A

Project Report On

Ubuntu Upgradation Using Ansible/Splunk

Prepared by:

Vasu Patel (CE088)

15CEUON018

A project is submitted

in

partial fulfilment of the requirements

for the degree of

BACHELOR OF TECHNOLOGY

in

Computer Engineering

Internal Guide

Dr. C K Bhensdadia

Head,

Dept. of Comp. Engg.

External Guide

Mr. Keyur Barapatre

CloudOps SRE

Crest Data System

Faculty of Technology

Department of Computer Engineering

Dharmsinh Desai University

April 2019

CANDIDATE'S DECLARATION

I declare that final semester report entitled "**Ubuntu Upgradation Using Ansible/Splunk**" is my own work conducted under the supervision of the external guide "Keyur Barapatre" from Crest Data Systems.

I further declare that to the best of my knowledge the report for B.Tech. final semester does not contain part of the work which has been submitted for the award of B.Tech. Degree either in this or any other university without proper citation.
Also, I declare that following students also worked in this project:

Jaydev Patel (IT - 02)
Department of IT, DD University

Candidate's Signature
Vasu Patel
Branch:CE Student ID: 15CEUON018

CERTIFICATE

This is to certify that the project work titled

"**Ubuntu Upgradation Using Ansible/Splunk**"
is the bonafide work of

Vasu Patel

15CEUBS038

carried out in the partial fulfilment of the degree of Bachelor of Technology in Computer Engineering at

Dharmsinh Desai University in the academic session

December 2018 to April 2019

Dr. C. K. Bhensdadia
Head,
Dept. of Computer Engg.

Faculty of Technology
Department of Computer Engineering
Dharmsinh Desai University
April - 2019

ACKNOWLEDGEMENT

I would sincerely like to thank Prof. C K Bhensdadia (Department of Computer Engineering) for the unconditional and an unbiased support during the whole session of study and development and also for guiding us throughout the whole internship period. They all together provided us a favorable environment, without them we would not have achieved our goal. He had always been there for us despite his busy schedule and was always a great source of inspiration himself. He had been easily approachable during and even after college hours. We sincerely thank him for that.

We would also like to thank Mr. Keyur Barapatre, Site Reliability Engineer (Crest Data System Pvt. Ltd.) for supporting us during whole internship period. Being with us all time face to face and guiding us within busy time schedule of him. We would like thank him for that. We would also like to thank our other senior colleagues who also helped us whenever we required any kind of support.

A blend of gratitude, pleasure and great satisfaction is what we feel to convey our indebtedness to all those who have directly and indirectly contributed to the successful completion of the project.

With sincere regards,
Vasu Patel
(ID-15CEUON018)

Table of Contents

ABSTRACT

Amazon Web Services (AWS) is a comprehensive, evolving cloud computing platform provided by Amazon. It provides a mix of infrastructure as a service (IaaS), platform as a service (PaaS) and packaged software as a service (SaaS) offerings. More than 100 services comprise the Amazon Web Services portfolio, including those for compute, databases, infrastructure management, application development and security.

Ansible is an IT automation tool. It can configure systems, deploy software, and orchestrate more advanced IT tasks such as continuous deployments or zero downtime rolling updates.

Splunk Enterprise is a software platform to search, analyze and visualize the machine-generated data gathered from the websites, applications, sensors, devices etc. which make up your IT infrastructure and business.

Splunk indexes machine data. This includes data streaming from packaged and custom applications, application servers, web servers, databases, networks, virtual machines, telecoms equipment, operating system, sensors and so on, that make up your IT infrastructure.

As we all know today there are lots of data generated by human. In order to monitor and analyze the data Splunk architecture is very useful and we also well aware about Ubuntu. It is very tedious task to upgrade these both systems. The project deals with upgradation of Ubuntu and Splunk.

List of Figures

List of Tables

1.0 INTRODUCTION

1.1 Purpose

As the computers and electronics devices has become major in the future. Automation and its implementation for various projects very popular field nowadays. Ansible and Splunk is the one of the tools that does same.

Purpose of this project is that it is very important to test how automation works using various tools and make mankind easy. We can write scripts in several ways and using several tools which makes automation possible.

1.1 Scope

This project is aimed for the operations engineer to automate his tasks for the upgradation of the Splunk Cloud cluster without any manual interference. It achieves the task of upgrading the Splunk cloud software for the instances on Amazon Web Services and upgrading the underlying Ubuntu OS (if needed) successfully. It will be really handy for the case when the upgrades are to be performed on instances which can be in big numbers (and in some cases it can be a few hundred instances). It also helps to back the instances with EBS, another service provided by Amazon Web Services, which helps to recover the data back if the instance fails (due to hardware or OS-level errors).

1.2 Definitions

AWS – Amazon Web Services

AWS Cloud Platform - A container image is a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it: code, run time, system tools, system libraries, settings

Terminal – Command Line Interface

User - A person who uses the system

CloudOps - Ensures that the applications and data stored in cloud are managed properly, to function well over a long period of time.

1.3 Objective

This automation will reduce the burden of user , instead of manually upgrading the instance having splunk installed with specific version and then upgrading Ubuntu OS installed with any version. So this automation script will make easy for user to just execute the script with proper credentials and hence its all done.

2.0 PROJECT MANAGEMENT

2.1 Feasibility Study

Once scope has been identified, it is reasonable to ask whether we can build software that meets this scope. Is this project feasible?

The feasibility of software can be tested in four dimensions:

2.1.1 Technical Feasibility

What we have planned to implement is technically feasible. Do we have sufficient amount of knowledge or technology to make it reality?

And the answer is fairly easy, because we have found out that we can use Splunk to index any data and visualize it. So we can also index query performance back to Splunk. We have tried to maintain the coordination between development, design and testing. Now we can conclude that the system is technically feasible.

2.1.2 Time Schedule Feasibility

We checked whether our system can be ready in time without any error. We have planned all its phase keeping the aspect in our mind, that if we find any bug or error after testing phase then we can move our deadline to 2-4 days, as we set our deadline before the actual submission date to the client It requires a minimum of 3 months for the implementation of the complete project with all the features implemented. This also includes the testing and debugging phase.

2.1.3 Operational Feasibility

What How the project will work and who will use it, all such concerns arise in this phase. We have to study what the existing system's problem is, and is it worth solving or not. As testing of Upgradation mostly time and brain used while testing query execution performance.

2.1.4 Implementation Feasibility

User specifically System admin can monitor the following performance parameters via dashboards so as to increase the feasibility of the machine:

- Upgrade Ubuntu
- Upgrade Splunk
- Make EBS backed

2.2 Project Planning

In the development of this project, we will first check to see if our project is feasible functionally, technically and economically. Then we collect the requirements. Hence, we gather all the requirements which we need to develop our system. Then, after thoroughly understanding the requirements, we will start development. Our development process divides basically into two parts: Upgrade Splunk Cluster and then Upgrade Ansible.

2.2.1 Project Development Approach

The Agile Model is used for the project development. We have selected Agile. Model because of its beneficial speed without effecting quality of product and agile makes team so much more productive.

