**ABSTRACT**

We all are aware of the exponential growth in machine data over the last decade. It was partly because of the growing number of machines in the IT infrastructure and partly because of the increased use of IoT devices. This machine data has a lot of valuable information that can drive efficiency, productivity and visibility for the business. Splunk was found for one purpose: *To Make Sense of Machine Generated Log Data.*

Splunk is a software platform which provides the engine for monitoring, searching, analysing, visualizing and acting on voluminous streams of real-time machine data. Its wide application and suitability make it a versatile technology.

To setup the Splunk, we need to use some sort of scalable computing platform. For the elasticity and better service packages AWS is best. Amazon Web Services provides cloud based computing resources with many useful services like EBS, S3, VPC and a lot more. These all services make the application easy to manage and more efficient in term of computing.

On AWS, we can customize our services as per our need. So, no need to pay extra for extra unused resources. If we don’t want manages thing like network routing or load balancer or anything, AWS provides good services for each requirement.

As the enterprise grows, number of client also increases. This causes complexity in management and deployment. Here comes the Ansible to manage and configure huge number of machines in an easy and secure way. From one machine, we can do configuration on any number of machines at a same time and also controls the services like AWS.

To deploy Splunk we are using AWS as a cloud platform and considering a real-time scenario, to manage a large number of machines, we are building Ansible playbooks to perform everything, starting from creating instance on AWS to configuration of Splunk-cluster

**ACKNOWLEDGMENT**

With immense pleasure and commitment, I would like to present the project assignment. The nature of project on the development of **Deploy Splunk-cluster on AWS using Ansible** has given me wide opportunity to think, implement and interact with various aspects of management skills as well as the new emerging facilities and the technology used in architecture and the enhancements given to the students with a boon of spirituality and curricular activities.

I am highly indebted to **Mr. Aditya Khetan** for his guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

I express deep sense of gratitude towards our project guide and Head of Information Technology department **Dr. Parth Shah** towards their innovative suggestions and efforts to make project a success. It is their sincerity that prompted us throughout the project to do hard work using the industry adopted technologies.

I am sincerely thankful him for the unconditional and an unbiased support during the whole session of study and development. They altogether provide us favourable environment, without them I would not have achieved my Project.

Kandarp Parikh (15IT065)

Karan Rawal (15IT118)

Table of Contents

[CHAPTER.1 INTRODUCTION 1](#_Toc6872977)

[**1.1** **PROJECT OVERVIEW** 1](#_Toc6872978)

[**1.2** **OBJECTIVE** 1](#_Toc6872979)

[**1.3** **SCOPE** 1](#_Toc6872980)

[**1.4** **TOOLS & TECHNOLOGY USED** 2](#_Toc6872981)

[CHAPTER.2 PROJECT MANAGEMENT 4](#_Toc6872982)

[**2.1 PROJECT FLOW** 4](#_Toc6872983)

[**2.1.2 Project Effort and Time, Cost Estimation** 5](#_Toc6872984)

[**2.2** **Project Work Scheduling (Gantt Chart/PERT/Network Chart)** 6](#_Toc6872985)

[**2.2.1** **Pert Chart Representation** 6](#_Toc6872986)

[CHAPTER.3 SYSTEM REQUIREMENTS STUDY 7](#_Toc6872987)

[**3.1 User Characteristics** 7](#_Toc6872988)

[**3.2 Hardware and Software Requirements** 7](#_Toc6872989)

[**3.3 Assumptions and Dependencies** 7](#_Toc6872990)

[CHAPTER.4 SYSTEM ANALYSIS 8](#_Toc6872991)

[**4.1 Product Function** 8](#_Toc6872992)

[**4.2 User Characteristics** 8](#_Toc6872993)

[**4.3 Requirements of new System** 8](#_Toc6872994)

[**4.3.1 Functional Requirements** 8](#_Toc6872995)

[**4.3.2** **Non-Functional Requirements** 9](#_Toc6872996)

[**4.4** **Use case Diagram** 10](#_Toc6872997)

[The above Use case diagram shows the Use cases for a Site Reliability Engineer 10](#_Toc6872998)

[**4.5** **Sequence Diagram** 11](#_Toc6872999)

[**4.5.1 Sequence Diagram for creating the stack** 11](#_Toc6873000)

[**4.5.2** **Sequence Diagram for adding new nodes to the cluster** 12](#_Toc6873001)

[**4.6** **Activity Diagram** 13](#_Toc6873002)

[Chapter.5 SYSTEM DESIGN 14](#_Toc6873003)

[**5.1 PRODUCT PERSPECTIVE** 14](#_Toc6873004)

[**5.1.1 System Interface** 14](#_Toc6873005)

[**5.1.2 Interface** 14](#_Toc6873006)

[**5.1.3 Hardware Interface** 14](#_Toc6873007)

[**5.1.4 Software Interface** 14](#_Toc6873008)

[**5.1.5 Communication Interface** 14](#_Toc6873009)

[**5.1.6 Memory Constraints** 14](#_Toc6873010)

[Chapter.6 IMPLEMENTATION AND TESTING 15](#_Toc6873011)

[**6.1 SPLUNK** 15](#_Toc6873012)

[**6.1.1. Architecture** 16](#_Toc6873013)

[**6.1.2. Splunk Indexes** 18](#_Toc6873014)

[**6.2. Ansible Playbooks** 25](#_Toc6873015)

[**6.3. System Study** 25](#_Toc6873016)

[**6.4. Implementation Of Application** 25](#_Toc6873017)

[**6.5. Screenshots** 26](#_Toc6873018)

[**6.6. Testing Plan** 29](#_Toc6873019)

[**6.7. Testing Cases** 29](#_Toc6873020)

[Chapter.7 FUTURE ENHANCEMENT 30](#_Toc6873021)

[**7.1 FUTURE ENHANCEMENT** 30](#_Toc6873022)

[CHAPTER.8 CONCLUSION 31](#_Toc6873023)

[**8.1 CONCLUSION** 31](#_Toc6873024)

[Reference:: 32](#_Toc6873025)

List of Figures

[Figure 1 Project Workflow 4](#_Toc6872810)

[Figure 2 Pert Chart 6](#_Toc6872811)

[Figure 3 Use case diagram 10](#_Toc6872812)

[Figure 4 Sequence Diagram for creating the stack 11](#_Toc6872813)

[Figure 5 Sequence Diagram for adding new nodes to the cluster 12](#_Toc6872814)

[Figure 6 Activity Diagram 13](#_Toc6872815)

[Figure 7 Ansible playbook status 26](#_Toc6872816)

[Figure 8 AWS instances created through Ansible 27](#_Toc6872817)

[*Figure 9 Data is searchable from SH: 54.209.127.20 on port 8000.* 27](#_Toc6872818)

[*Figure 10 Indexer clustering (SF/RF met). Cluster status verified through Cluster-master’s UI* 28](#_Toc6872819)

# **CHAPTER.1 INTRODUCTION**

## **PROJECT OVERVIEW**

We will be deploying a splunk cluster which will include indexer, search head, cluster master, license master on AWS using Ec2 instances and we will deploy above mentioned by automation using Ansible.

## **OBJECTIVE**

The purpose of Software Requirements Specification (SRS) is to provide a detailed description of **Deploy Splunk-cluster on AWS using Ansible.**

SRS will give the complete understanding of purpose and its functionality. This document helps developer to understand software correctly as well as it can be used as software validation document for user.

## **SCOPE**

Manual configuration of Splunk on AWS is very tedious job, like we need to create instances, manage SG’s and other network related part. We need to install and setup Splunk on each instance.

This process is very time consuming as well as erroneous because as there is a human intervention, there is a little scope of error for sure. Another reason is for the big cluster having hundreds of instances, this manual work can’t be possible.

This Ansible automation system will take care of this lengthy process by reducing the scope of error thereby atomising this deployment.

