_
port> -secret <security_key> -shcluster_
label <label>

splunk restart

-auth = the local Splunk instance
username and password to authenticate.
-mgmt_uri = local Splunk instance
URI:management port. 8089 is the default
management port.
-replication_port = The port on local instance to
be used for replication traffic. Using the same
replication port across all SHC members is
recommended,
but do not reuse ports already being used. This
must be an unused port.
-replication_factor = How many copies of data the
cluster should copy.
-conf_deploy_fetch_url = This is the deployer URI
and management port.
-secret = The cluster secret configured on the
deployer.
-shcluster_label = The cluster label. As mentioned,
this is optional but it must be the same across all
indexers and the deployer.
```

An example command might look like this:

```
splunk init shcluster-config -auth
admin:changed -mgmt_uri https://sh1.
example.com:8089 -replication_port 34567
-replication_factor 2 -conf_deploy_fetch_
url https://10.160.31.200:8089 -secret
mykey -shcluster_label shcluster1
```

All of these configuration settings are explained in the SHC deployment overview documentation.

5. Bring up the cluster captain – The initial captain selection is done manually. It doesn't matter which instance is selected as long as it is a member of the cluster. To elect the captain, run the splunk bootstrap shcluster-captain command:

```
splunk bootstrap shcluster-captain
-servers_list "<URI>:<management_
port>,<URI>:<management_port>,..." -auth
<username>:<password>
```

-servers_list = comma separated list of the cluster members, including the captain that is executing this command.

An example captain election command might look like this:

```
splunk bootstrap shcluster-captain
-servers_list "https://sh1.example.
com:8089,https://sh2.example.
com:8089,https://sh3.example.
com:8089,https://sh4.example.com:8089"
-auth admin:changed
```

Important note: The URIs specified in servers_list must be exactly the same as the URIs specified when each cluster member was initialized in the earlier step.

6. Perform post-deployment set-up:
   a. Connect SHC to search peers. This is a required step, but the instructions depend on whether the peers are in an indexer cluster or not.
      i. If the peers are in a cluster, follow these instructions.
      ii. If the peers are not in an indexer, follow these instructions.
   b. Add users. This is a required step to add users who will be logging into Splunk Enterprise. There are several options—Splunk authentication, LDAP, SAML or scripted. To read about these various options with instructions for each, please see the related documentation.
   c. Deploy a load balancer in front of the search heads. This is an optional step, but typically recommended. Instructions for creating an HTTPS ELB can be found in these instructions from AWS. In step 5, when registering EC2 instances with the load balancer, add each of the members of the SHC.

EBS vs Ephemeral

This is likely going to be the biggest cost consideration of any distributed deployment. Utilizing ephemeral storage for clustered deployments is recommended, and the following example illustrates the cost difference between two storage types.

Consider a 500GB/day multi-site clustered deployment, with a search head cluster and a one-year data retention policy. A Splunk deployment typically has a 2:1 compression ratio on the indexes, which leaves a total data storage requirement of ~90TB per cluster site. (All pricing assumes on-demand rates.)

RAID0 is utilized across the d2 instance types to create a single filesystem across all drives. At the time of this writing (Nov 2016), and using this scenario, the luxury of EBS comes with an almost 48% cost premium, and that doesn't include the additional expense of snapshots. While this price premium will point most customers to an ephemeral deployment, there are reasons to still consider EBS. The following assumes the same deployment size as above:

- Cluster recovery speed – When using EBS and snapshots, the recovery time after a node failure is going to be significantly faster than with an ephemeral cluster. Instead of potentially rebuilding upwards of 20TB with an ephemeral deployment, recovering from a snapshot should result in less than 500GB of data re-population, providing a minimum of once-daily snapshots as recommended earlier.
- Data migration – If a customer ever wants to move data from one region to another, or even from one instance to another, EBS provides a significantly better experience than that of ephemeral instances.

- Operational overhead — When dealing with 100 spinning drives in RAID0, as in this scenario, failure rates are a real concern. With any drive failure, a total instance rebuild would be required. Some organizations may want to mitigate this risk by relying on the stronger reliability properties of EBS as compared to ephemeral, or choose to pay for more/larger instance types and utilize RAID10.

It's also important to point out that this example does not differentiate hot and cold storage. Depending upon the use case for a deployment, there are often opportunities to reduce cost by utilizing a lesser performing storage medium on cold data. If a Splunk Enterprise deployment on AWS is going to have rarely searched data, utilize the new st1 and sc1 EBS volume types. More information, including specific cost models, can be found on Splunk's blog.

Additional Considerations

Splunk App for AWS

For any deployment on AWS, both new and existing, we highly recommend installing the Splunk App for AWS. Once the app is installed and configured, the user gains significant operational and security insights into the AWS account(s) configured. While the add-on can be installed on a search head, the recommended configuration is to install the app on a search head and the add-on on a heavy forwarder. Since the Splunk App for AWS talks directly to AWS services for its data, it's not required that the app is installed on an AWS instance. It's equally possible for a user to install the Splunk App for AWS on their own hardware that has internet access to AWS. The installation documentation of the Splunk App for AWS should be able to answer any additional installation questions.

CloudFormation

AWS CloudFormation gives users an easy way to deploy and manage collections of related AWS resources. In essence, a properly constructed CloudFormation template can launch an entire Splunk Enterprise deployment in just a few minutes. For example, we demonstrated a five indexer and single search head deployment on CloudFormation that was fully operational in about three minutes during a "Deploying Splunk Enterprise on AWS" session at the 2015 .conf conference. (Links to session video and slides). Splunk provides several templates, including the template used in the demonstration, as a great way to get started with CloudFormation and Splunk. They're available on the Splunk github page.

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.