2.2.2 Milestones and Deliverables

Agile model believes that every project needs to be handled differently and the existing method need to tailored to the best suit the project requirements. In agile the tasks are divided to time boxes

2.3.3 Roles and Responsibilities

Name	Analysis	Designing	Coding	Testing	Documentation
Jaydev	✓	✓	✓	✓	✓
Vasu	✓	✓	✓	✓	✓

2.4.4 Project Scheduling Chart

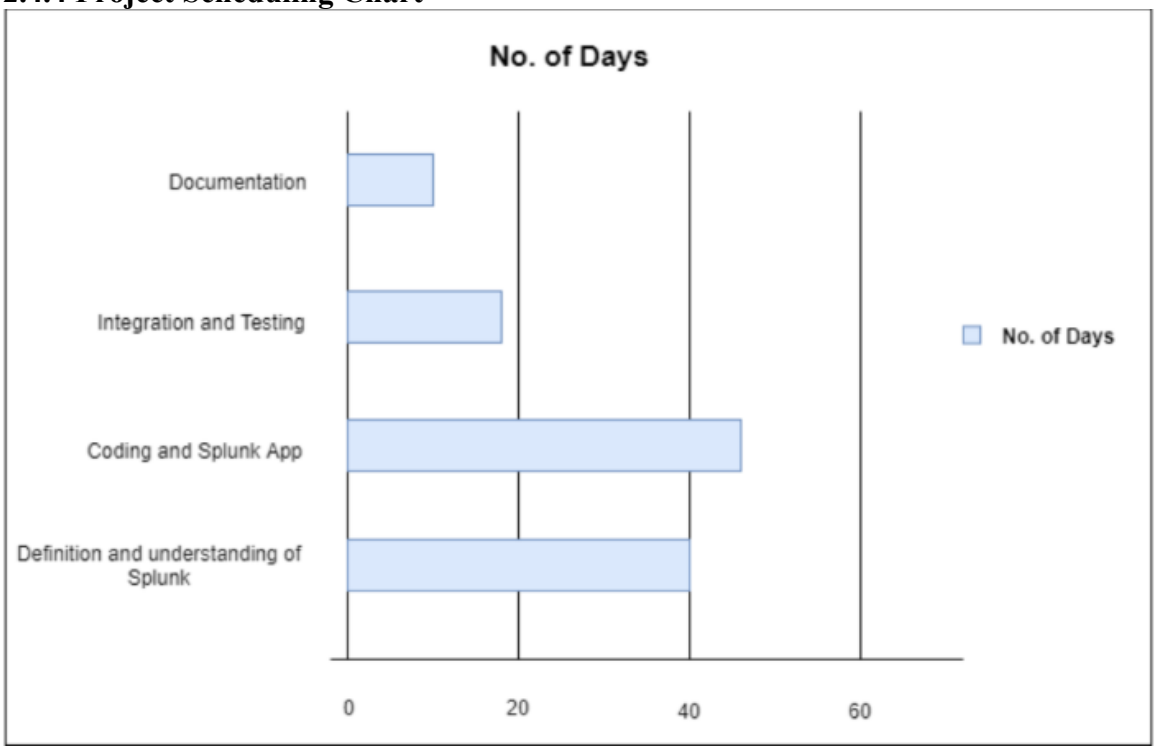


Fig 2.1 Project Chart

3.0 SYSTEM REQUIREMENT STUDY

3.1 User Characteristics

This scripts specifically does upgradation of splunk enterprise software and ansible on cloud infrastructure which was already setup. Upgrades are performed to increase the processing capability of splunk enterprise and to generate results quickly.

3.2 Assumption and Dependencies

One assumption about the system is that it will always be used on a machine that have enough processing capability to run splunk enterprise.

3.3 Hardware and Software Requirements

Here , for this project we require one PC with minimum of 8Gb of Ram and 32GB of Disk Space . We require AWS account with all the free-tier criteria. The AWS account will have Ubuntu 16.04 OS .No special other hardware and software requirements required.

3.4 Constraints

- Splunk (older or latest) must be already installed on cloud instances.
- All status checks must be passed of cloud instances.
- Must able to do SSH to cloud instance.

3.4.1 Product Perspective

The Perform Upgrades on Existing Splunk Cloud Infrastructure is an independent stand–alone system. It is totally self-contained.

3.4.2 System Interfaces

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

3.4.3 Interfaces

The Perform Upgrades on Existing Splunk Cloud Infrastructure shall provide a simple terminal interface and it will display basic progress during procedure. One needs to have access to splunk account in order to see update.

3.4.4 Hardware Interfaces

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

3.4.5 Software Interfaces

- Server side
  This is the machine on which scripts will run. It will act as deployer for deploying of new software

  Simple Terminal will show messages related to ansible scripts/upgrade procedure Ansible needs to be installed on this machine.

- Client side
  Splunk needs to be already installed and check the splunk version through UI.Make sure data is intact after upgrade.

3.4.6 Communication Interfaces

Upgrades on Existing Splunk Cloud Infrastructure uses Wide Area Network (WAN) to maintain communication with all its Splunk instances. The OpenSSH utility will be used to facilitate communication between the client and server

3.4.7 Memory Constraint

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

4.0 SYSTEM ANALYSIS

4.1 System 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 → Perform Pre-checks of existing cluster

Description: The utility can perform pre-checks of the system and decide in which way the upgrade will be performed

Input : Select the instance you want to perform upgrades on
Output : Shows the health of the instance and show which method will be used to perform the upgrade

R2 → Take backups of the existing system

Description: The utility will perform backups of the existing system and will upload it to Amazon S3

Input : None
Output: Backup of the required files on to Amazon S3 bucket

R3 → Determine the instance type and perform upgrade accordingly

Description: The utility will determine whether the instance is a search head, indexer or Cluster-master and will perform upgrades accordingly
Input: None
Output : Shows the instance type and perform upgrades accordingly

R4 → Perform upgrade of Splunk incrementally

Description: The utility will perform upgrade on Splunk software based on the version installed and will upgrade the same to the next version

Input: None
Output : Shows the current version and the version in which it will upgrade to

R5 → Perform upgrade of Ubuntu OS incrementally
Description: The utility will perform upgrade on Ubuntu OS based on the version installed and will upgrade the same to the next version.

R6 → Restore the backup post-upgrade
Description: The utility will restore the backup taken pre-upgrade from S3 post upgrade
Input : None
Output : The files restored at respective locations post upgrade

R7 → Create an EBS volume
Description: The utility will create an EBS volume for persistence of the data from the EC2 instance

Input : None
Output : The ID of the EBS volume created

R8 → Attach EBS volume if not attached already
Description: The utility will attach the created EBS volume onto the root directory of the EC2 instance and sync the data from the root volume to the EBS

Input: Select Docker Top dashboard from panel and select hostname, container name and process ID you want to monitor