## **TOOLS & TECHNOLOGY USED**

The create and understand the whole system we require to get basic knowledge of three main technology

1. Splunk: Software
2. AWS: ASG , S3 ,VPC , EC2
3. Ansible: Automation tool reduce the manual efforts

**Splunk:**

Splunk is a tool which aim to squeeze the important information from unorganized machine data. Splunk provides easy and faster service to search and indexing the unorganized machine data. Splunk has three main components

1. Search-Head:

It is the console where all searching and visualization of data is being performed.

1. Indexer:

It indexes the data so Splunk search it in a faster way. Basically, it the instance where all data get stored.

1. Forwarder

It setup on data generation machine. It will send data to indexers. The data sent by the forwarder is a raw data which is not organized or structured.

Splunk provides highly scalable architecture to fulfils the giant need of searching and indexing the terabytes of data. It provides clustering on search-head and indexers.

Splunk Enterprise monitors and analyses machine data from any source to deliver Operational Intelligence to optimize your IT, security and business performance. With intuitive analysis features, machine learning, packaged applications and open APIs, Splunk Enterprise is a flexible platform that scales from focused use cases to an enterprise-wide analytics backbone.

**AWS:**

AWS means Amazon Web Services. It is rental cloud computing platform which on spot provides elastic resources to fulfil our computing need. With the computing resources, it also provides other services to manage the security and other stuffs.

Some useful services provided by AWS:

1. EC2:

A VM with customized hardware and software requirement.

1. S3:

Storage service to statically manage data

1. VPC:

Virtual Private Cloud which is useful to manage cloud network

1. SG:

Security group which is helpful in restricting the inbound and outbound traffic

**Ansible:**

Ansible is an open source automation platform. It is very, very simple to setup and yet powerful. Ansible can help you with configuration management, application deployment, task automation. It can also do IT orchestration, where you have to run tasks in sequence and create a chain of events which must happen on several different servers or devices.

Ansible don’t need any client to be installed on remote machine. All remote machine is accessed by SSH in Ansible. It also useful in managing services like AWS. So, here we are using Ansible to create AWS instances and configure it with Splunk.

# **CHAPTER.2 PROJECT MANAGEMENT**

## **2.1 PROJECT FLOW**

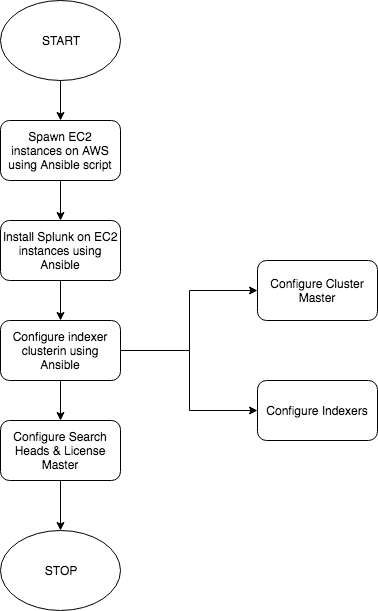


Figure 1 Project Workflow

**Brief Description:**

The above sequence diagram shows how a Splunk cluster is created

1. The Instances are provisioned as per the dynamic count provided by user.
2. Splunk Enterprise is installed on all the instances via Ansible Play.
3. Instances are configured as Indexer.
4. Indexer clustering is established amongst all the indexer associated with one Cluster-Master
5. Search-head and License-Master are configured on ec2 instances
6. Ansible play completes with Splunk cluster running on ec2 instances

### **2.1.2 Project Effort and Time, Cost Estimation**

Estimation technique used: COCOMO model

For evaluating the cost of this project the things which needed to be considered is:-

1. Effort-(how many people)  
2. Time- (duration)

3. Person involved

4. Cost

KLOC = 1.10

Effort = ab\* (KLOC)^bb

= 3.0\*(1.10) ^1.12

= 3.0\*1.1126 PM

= 3.33 PM

Tdev = cb\* (Effort) ^db

= 2.5\*(3.33) ^0.35

= 3.71 Months

Cost = Avg. salary \* Tdev

= 5000 \* 3.71

= 20000 Rs.

* 1. **Project Work Scheduling (Gantt Chart/PERT/Network Chart)**
     1. **Pert Chart Representation**

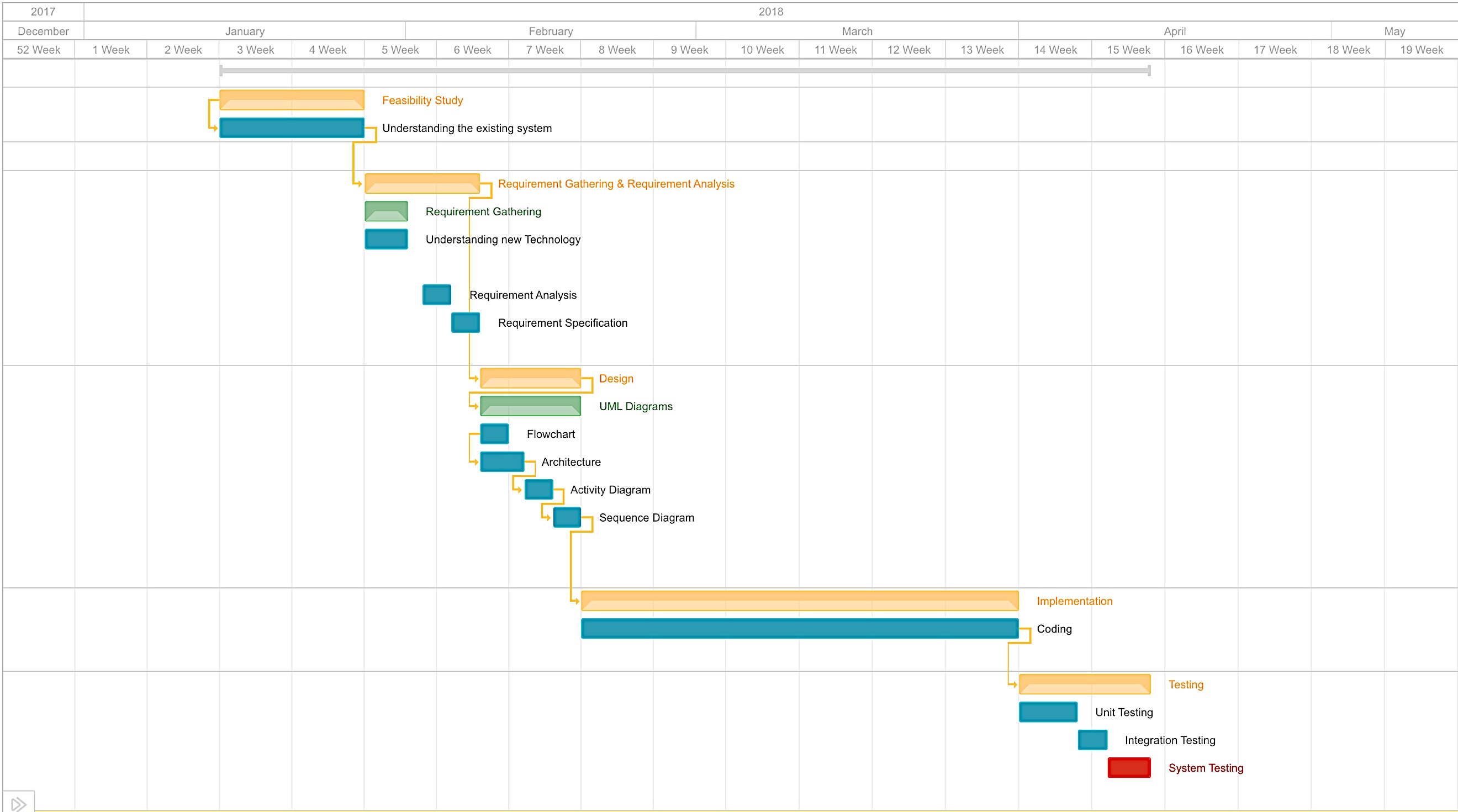
****

Figure 2 Pert Chart

**Brief Description:**

The tasks were not assigned at the beginning of the project. Hence, this is the resultant time allocation regarding the project requirement.

# **CHAPTER.3 SYSTEM REQUIREMENTS STUDY**

## 

## **3.1** **User Characteristics**

This Application focuses on the problems faced by SRE to deploy and manage the Splunk architecture. Below mentioned are the requirements the user must have to be able to use this system:

* Basic knowledge of splunk
* Basic knowledge of linux
* Knowledge of AWS resources
* Should know what is Ansible and how the script works.

## **3.2 Hardware and Software Requirements**

**Software Requirements:**

* O.S : Ubuntu
* Configuration Management Tool : Ansible
* Integration Tool: Jenkins
* Technology : Splunk, AWS EC2 Storage: AWS S3

**Hardware Requirements(Deployment):**

* 1x c3.8xl (search Head) 60 GB RAM,32 vCPU (c stands for Compute optimized)
* 2x m3.l (License master, Cluster master) 7.5 GB RAM, 2 vCPU (m stands for general purpose)
* 3x d2.8xl (Indexers) 244 GB RAM, 36 vCPU (d stands for storage optimized)

## **3.3** **Assumptions and Dependencies**

Cloud SRE must have a basic knowledge to execute ansible plays. He/She must have basic knowledge of how splunk’s clustered environment works. He/She must have basic knowledge of AWS as well.

It is assumed that the machine has a valid AWS credentials (Access key and Secret Key). It has Ansible installed properly. Internet Connectivity.

# **CHAPTER.4 SYSTEM ANALYSIS**

## **4.1** **Product Function**

The user specifically SRE (Site Reliability Engineer) can create Splunk architecture on AWS using single script. SRE can provide options like how many indexers should be there, how many search-head should be there .Even after deployment SRE can modify the system like adding or removing the indexers and search-heads from cluster. SRE can reset the configuration.

## **4.2** **User Characteristics**

This Application focuses on the problems faced by SRE to deploy and manage the Splunk architecture.

## **4.3 Requirements of new System**

### **4.3.1 Functional Requirements**

This section contains all of the functional and quality requirements of the system. It gives a detailed description of the system and all its features.

R1 → Creating the Splunk architecture

Description: It is used to create the Splunk architecture for the first

Input: No. of search-head and indexers on cluster, stack name

Output: Status regarding the formation of Splunk architecture

R2 → Adding indexers to the indexer cluster

Description: It is used to modify the indexer cluster

Input: stack name and number indexer needs to be added

Output: Status of the work

R3 → Removing indexers to the indexer cluster

Description: It is used to modify the indexer cluster

Input: stack name and number indexer needs to be removed

Output: Status of the work

* + 1. **Non-Functional Requirements**

•Reliability

It is the measure of how consistently a computer hardware/software can perform according to its specifications

•Availability

It is the ratio of time a given system is functional to the total time it is expected to be functional

•Serviceability

Set of features that support the ease and speed with which corrective or preventive maintenance can be conducted on a system

•Portability

Ability to obtain and reuse data across different services

•Security

Protecting critical information from theft , data leakage and deletion

## **Use case Diagram**

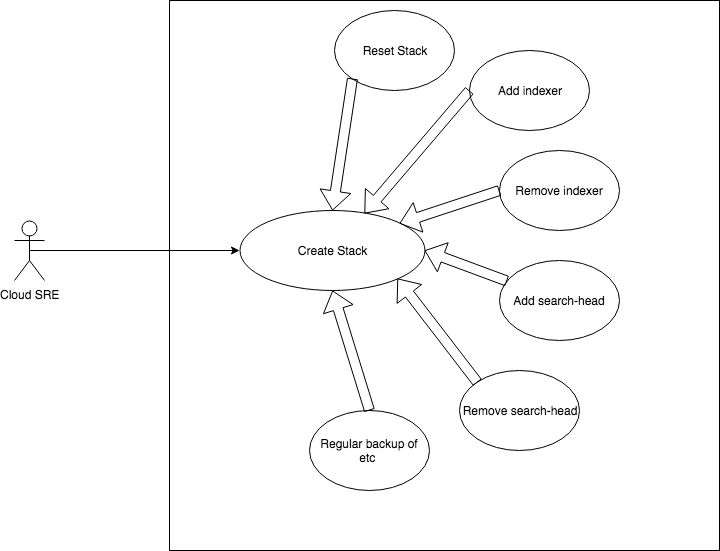
****

Figure 3 Use case diagram

**Brief Description:**

### The above Use case diagram shows the Use cases for a Site Reliability Engineer

1. The Instances are provisioned as per the dynamic count provided by SRE.
2. Splunk Enterprise is installed on all the instances via Ansible Play.
3. Instances are configured as Indexer.
4. Indexer clustering is established amongst all the indexer associated with one Cluster-Master.
5. Search-head and License-Master are configured on ec2 instances.
6. Ansible play completes with Splunk cluster running on ec2 instances.
7. SRE can also reduce the number of indexer or search-head count.

## **Sequence Diagram**

### **4.5.1 Sequence Diagram for creating the stack**

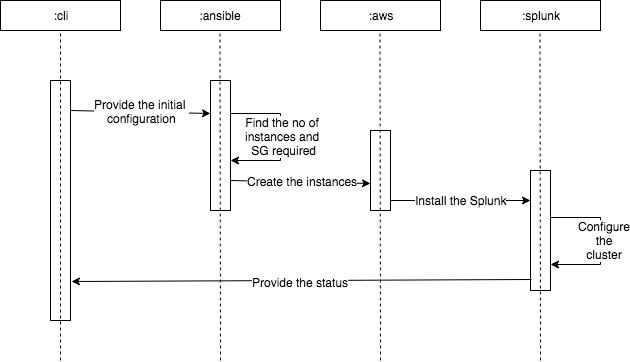


Figure 4 Sequence Diagram for creating the stack

**Brief Description:**

The above sequence diagram shows how the stack is being creating for the first time

Here is the flow sequentially:

1. Initially user will provide all configuration through CLI Ansible playbook
2. Ansible playbook will call AWS module to create the required instances and other things like SG (Security Group)
3. After Instance creation Splunk will be installed on each machine
4. After successful installation Ansible will configure the cluster as per the configuration.
5. Now user will get the status of each task

### **Sequence Diagram for adding new nodes to the cluster**

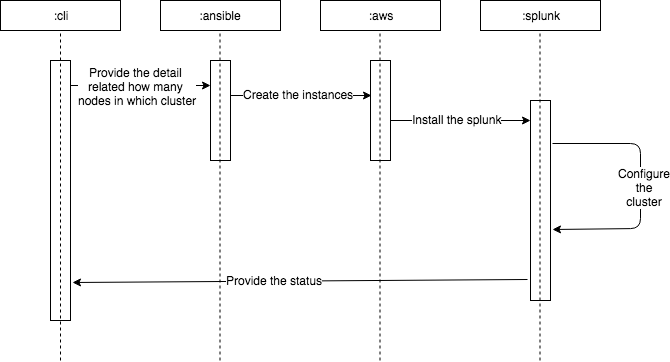


Figure 5 Sequence Diagram for adding new nodes to the cluster

**Brief Description:**

The above sequence diagram shows the interaction for adding node to the cluster

Here is the flow sequentially:

1. User will provide the details of how much node should be added in which cluster (Search-head cluster or indexer cluster)
2. Ansible will call the AWS module to create the instances
3. On this newly created instances Splunk will be installed.
4. After successful installation, the cluster is reconfigured to accept the new node
5. The status will be provided to the user