Output : The volume attached to the instance and files synced successful

4.2 Use-Case Diagram

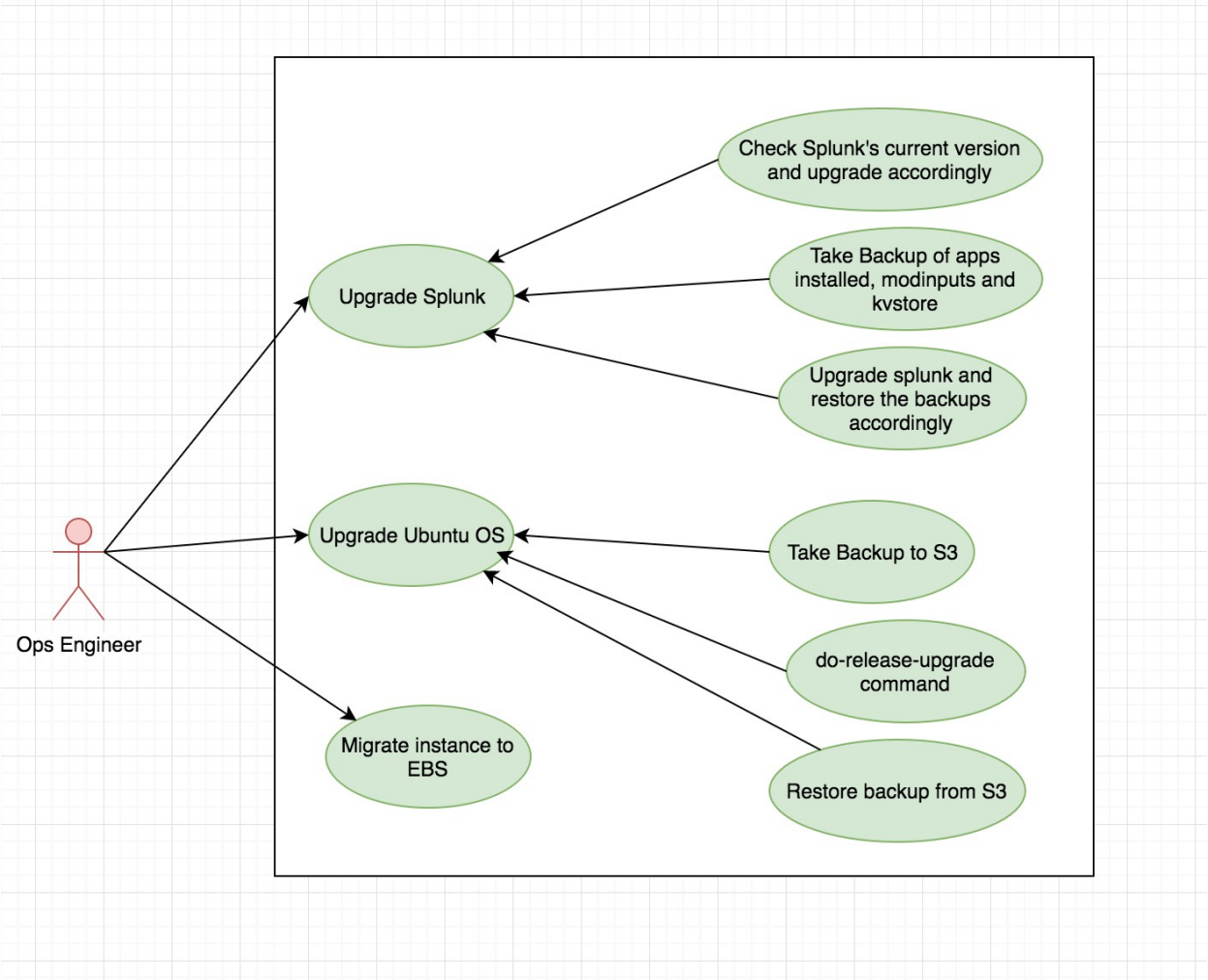


Fig 4.1 Use Case Diagram

4.3Sequence Diagrams

4.3.1 Sequence Diagram for Splunk upgrade

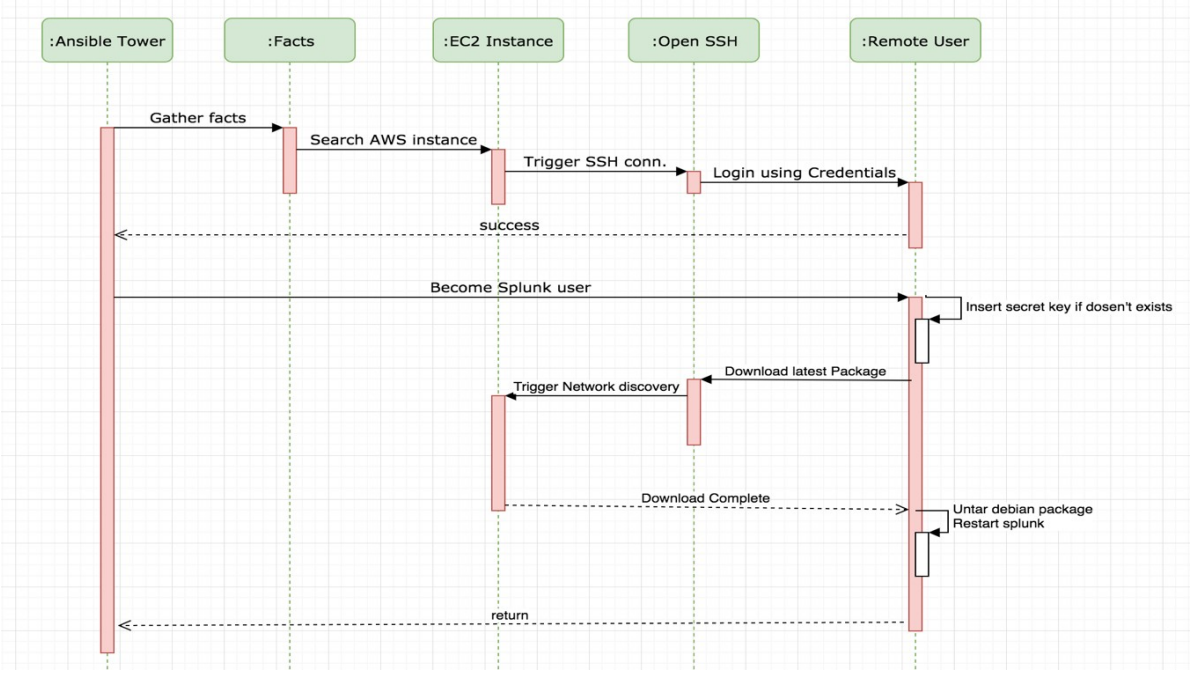


Fig 4.2 Sequence Diagram for Splunk Upgrade

Brief Description:

The above sequence diagram show the 12ebian12ized flow of splunk upgrade.
Here is the flow sequentially:

1. Initially Ansible Tower will gather facts related to its hosts on which commands are going to be fired.
2. Login as splunk user.
3. Insert the secret key which is user for clustering of splunk, if it doesn't exist.
4. Then it will trigger download of latest version of splunk which will be handled by AWS Networking framework. It will discover the splunk package and download it as 12ebian package
5. Downloaded package must be untared as a splunk user so no need to change ownership of splunk directory.
6. After completion of whole tasks it will send success status to Ansible tower.

4.3.2 Sequence Diagram for making EBS backed instance

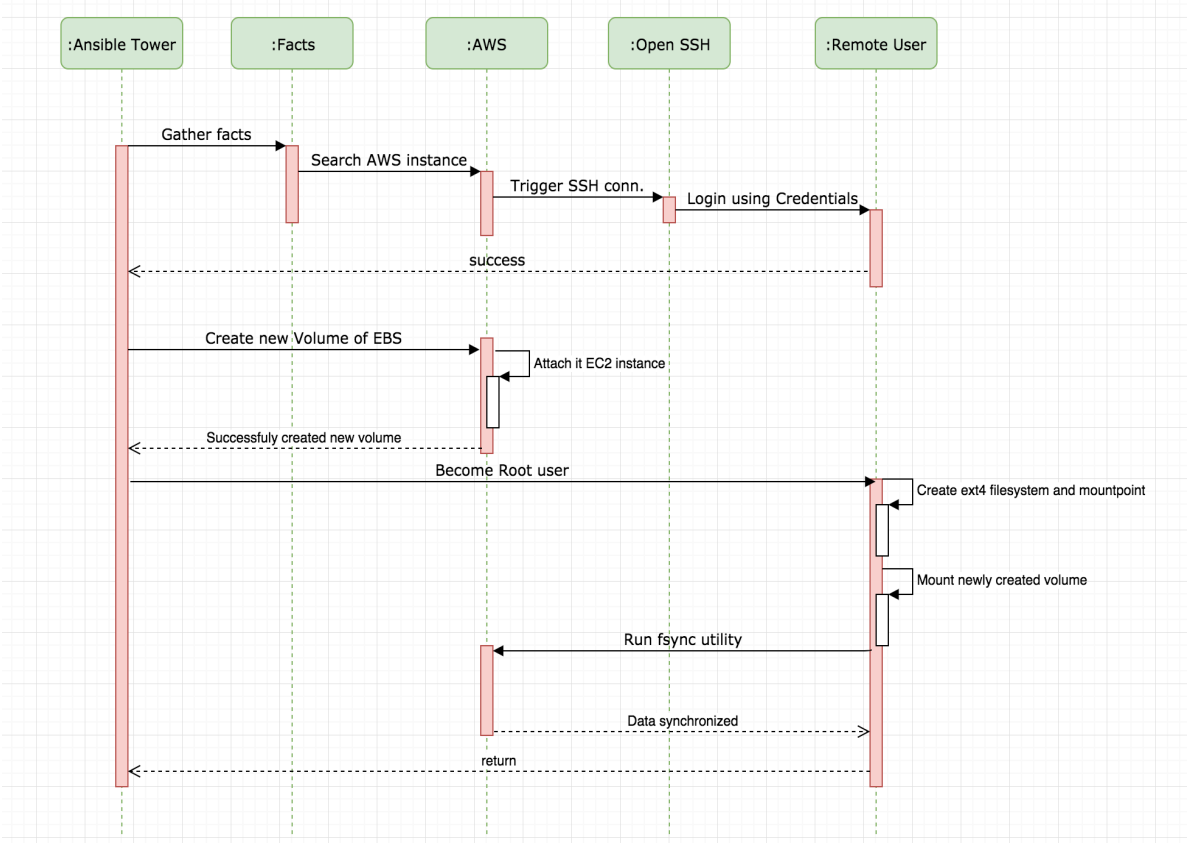


Fig 4.3 Sequence Diagram for making EBS backed instance

Brief Description:

The above sequence diagram shows the creation of new volume and attaching it to the instance.

Here is the flow sequentially:

1. Check SSH connection to instance.
2. Create new EBS volume
3. After this login as root user and create ext4 filesystem and also mount point for it.
4. Run fsync utility to synchronize data between newly created volume and root volume of the instance. Thus, this volume can be attached to other instance without loss of any data.

4.3.3 Sequence Diagram for Ubuntu Upgrade

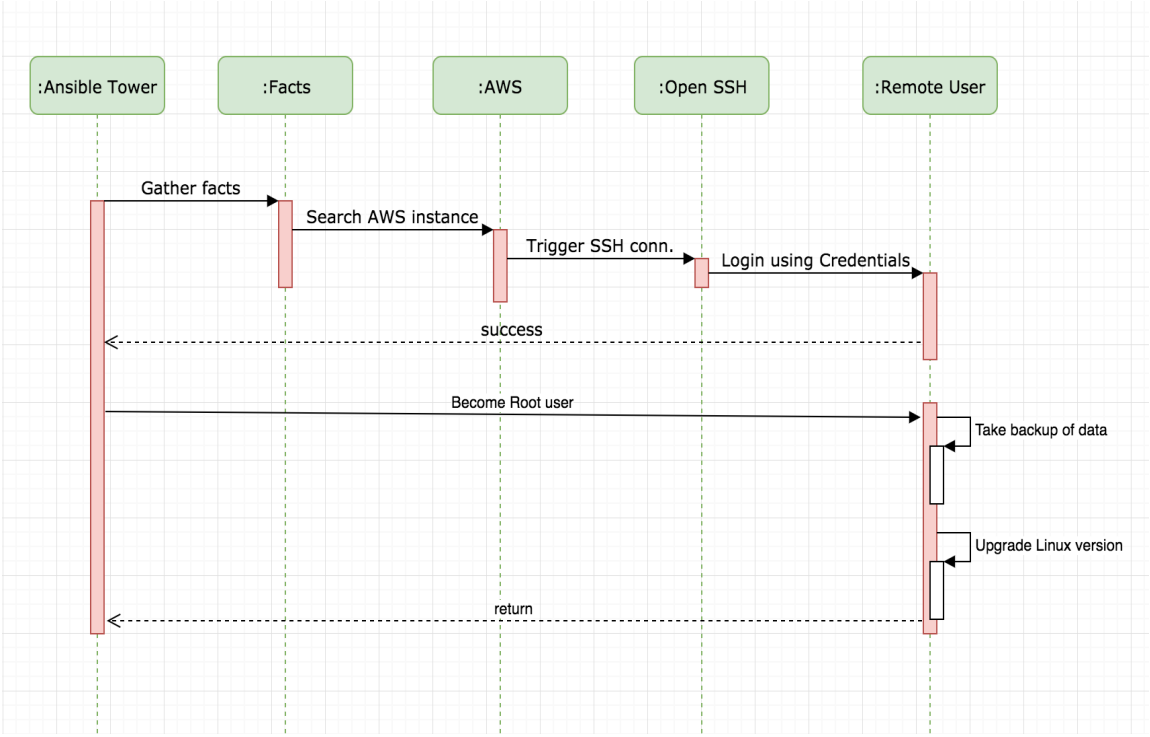


Fig 4.4 Sequence Diagram for Ubuntu Upgrade

Brief Description:

The above sequence diagram shows the interaction for getting the CPU details for a particular container.

Here is the flow sequentially:

1. Check for SSH connection
2. Take backup of specified directories
3. Upgrade Linux version
4. It will return with status of upgrade to Ansible tower.

.