## **Activity Diagram**

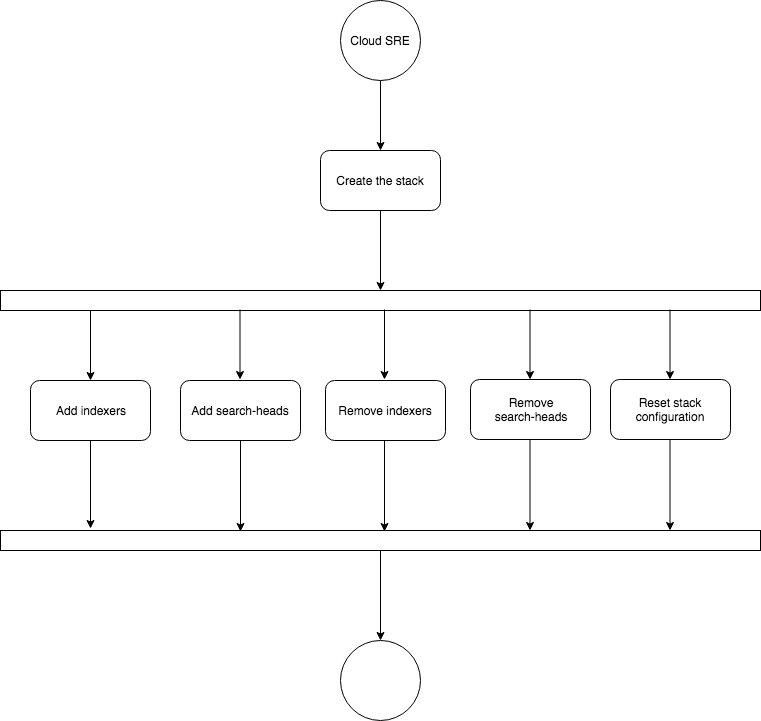


Figure 6 Activity Diagram

**Brief Description:**

The above activity diagram shows how the stack is being creating for the first time

Here is the flow sequentially:

1. Initially user will provide all configuration through CLI Ansible playbook
2. Ansible playbook will call AWS module to create the required instances and other things like SG (Security Group)
3. After Instance creation Splunk will be installed on each machine
4. After successful installation Ansible will configure the cluster as per the configuration.

# **Chapter.5 SYSTEM DESIGN**

## **5.1** **PRODUCT PERSPECTIVE**

Deploy Splunk-cluster on AWS using Ansible is totally independent system.

### **5.1.1 System Interface**

This system will not interact with any types of external interfaces except the CLI.

### **5.1.2 Interface**

This is the command line tool which helps to reduce the manual efforts. So, there is no GUI but only CLI is available to interact with the tool.

### **5.1.3** **Hardware Interface**

The system will run on any basic PC which is connected through Internet and does not require any external hardware interfaces.

### **5.1.4 Software Interface**

Ansible host is the machine on which Ansible is installed and configure with AWS.

### **5.1.5 Communication Interface**

Ansible will communicate with the instances over the internet using SSH protocol

### **5.1.6 Memory Constraints**

There is no specific memory constraint, but system can be best run on machine having primary memory greater than 250 MB.

# **Chapter.6 IMPLEMENTATION AND TESTING**

## **6.1 SPLUNK**

You see servers and devices, apps and logs, traffic and clouds. We see data everywhere. Splunk offers the leading platform for Operational Intelligence. It enables the curious to look closely at what others ignore machine data and find what others never see: insights that can help make your company more productive, profitable, competitive and secure.

Splunk Enterprise monitors and analyses machine data from any source to deliver Operational Intelligence to optimize your IT, security and business performance. With intuitive analysis features, machine learning, packaged applications and open APIs, Splunk Enterprise is a flexible platform that scales from focused use cases to an enterprise-wide analytics backbone.

* Collects and indexes log and machine data from any source
* Powerful search, analysis and visualization capabilities empower users of all types
* Apps provide solutions for security, IT ops, business analysis and more
* Enables visibility across on premise, cloud and hybrid environments

Machine-generated data is one of the fastest growing and complex areas of big data. It's also one of the most valuable, containing a definitive record of all user transactions, customer behaviour, machine behaviour, security threats, fraudulent activity and more. Splunk turns machine data into valuable insights no matter what business you're in. It's what we call Operational Intelligence.

Operational Intelligence gives you a real-time understanding of what’s happening across your IT systems and technology infrastructure so you can make informed decisions. It is enabled by the Splunk platform, the foundation for all of Splunk’s products, premium solutions, apps and add-ons.

Whatever you call it, machine data is one of the most underused and undervalued assets of any organization. But some of the most important insights that you can gain across IT and the business—are hidden in this data: where things went wrong, how to optimize the customer experience, the fingerprints of fraud. All of these insights can be found in the machine data that’s generated by the normal operations of your organization.

Machine data is valuable because it contains a definitive record of all the activity and behaviour of your customers, users, transactions, applications, servers, networks and mobile devices. It includes configurations, data from APIs, message queues, change events, the output of diagnostic commands, call detail records and sensor data from industrial systems, and more.

The challenge with leveraging machine data is that it comes in a dizzying array of unpredictable formats, and traditional monitoring and analysis tools weren’t designed for the variety, velocity, volume or variability of this data. This is where Splunk comes in.

The Splunk platform uses machine data—the digital exhaust created by the systems, technologies and infrastructure powering modern businesses—to address big data, IT operations, security and analytics use cases. The insights gained from machine data can support any number of use cases across an organization and can also be enriched with data from other sources. The enterprise machine data fabric shares and provides access to machine data across the organization to facilitate these insights. It’s what we call Operational Intelligence.

### **6.1.1. Architecture**

There are 3 main components (instances) in Splunk,

* Search-head
* Indexer
* Forwarder

All this component can reside in a single machine as well as each on a different machine. You can get the best results if all the three are on different machines. All this are instances of Splunk means all the machine will have the same Splunk installed only the configuration for each of them will be different.

**Search-head:**

Search head will fire queries which are made using Splunk processing language and this query are made on the indexer. After the query gets executed the data can be visualized in graphical formats like charts, graphs.

**Indexer:**

The data can be injected into Splunk in three ways:

1. By monitoring the files and directories
2. By API calls
3. By Forwarding the files

The data from the various machines will be moved to the indexer. Here the data will be indexed and then stored which then can be searched by the Search Head.

**Forwarder**:

There are two types of forwarder,

* Universal Forwarder

It just moves the data to indexer

* Heavy Forwarder

It not just moves the data but it filters the data. This can be used to avoid not usable data to be indexed.

**Clustering**:

Clustering refers to a group of similar objects here there are search head clusters, indexer cluster.

**Replication Factor:**

When we store the data in indexer if the indexer goes down then you may lose the data. So, for that you can make replication of the data on several indexers. One Copy of data will act as a master copy and other will act as duplicate copy. Search head will first search for data in the master copy.

**Search Factor**:

It is denoting the number of parallel searches can be possible on indexer. It must be less than or equal then the replication factor.

**Cluster master**:

It will hold the data of the indexer's. Which data is stored in which indexer is known to the index master. So, when search head queries for data then it will be first directed to the index master from which it will find which indexer has the data the search head is looking for.

### **6.1.2. Splunk Indexes**

When first hearing about Splunk some think “database”. But that is a misconception. Where a database requires you to define tables and fields before you can store data Splunk accepts almost anything immediately after installation. In other words, Splunk does not have a fixed schema.

Instead, it performs field extraction at search time. Many log formats are recognized automatically, everything else can be specified in configuration files or right in the search expression.

This approach allows for great flexibility. Just as Google crawls any web page without knowing anything about a site’s layout, Splunk indexes any kind of machine data that can be represented as text.