4.3.4 Sequence Diagram for Overall upgrade procedures

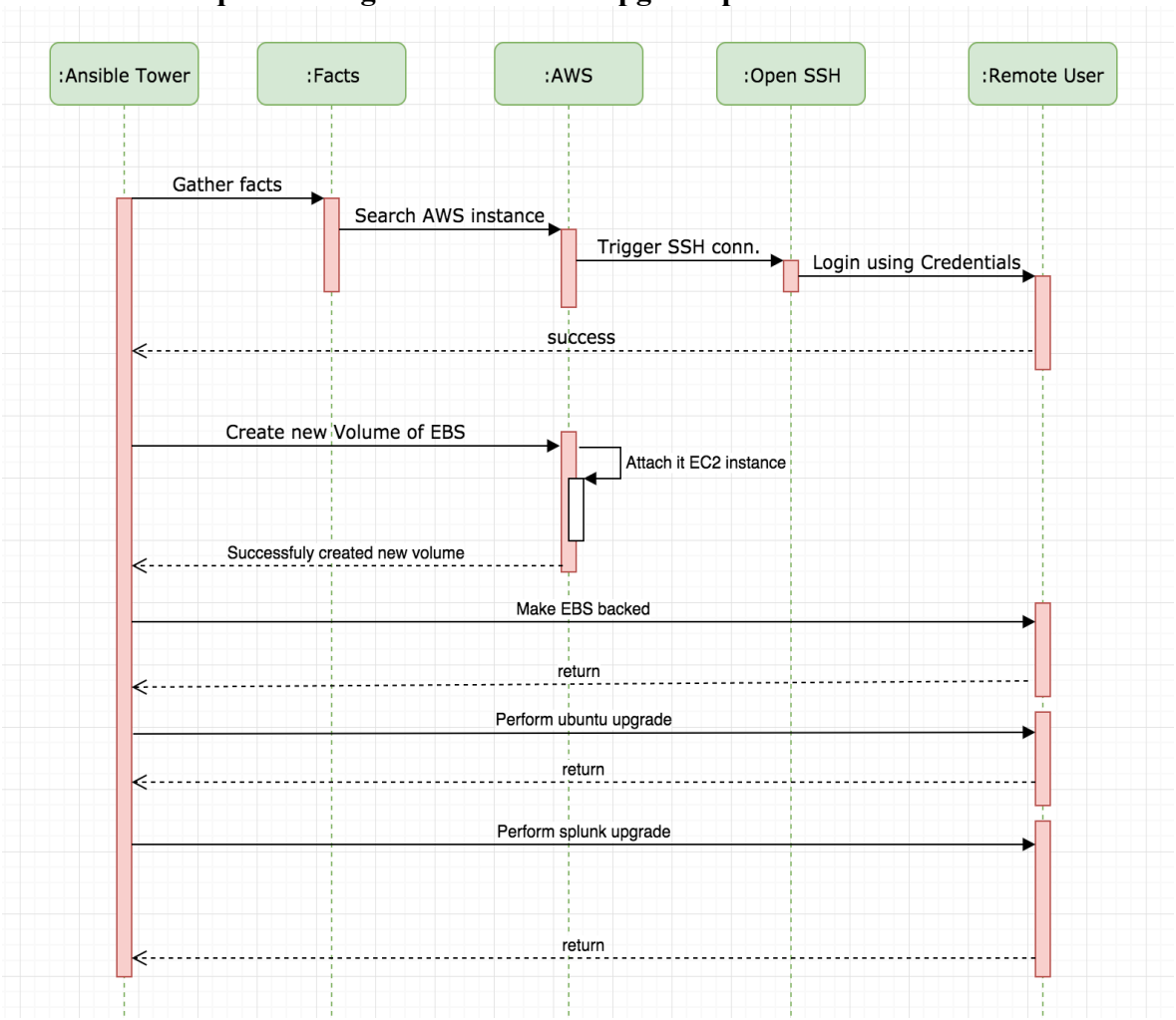


Fig 4.5 Sequence Diagram for Overall upgrade procedures

Brief Description:

The above sequence diagram shows overall view of the upgrade procedure Here is the flow sequentially:

1. Check for SSH connection
2. Make a new volume for EBS.
3. Mount that volume to instance and sync new volume to root volume.
4. Upgrade Linux version.
5. Upgrade Splunk version

4.4 Activity Diagram

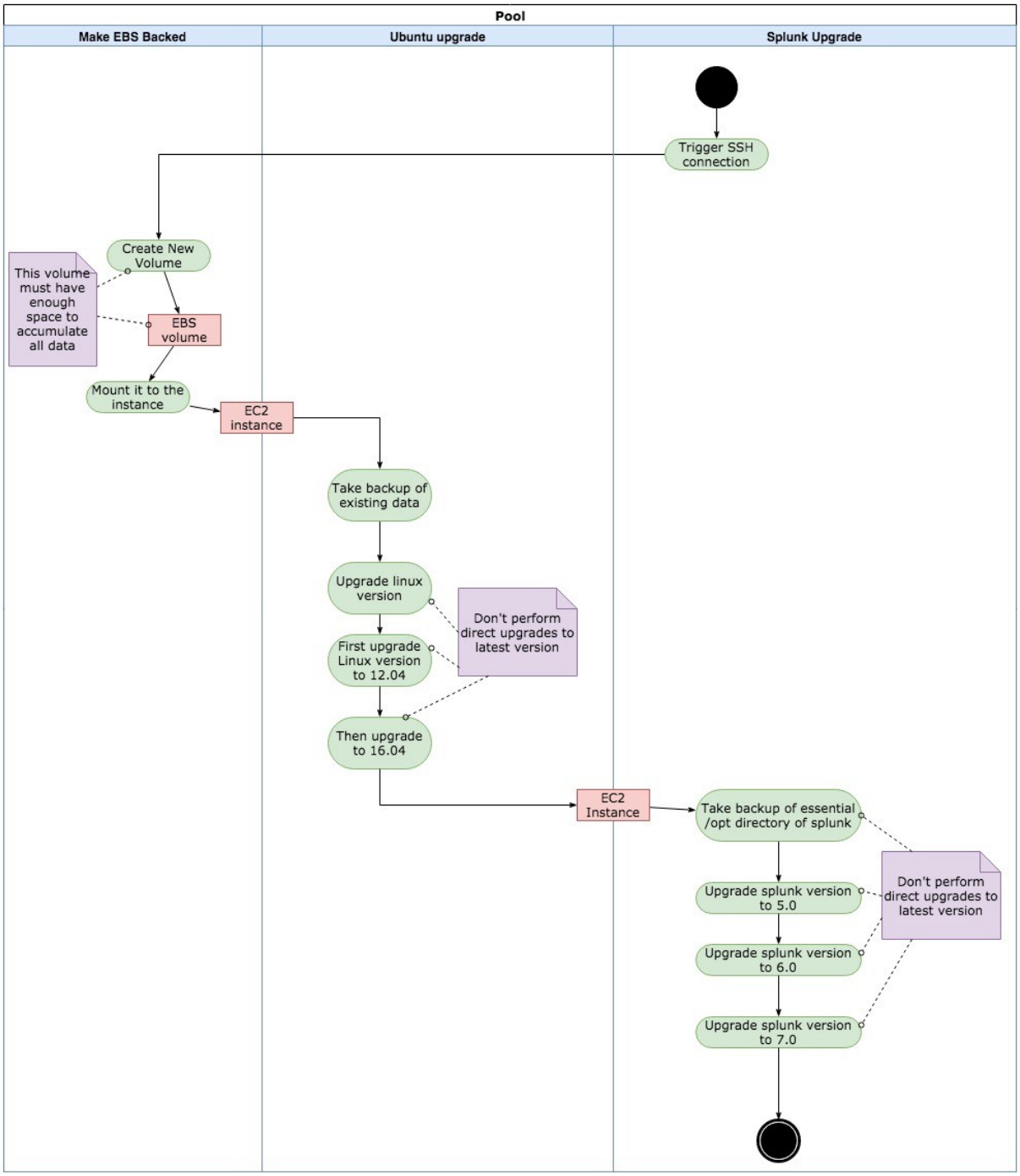


Fig 4.6 Activity Diagram for Whole application

5.0 IMPLEMENTATION

5.1 Ansible

Ansible is software that automates software provisioning, configuration management, and application deployment

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. An example is if you have a group of web servers behind a load balancer. Ansible can upgrade the web servers one at a time and while upgrading it can remove the current web server from the load balancer and disable it in your Nagios monitoring system.

Unlike Puppet or Chef, it doesn't use an agent on the remote host. Instead Ansible uses SSH which is assumed to be installed on all the systems you want to manage. Also, it's written in Python which needs to be installed on the remote host. This means that you don't have to setup a client server environment before using Ansible, you can just run it from any of your machines and from the client's point of view there is no knowledge of any Ansible server (you can run Puppet in standalone mode, but Puppet still needs to be installed). There are some other requirements though, for example if you want to do something related to git on a remote machine a git package must first be installed on the remote machine. The real strength of Ansible lies in its Playbooks. A playbook is like a recipe or an instructions manual which tells Ansible what to do when it connects to each machine. Playbooks are written in YAML, which simplistically could be viewed as XML but human readable. When I started using Ansible I'd never looked at YAML, but within hours I was able to write powerful playbooks.

The real strength of Ansible lies in its Playbooks. A playbook is like a recipe or an instructions manual which tells Ansible what to do when it connects to each machine. Playbooks are written in YAML, which simplistically could be viewed as XML but human readable. When I started using Ansible I'd never looked at YAML, but within hours I was able to write powerful playbooks. Also, there are a lot of examples online to help you while you learn.