During the indexing phase, when Splunk processes incoming data and prepares it for storage, the indexer makes one significant modification: it chops up the stream of characters into individual events. Events typically correspond to lines in the log file being processed. Each event gets a time-stamp, typically parsed directly from the input line, and a few other default properties like the originating machine. Then event keywords are added to an index file to speed up later searches and the event text is stored in a compressed file sitting right in the file system.

Infinite retention without losing granularity. Some monitoring products only allow you to keep so many months, weeks or even days’ worth of data. Others reduce the granularity of older events, compressing many data points into one because of capacity limits. The same is not true for Splunk. It can literally index hundreds of terabytes per day and keep practically unlimited amounts of data.

**Types of Data Splunk Can Read:**

One of the common characteristics of machine data is that it almost always contains some indication of when the data was created or when an event described by the data occurred. Given this characteristic, Splunk’s indexes are optimized to retrieve events in time-series order. If the raw data does not have an explicit timestamp, Splunk assigns the time at which the event was indexed by Splunk to the events in the data or uses other approximations, such as the time the file was last modified or the timestamp of previous events.

The only other requirement is that the machine data be textual, not binary, data. Image and sound files are common examples of binary data files. Some types of binary files, like the core dump produced when the program crashes, can be converted to textual information, such as a stack trace. Splunk can call your scripts to do that conversion before indexing the data. Ultimately, though, Splunk data must have a textual representation to be indexed and searched.

**Meaning of Index:**

Splunk Enterprise stores all of the data it processes in indexes. An index is a

collection of databases, which are sub-directories located at $SPLUNK\_HOME/var/lib/splunk.

Indexes consist of two types of files: raw data files and index files. Splunk Enterprise can index any type of time-series data (data with time-stamps). When Splunk Enterprise indexes data, it breaks it into events, based on the time-stamps.

**Event processing and the data pipeline**

Data enters the indexer and proceeds through a pipeline where event processing occurs. Finally, the processed data it is written to disk. This pipeline consists of several shorter pipelines that are strung together. A single instance of this end-to-end data pipeline is called a pipeline set.

Event processing occurs in two main stages, parsing and indexing. All data that comes into Splunk Enterprise enters through the parsing pipeline as large (10,000 bytes) chunks.

During parsing, Splunk Enterprise breaks these chunks into events which it hands off to the indexing pipeline, where final processing occurs.

While parsing, Splunk Enterprise performs a number of actions, including:

1. Extracting a set of default fields for each event, including host, source and source type.
2. Configuration character set encoding.
3. Identifying line termination using line breaking rules. While many events are short and only take up a line or two, others can be long.
4. Identifying time-stamps or creating them if they don't exist. At the same time that it processes time-stamps, Splunk identifies event boundaries.
5. Splunk can be set up to mask sensitive event data (such as credit card or social security numbers) at this stage. It can also be configured to apply custom metadata to incoming events.

In the indexing pipeline, Splunk Enterprise performs additional processing, including:

1. Breaking all events into segments that can then be searched upon. You can determine the level of segmentation, which affects indexing and searching speed, search capability and efficiency of disk compression.
2. Building the index data structures
3. Writing the raw data and index files to disk, where post-indexing compression occurs.

The breakdown between the parsing and indexing pipelines is of relevance mainly when deploying forwarders. Heavy forwarders can run raw data through the parsing pipeline and then forward the parsed data on to indexers for final indexing. Universal forwarders do not parse data in this way. Instead, universal forwarders forward the raw data to the indexer, which then processes it through both pipelines. Note, however, that both types of forwarders do perform a type of parsing on certain structured data.

When Splunk software indexes data, it tags each event with a number of fields. These fields become part of the index event data. The fields that are added automatically are known as default fields. Default fields serve a number of purposes:

Internal fields:

* \_raw
* \_time
* \_indextime
* \_cd

These fields contain information that Splunk software uses for its internal processes.

Basic default fields:

* host
* index
* linecount
* punct
* source
* sourcetype
* splunk\_server
* timestamp

These fields provide basic information about an event, such as where it originated, what kind of data it contains, what index it's located in, how many lines it contains, and when it occurred

**Default date-time fields**:

date\_hour, date\_mday, date\_minute, date\_month, date\_second, date\_wday, date\_year, date\_zone

These fields provide additional searchable granularity to event timestamps.

Three basic default fields are as follows:

* Host

A default field that contains the host name or IP address of the network device that generated an event. Each event has a host field. The indexer generates the host field at index time. You use the host field in searches to narrow the search results to events that originate from a specific device.

You can configure host values for events when events are input into Splunk Enterprise. You can set a default host for a Splunk Enterprise server, file, or directory input. You can also have Splunk Enterprise assign host values to events based on data in those events.

* Source

A default field that identifies the source of an event, that is, where the event originated. In the case of data monitored from files and directories, the source consists of the full path name of the file or directory. In the case of a network-based source, the source field consists of the protocol and port, such as UDP:514.

Each event has a source field. The indexer generates the source field at index time. Searches often use the source as a criterion

* Source-type

A default field that identifies the data structure of an event. A source type determines how Splunk Enterprise formats the data during the indexing process. Splunk Enterprise comes with a large set of predefined source types, and it assigns a source type to your data. You can override this assignment by assigning an existing source type or creating a custom source type.

The indexer identifies and adds the source type field when it indexes the data. As a result, each indexed event has a source-type field. Use the source-type field in searches to find all data of a certain type (as opposed to all data from a certain source).

**Default set of indexes**:

Splunk Enterprise comes with a number of preconfigured indexes, including:

main:

This is the default Splunk Enterprise index. All processed data is stored here unless otherwise specified.

\_internal:

Stores Splunk Enterprise internal logs and processing metrics.

\_audit:

Contains events related to the file system change monitor, auditing, and all user search history.

A Splunk Enterprise administrator can create new indexes, edit index properties, remove unwanted indexes, and relocate existing indexes. Splunk Enterprise administrators manage indexes through Splunk Web, the CLI, and configuration files such as indexes.conf.

**Indexer Workflow to store Indexes:**

As the indexer indexes your data, it creates a number of files. These files contain two types of data:

* The raw data in compressed form (raw data)
* Indexes that point to the raw data, plus some metadata files (index files, also known as tsidx files)

Together, these files constitute the Splunk Enterprise index. The files reside in sets of directories organized by age. Some directories contain newly indexed data, others contain previously indexed data. The number of such directories can grow quite large, depending on how much data you're indexing.

**Data Aging**:

Each of the index directories is known as a bucket. To summarize so far:

* An "index" contains compressed raw data and associated index files.
* An index resides across many age-designated index directories.
* An index directory is called a bucket.
* A bucket moves through several stages as it ages:
  + Hot
  + Warm
  + Cold
  + Frozen
  + Thawed
* As buckets age, they "roll" from one stage to the next. As data is indexed, it goes into a hot bucket. Hot buckets are both searchable and actively being written to.
* An index can have several hot buckets open at a time
* When certain conditions occur (for example, the hot bucket reaches a certain size or Splunk service gets restarted), the hot bucket becomes a warm bucket ("rolls to warm") and a new hot bucket is created in its place. Warm buckets are searchable, but are not actively written to. There are many warm buckets
* Once further conditions are met (for example, the index reaches some maximum number of warm buckets), the indexer begins to roll the warm buckets to cold, based on their age.
* It always selects the oldest warm bucket to roll to cold
* Buckets continue to roll to cold as they age in this manner
* After a set period of time, cold buckets roll to frozen, at which point they are either archived or deleted. By editing attributes in indexes.conf, you can specify the bucket aging policy, which determines when a bucket moves from one stage to the next
* If the frozen data has been archived, it can later be thawed. Thawed data is available for searches
* The collection of buckets in a particular stage is sometimes referred to as a database or "db": the "hot db", the "warm db", the "cold db", etc

**Index Directories:**

Each index occupies its own directory under $SPLUNK\_HOME/var/lib/splunk. The name of the directory is the same as the index name. Under the index directory are a series of sub-directories that categorize the buckets by stage (hot/warm, cold, or thawed). The buckets themselves are sub-directories within those directories. The bucket directory names are based on the age of the data.

## **6.2. Ansible Playbooks**

Ansible playbooks is the YAML file which helps to interact with Ansible. It is a script which has instructions written to perform specific task. According to the logic and instructions given on the playbook Ansible will SSH to the node and perform specific task.

## **6.3. System Study**

* AWS provides on spot instances creation which is helpful while deploying the Splunk architecture and modifying the cluster
* Ansible provide the orchestration of the task being performed on each Linux machine
* Splunk is the tool which is deployed on AWS

## **6.4. Implementation Of Application**

The implementation is divided into two parts

* Deploying the stack for the first time
* Modifying the cluster

**Deploying the cluster for the first time**:

This is the script which do below tasks:

* Creating the AWS instances
* Installing Splunk
* Configure indexer cluster

**Modifying the cluster**:

This script is used to modify the search-head cluster and indexer cluster. Add or remove the node from the cluster is done through this script.

For removing the indexer from the cluster, it requires to stabilise the cluster before removing that indexer. So, it takes some time to coping data from that indexer to the other indexers. After that it can be decommissioned.

## **6.5. Screenshots**

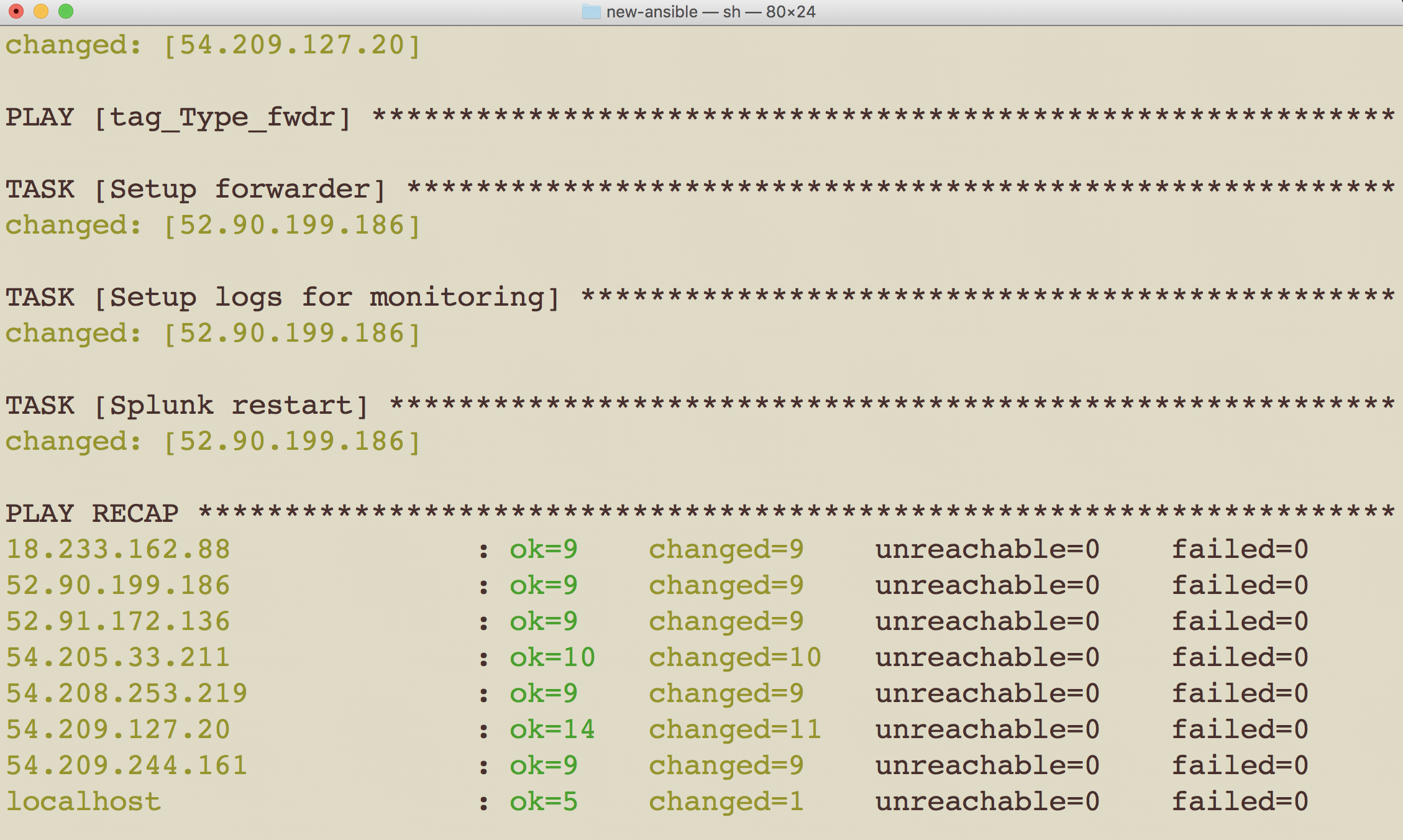


Figure 7 Ansible playbook status

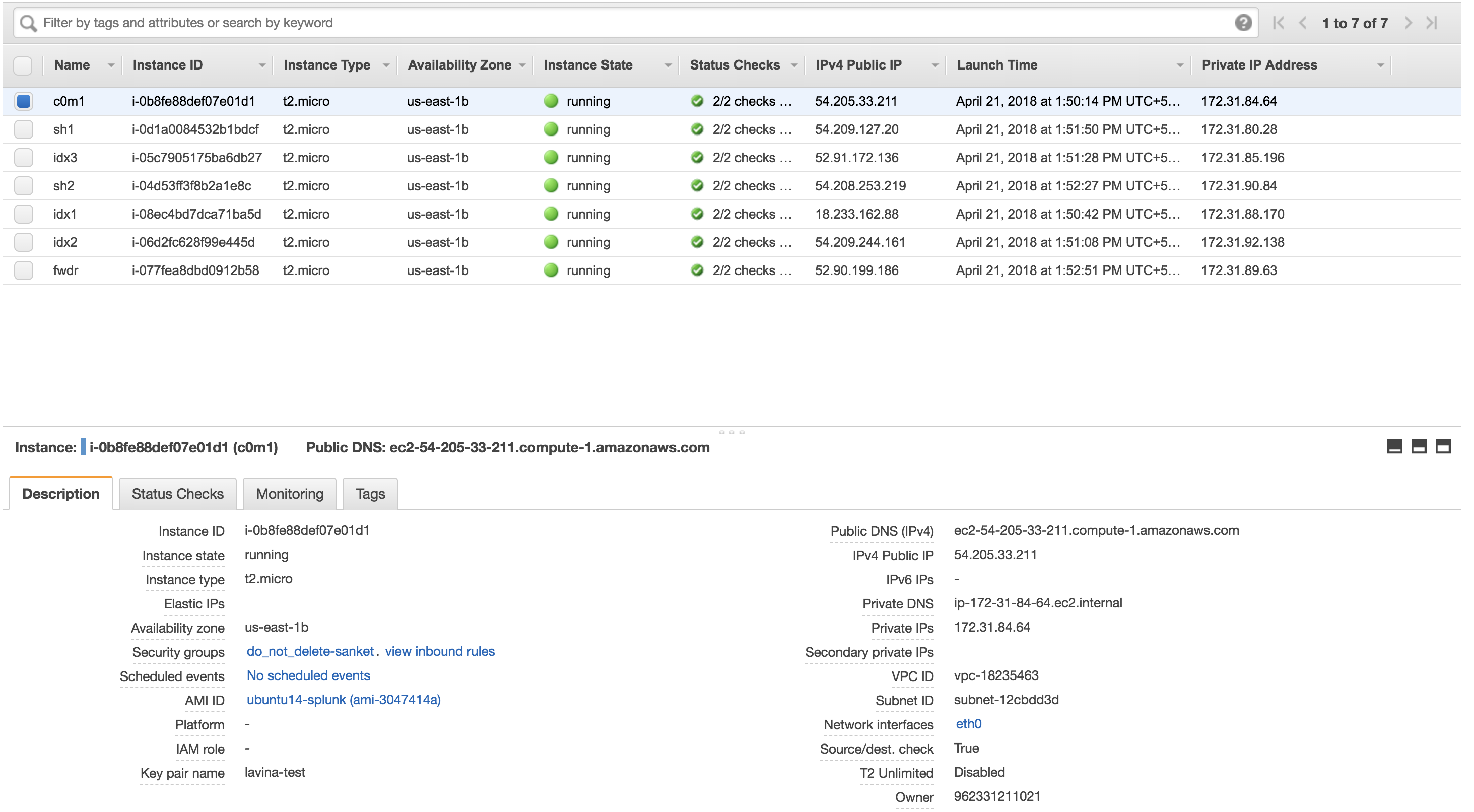
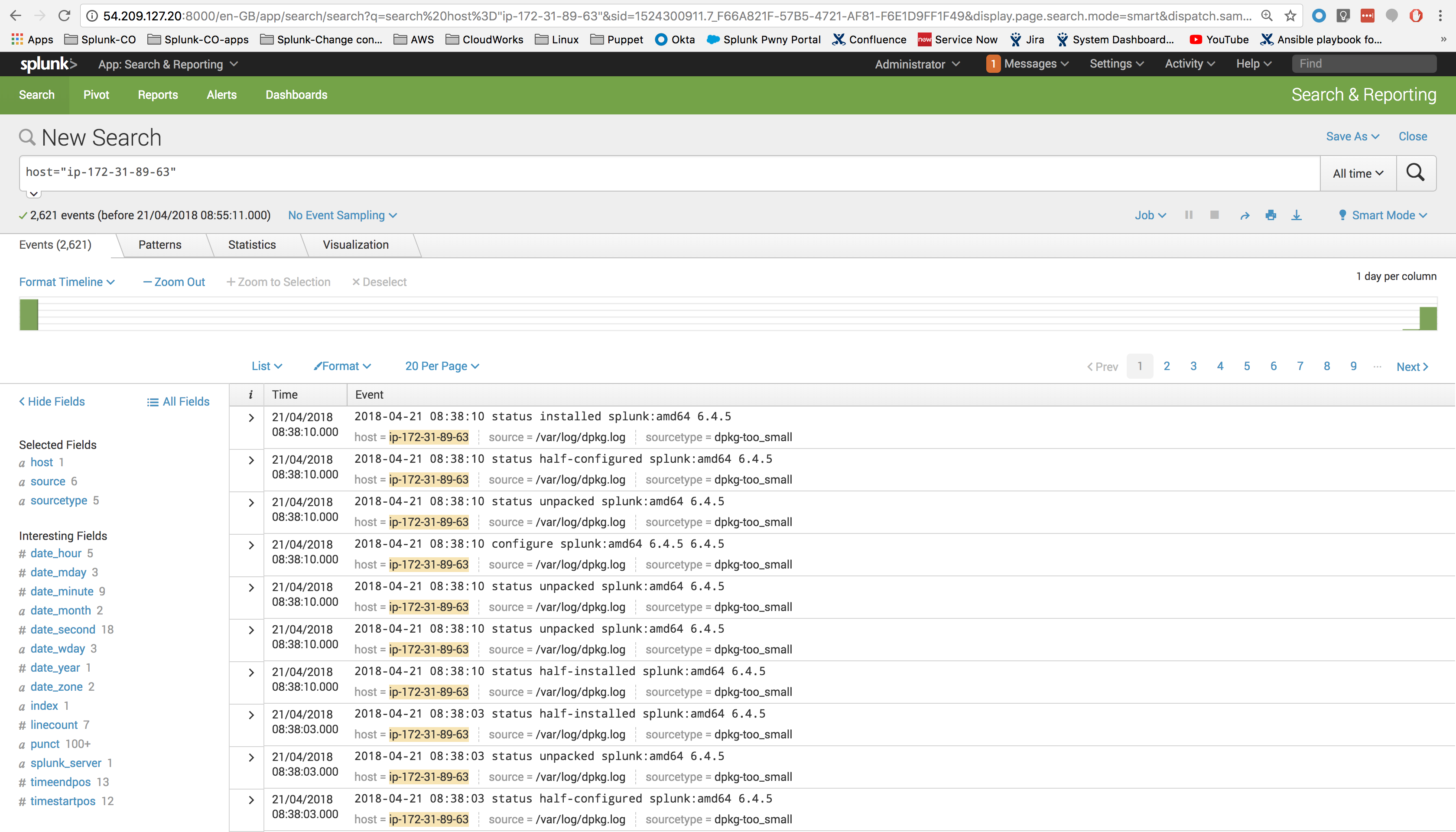
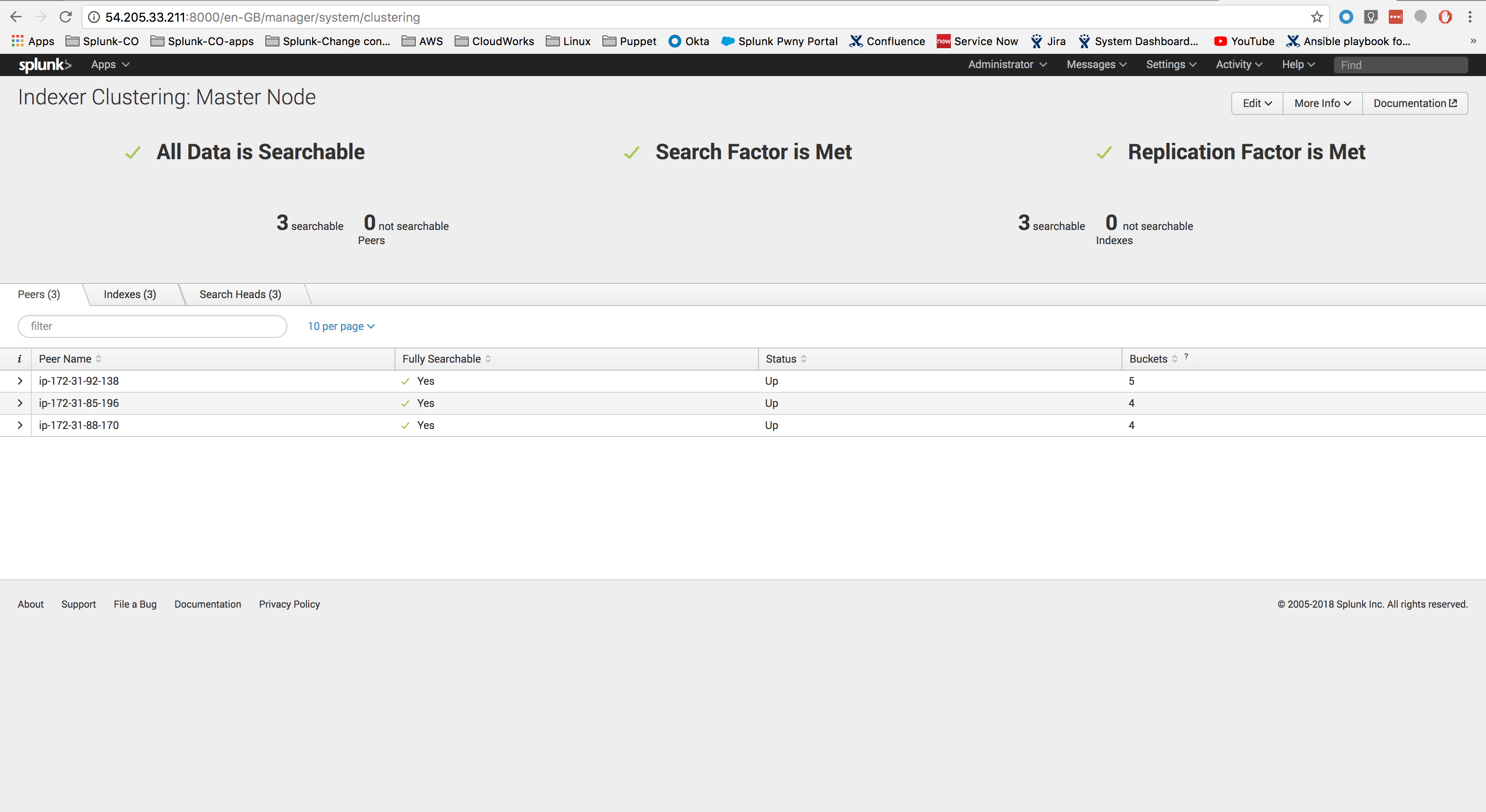