You could have a Playbook which configures your servers according to a baseline you have defined, so they all are using the correct sshd config and central authentication. Then you use roles for specific server groups. Say you have some groups for web servers, database servers and monitoring servers. Then you decide to add a web server. When you fire off your Playbook, Ansible will install and configure the web server. It will make sure your database server allows connections from the new server, and then add the new server to your network monitoring solution so that you are informed if the server suffers a failure.

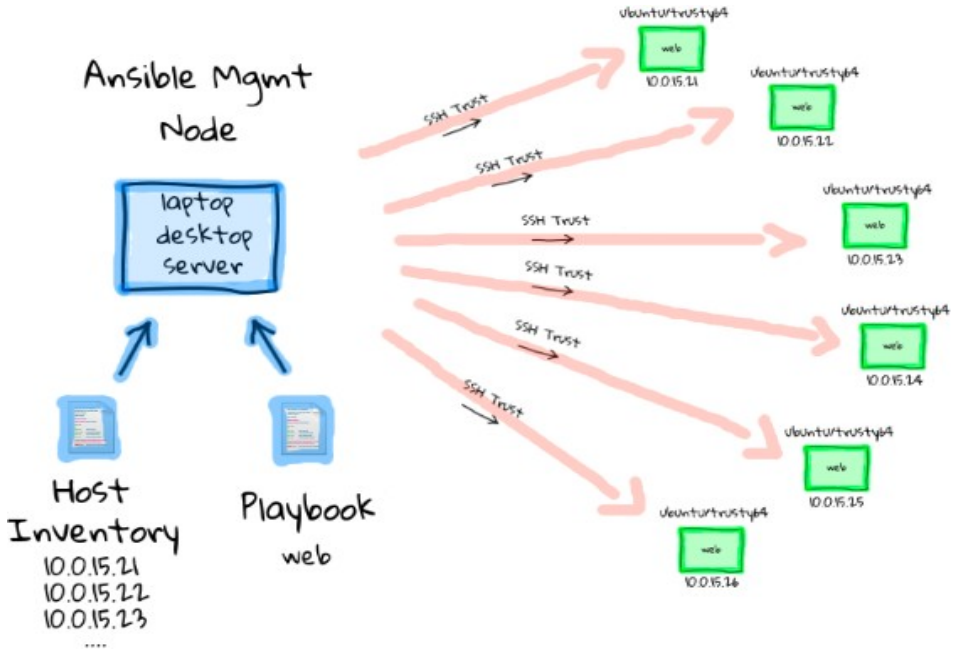
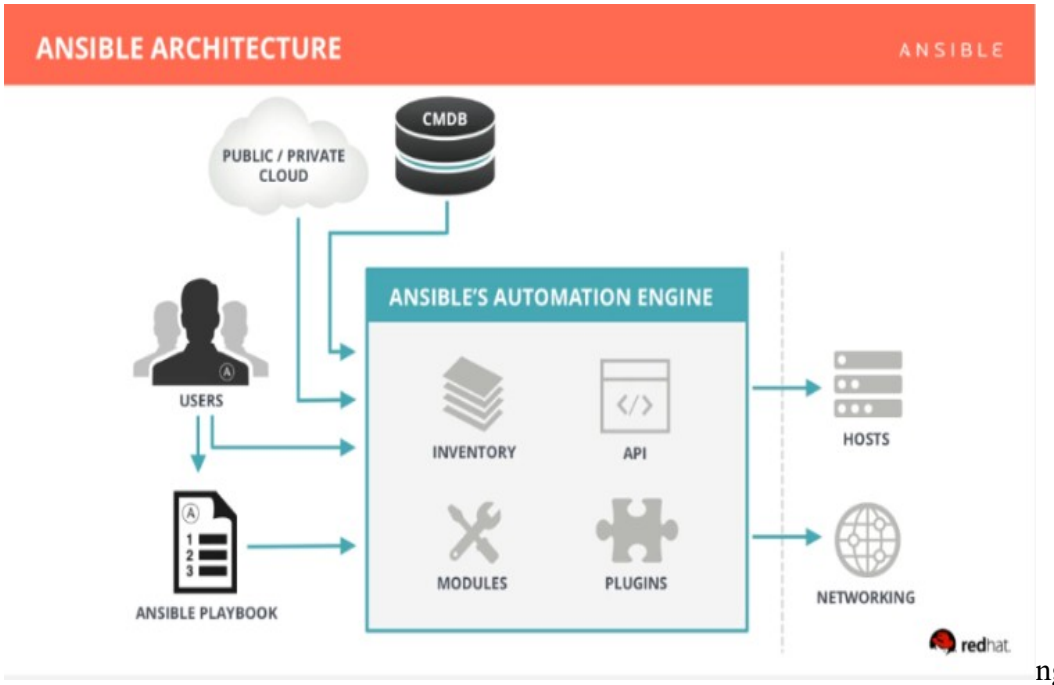


Fig. 5.1 Ansible Management

In DevOps, as we know development and operations work is integrated. This integration is very important for modern test-driven application design. Hence, Ansible integrates this by providing a stable environment to both development and operations resulting in smooth orchestration.

When developers begin to think of infrastructure as part of their application i.e as Infrastructure as code (IaC), stability and performance become normative. Infrastructure as Code is the process of managing and provisioning computing infrastructure (processes, bare-metal servers, virtual servers, etc.) and their configuration through machine-processable definition files.

In DevOps, Sysadmins work tightly with developers, development velocity is improved, and more time is spent doing activities like performance tuning, experimenting, and getting the



ngs done, and less time is spent fixing problems. Refer to the diagram below to understand how the tasks of sysadmins and other users are simplified by Ansible.

Fig. 5.2 Ansible Architecture

The Ansible Automation engine consists of:

- **Inventories:** Ansible inventories are lists of hosts (nodes) along with their IP addresses, servers, databases etc. which needs to be managed. Ansible then takes action via a transport – SSH for UNIX, Linux or Networking devices and WinRM for Windows system.
- **APIs:** APIs in Ansible are used as transport for Cloud services, public or private.
- **Modules:** Modules are executed directly on remote hosts through playbooks. The modules can control system resources, like services, packages, or files (anything really), or execute system commands. Modules do it by acting on system files, installing packages or making API calls to the service network. There are over 450 Ansible- provided modules that automate nearly every part of your environment. For e.g.

  Cloud Modules like *cloud formation* which creates or deletes an AWS cloud formation stack.

  Database modules like *mssql_db* which removes MYSQL databases from remote hosts.

Plugins: Plugins allows to execute Ansible tasks as a job build step. Plugins are pieces of code that augment Ansible's core functionality. Ansible ships with a number of handy plugins, and you can easily write your own. For example,

- *Cache* plugins are used to keep a cache of 'facts' to avoid costly fact- gathering operations.
- *Callback* plugins enable you to hook into Ansible events for display or logging purposes.

- There are a few more components in Ansible Architecture which are explained below:

Networking: Ansible can also be used to automate different networks. Ansible uses the same simple, powerful, and the agentless automation framework IT operations and development are already using. It uses a data model (a playbook or role) that is separate from the Ansible automation engine that easily spans different network hardware.

Hosts: The hosts in the Ansible architecture are just node systems which are getting automated by Ansible. It can be any kind of machine – Windows, Linux, RedHat etc.

Playbooks: Playbooks are simple files written in YAML format which describes the tasks to be executed by Ansible. Playbooks can declare configurations, but they can also orchestrate the steps of any manual ordered process, even if it contains jump statements. They can launch tasks synchronously or asynchronously.

CMDB : It is a repository that acts as a data warehouse for IT installations. It holds data relating to a collection of IT assets (commonly referred to as configuration items (CI)), as well as to describe relationships between such assets.

Cloud: It is a network of remote servers hosted on the Internet to store, manage, and process data, rather than a local server. You can launch your resources and instances on cloud and connect to your servers.

5.2 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 analyzes 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 behavior, machine behavior, 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 behavior 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.

5.2.1 Splunk Architecture

There are 3 main components (instances) in splunk - **Forwarder, Indexer, Search Head.**
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.

→ Forwarder :

There are two types of forwarder - **Heavy Forwarder** and **Universal Forwarder.**

1. **Universal Forwarder** just moves the data from one place to another.
2. **Heavy Forwarder** can move the data and also it parses the data (Parsing means converting the machine generated data into readable format) and it can filter the data.

→ 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 vmcenter will be moved to the indexer. Here the data will be indexed and then stored which then can be searched by the Search Head

→ Search Head:

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

- **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 indexer. 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 master:**
  Search master will manage the search heads.

- **Index master:**
  Index master 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.

5.2.2 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 a 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$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) Configuring 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.

5.3 AWS Cloud Platform

5.3.1 EC2 Instances

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers.

Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment. Amazon EC2 reduces the time required to obtain and boot new server instances to minutes, allowing you to quickly scale capacity, both up and down, as your computing requirements change. Amazon EC2 changes the economics of computing by allowing you to pay only for capacity that you actually use. Amazon EC2 provides developers the tools to build failure resilient applications and isolate them from common failure scenarios.

Benefits

ELASTIC WEB-SCALE COMPUTING

Amazon EC2 enables you to increase or decrease capacity within minutes, not hours or days. You can commission one, hundreds, or even thousands of server instances simultaneously. You can also use Amazon EC2 Auto Scaling to maintain availability of your EC2 fleet and automatically scale your fleet up and down depending on its needs in order to maximize performance and minimize cost. To scale multiple services, you can use AWS Auto Scaling.

COMPLETELY CONTROLLED

You have complete control of your instances including root access and the ability to interact with them as you would any machine. You can stop any instance while retaining the data on the boot partition, and then subsequently restart the same instance using web service APIs. Instances can be rebooted remotely using web service APIs, and you also have access to their console output.

FLEXIBLE CLOUD HOSTING SERVICES

You have the choice of multiple instance types, operating systems, and software packages. Amazon EC2 allows you to select a configuration of memory, CPU, instance storage, and the boot partition size that is optimal for your choice of operating system and application. For example, choice of operating systems includes numerous Linux distributions and Microsoft Windows Server.

INTEGRATED

Amazon EC2 is integrated with most AWS services such as Amazon Simple Storage Service (Amazon S3), Amazon Relational Database Service (Amazon RDS), and Amazon Virtual Private Cloud (Amazon VPC) to provide a complete, secure solution for computing, query processing, and cloud storage across a wide range of applications.

RELIABLE

Amazon EC2 offers a highly reliable environment where replacement instances can be rapidly and predictably commissioned. The service runs within Amazon's proven network infrastructure and data centers. The Amazon EC2 Service Level Agreement commitment is 99.99% availability for each Amazon EC2 Region.

SECURE

Cloud security at AWS is the highest priority. As an AWS customer, you will benefit from a data center and network architecture built to meet the requirements of the most security-sensitive organizations. Amazon EC2 works in conjunction with Amazon VPC to provide security and robust networking functionality for your compute resources.

5.3.2 EBS Optimized Instances

An Amazon EBS–optimized instance uses an optimized configuration stack and provides additional, dedicated capacity for Amazon EBS I/O. This optimization provides the best performance for your EBS volumes by minimizing contention between Amazon EBS I/O and other traffic from your instance.

EBS–optimized instances deliver dedicated bandwidth to Amazon EBS, with options between 425 Mbps and 14,000 Mbps, depending on the instance type you use. When attached to an EBS–optimized instance, General Purpose SSD (gp2) volumes are designed to deliver within 10% of their baseline and burst performance 99% of the time in a given year, and Provisioned IOPS SSD (io1) volumes are designed to deliver within 10% of their provisioned performance 99.9% of the time in a given year. Both Throughput Optimized HDD (st1) and Cold HDD (sc1) guarantee performance consistency of 90% of burst throughput 99% of the time in a given year. Non-compliant periods are approximately uniformly distributed, targeting 99% of expected total throughput each hour

The following table shows which instance types support EBS optimization, the dedicated bandwidth to Amazon EBS, the maximum number of IOPS the instance can support if you are using a 16 KiB I/O size, and the typical maximum aggregate throughput that can be achieved on that connection in MiB/s with a streaming read workload and 128 KiB I/O sizes. Choose an EBS–optimized instance that provides more dedicated Amazon EBS throughput than your application needs; otherwise, the connection between Amazon EBS and Amazon EC2 can become a performance bottleneck.

5.4 Ansible Playbooks

Ansible playbooks are a way to send commands to remote computers in a scripted way. Instead of using Ansible commands individually to remotely configure computers from the command line, you can configure entire complex environments by passing a script to one or more systems.

Ansible playbooks are written in the YAML data serialization format. If you don't know what a data serialization format is, think of it as a way to translate a programmatic data structure (lists, arrays, dictionaries etc) into a format that can be easily stored to disk. The file can then be used to recreate the structure at a later point. JSON is another popular data serialization format, but YAML is much easier to read.

Each playbook contains one or more plays, which map hosts to a certain function.
Ansible does this through something called tasks, which are basically module calls.

In our application, we have created six shell scripts that will run docker commands and output of that will be indexed as splunk event.

5.5 System Study

- Splunk is already installed on instances. Splunk is up and running already. In order to upgrade the splunk enterprise it needs to be stopped and downtime occurs.
- In distributed cloud deployment, there is no data loss at all as there will be always more than three indexers for a specific cloud deployment.

- Data is continuously ingested in splunk cloud infrastructure via heavy forwarders or lightweight forwarders

- One instance is upgraded at a time so if one indexer is down others can ingest data and hence whole data remain intact and if new data comes it doesn't get lost.
- Upgrade process of Splunk must be ordered one. In distributed deployment first Cluster master is upgraded. Then Search head is upgraded and at last indexer are upgraded.
- There are two types of upgrade procedure, rolling upgrade and non-rolling upgrade. These two procedures are dependent on splunk version and distributed architecture. Splunk version must be determined first before proceeding.
- There are certain dependencies between Splunk version and Ubuntu version.
- First Ubuntu upgrade is performed to avoid any type of conflict between future version of Splunk and its processing.
- EBS back instances play a crucial role in maintaining data backup. No data is lost during upgrading. Also, it is very easy to simply attach and detach EBS volumes