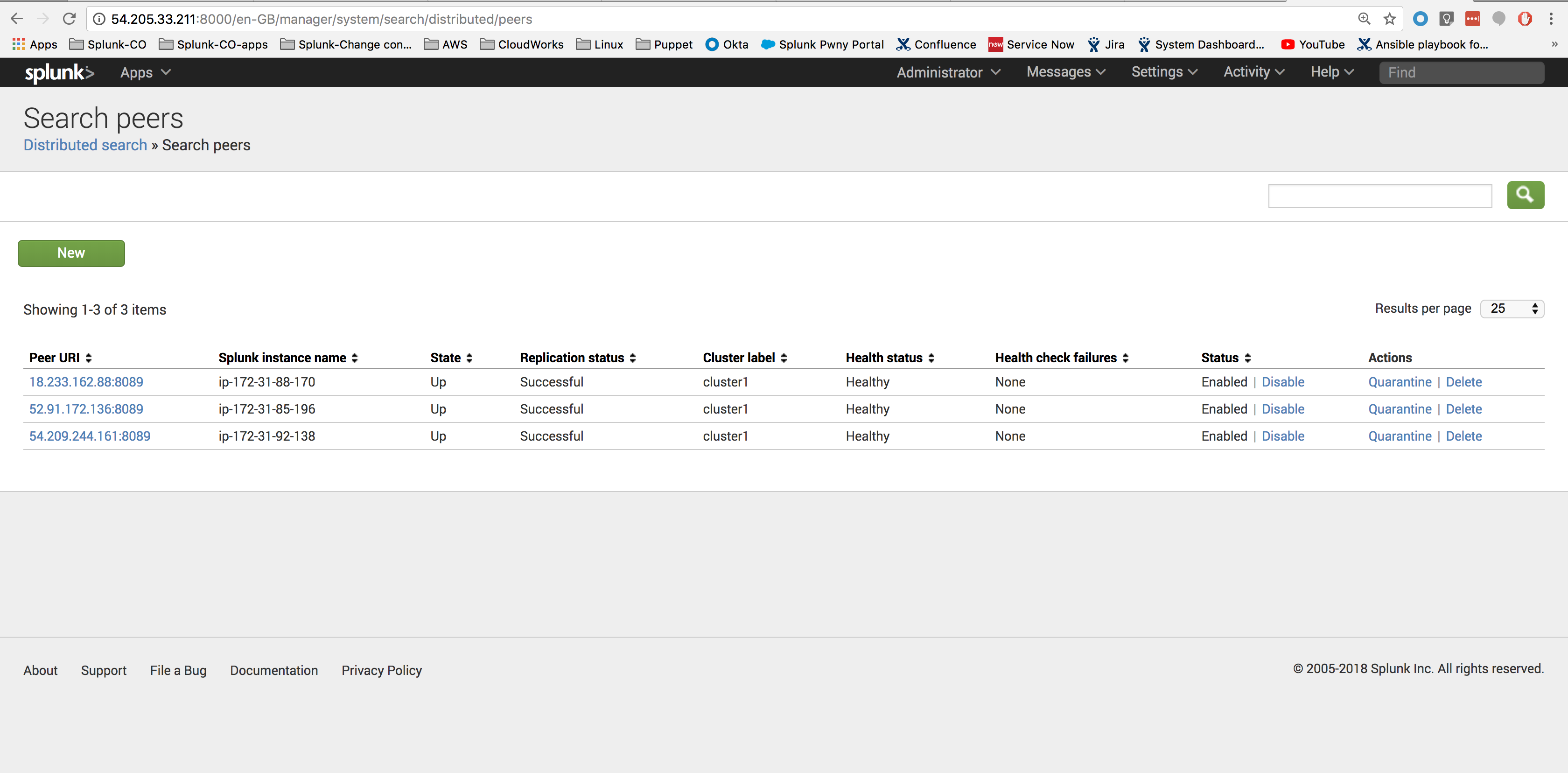
Figure 8 AWS instances created through Ansible



*Figure 9 Data is searchable from SH: 54.209.127.20 on port 8000.*



*Figure 10 Indexer clustering (SF/RF met). Cluster status verified through Cluster-master’s UI*



*Distributed search shows the status of SH’s*

## **6.6. Testing Plan**

The testing technique that has been used in the project is black box testing, that is expected inputs to the system are applied and only the outputs are checked.

## **6.7. Testing Cases**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Test case Name** | **Test case**  **description** | **steps** | **Expected**  **result** | **Actual result** |
| 1 | Install Splunk and node creation | The instances should be spawn on AWS and on each instance Splunk should be installed | Run the below command:  *ansible-playbook main.yml* | On AWS console, we could able to see the all instances and we could able to see the Splunk UI on [*https://public\_ip:8000*](https://public_ip:8000)of each instance | We can see the instances on AWS console and the Splunk UI is accessible of each instance |
| 2 | Cluster creation | The Splunk cluster should be created including indexer cluster and search-head cluster | Run the below command:  *ansible-playbook main.yml* | On cluster master UI,  We could able to see the indexer cluster and search-head cluster | We can see the indexer cluster and search-head cluster on cluster master UI |
| 3 | Add indexer to indexer cluster | The indexer node should be created and should be added to the indexer cluster | Run below command:  *ansible-playbook add\_idx.yml -e “act=add idx\_name=idx#”*  here # means the new indexer number | The new indexer should be seen on cluster master UI | We are able to find the new indexer on cluster master UI |
| 4 | Add search-head | The search-head node should be created and should be added to the indexer cluster | Run below command:  *ansible-playbook add\_sh.yml -e “act=add sh\_name=sh#”*  here # means the new indexer number | The new search-head should be seen on cluster master UI | We are able to find the new search-head on cluster master UI |

# **Chapter.7** **FUTURE ENHANCEMENT**

## **7.1 FUTURE ENHANCEMENT**

* It can be enhanced by including other automation tools like terraform and puppet.
* Bastion hosts can be introduced.
* Monitoring system like Nagios can be configured.
* Auto-scaling group can be configured for indexer cluster.
* GUI interface can be created for performing each task.

# **CHAPTER.8 CONCLUSION**

## **8.1 CONCLUSION**

The proposed system will work efficiently for deploying the Splunk-cluster on AWS using Ansible. It mainly focuses on reducing the manual efforts.

Using an automation script is a better approach rather than manually configuring all things because manual task has always scope of some mistakes and is also time consuming

# 

# **Reference::**

* **Splunk Documentation:** <https://docs.splunk.com/Documentation>
* **Splunk troubleshooting:** <https://answers.splunk.com/>
* **Ansible Documentation:** <http://docs.ansible.com/>
* **AWS Documentation:** <https://aws.amazon.com/documentation/>