5.6 Implementation of Application

Using Ansible with AWS

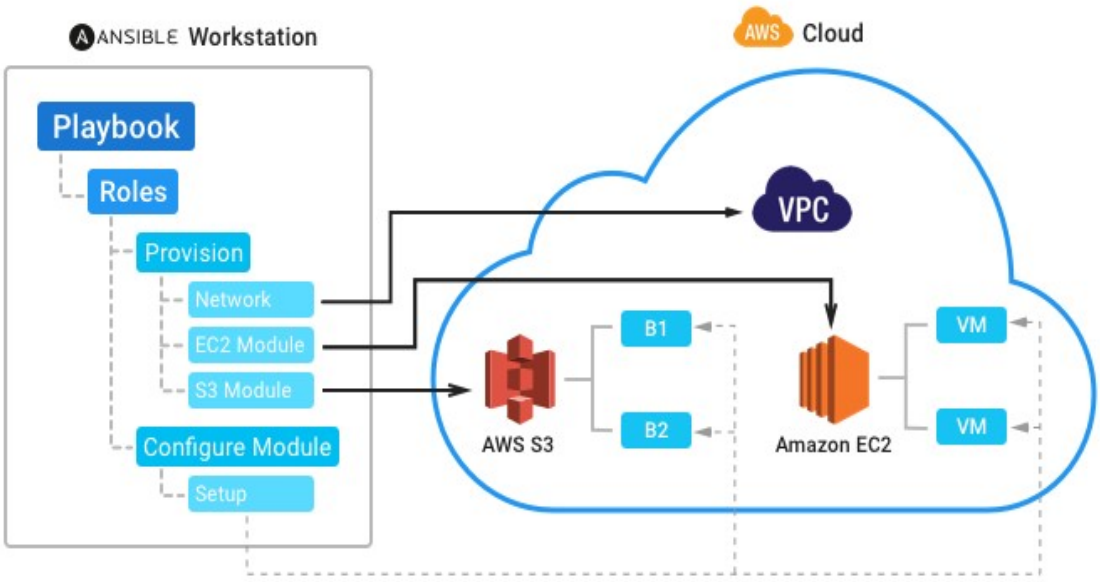


Fig. 5.3 Ansible with AWS

Dynamic inventory is used in this project. Benefit of dynamic inventory is that we can refresh cache of all instances whose access key and secret key is set in environment variable. To set AWS credentials use following stanzas

```
$ export AWS_ACCESS_KEY_ID='YOUR_AWS_API_KEY'
$ export AWS_SECRET_ACCESS_KEY='YOUR_AWS_API_SECRET_KEY'
```

To get started with dynamic inventory management, one need to grab the EC2.py script and the EC2.ini config file. The EC2.py script is written using the Boto EC2 library and will query AWS for your running Amazon EC2 instances. The EC2.ini file is the config file for EC2.py, and can be used to limit the scope of Ansible's reach. You can specify the regions, instance tags, or roles that the EC2.py script will find. Need to add own SSH key to bash agent or the agent which is used by ansible while playbook runs.

At their heart, inventory files are simply a mapping from some name to a destination address. The default ec2.ini settings are configured for running Ansible from outside EC2 (from your laptop for example) – and this is not the most efficient way to manage EC2.

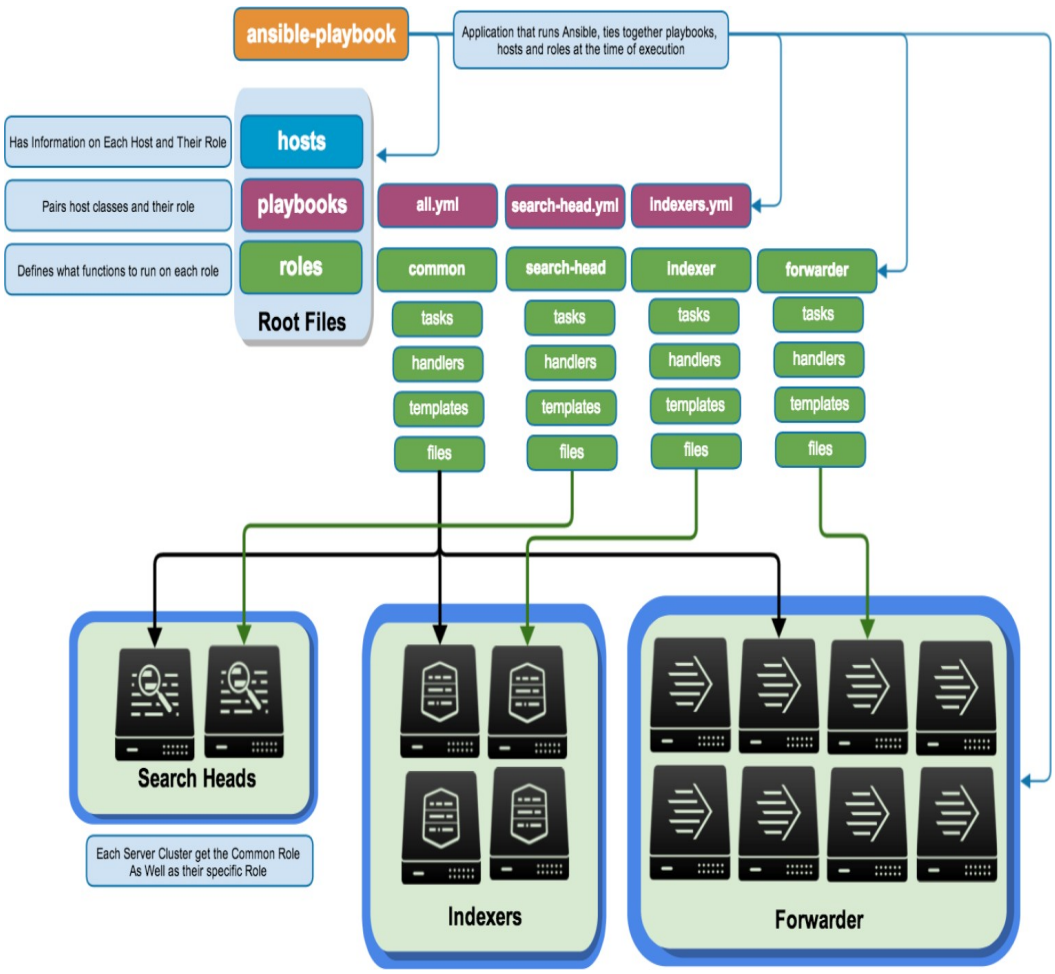


Fig 5.4 Directory Structure of Ansible scripts

Upgrade each tier separately

When upgrading tiers separately:
5.6.1 Upgrade the tiers in the prescribed order.

5.6.2 Within each tier, Upgrade all nodes as a single operation.
Functionality introduced in the new release will not be available until all tiers complete the upgrade.

Caution: Even when upgrading each tier separately, it is strongly recommended that you complete the entire upgrade process quickly, to avoid any possibility of incompatibilities between node types running different versions.

You must follow this order of upgrade when upgrading the tiers in discrete operations:

1. Upgrade the master node.
2. Upgrade the search head tier.
3. Upgrade the peer node tier.

1. Upgrade the master node

a. Stop the master.
    i. Upgrade the master, following the normal procedure for any Splunk Enterprise upgrade, as described in How to upgrade Splunk Enterprise in the *Installation Manual*.
b. Start the master, accepting all prompts, if it is not already running.
c. You can view the master dashboard to verify that all cluster nodes are up and running.

2. Upgrade the search head tier

The method that you use to upgrade the search head tier depends on whether or not the tier consists of a search head cluster:

5.6.1 If the search head tier consists of a search head cluster, follow the procedure in Upgrade a search head cluster. If desired, you can perform a rolling upgrade of the search head cluster, as described in that topic.

5.6.2 If the search head tier consists of independent search heads, follow this procedure:

1. Stop all the search heads.
2. Upgrade the search heads, following the normal procedure for any Splunk Enterprise upgrade, as described in How to upgrade Splunk Enterprise in the *Installation Manual*.
3. Start the search heads, if they are not already running.
   You can view the master dashboard to verify that all cluster nodes are up and running.

3. Upgrade the peer node tier

1. Run on the master.
   `splunk enable maintenance-mode`
   To confirm that the master is in maintenance mode `splunk show maintenance-mode` on the master.
2. This step prevents unnecessary bucket fix-ups. See Use maintenance mode.
3. Stop all the peer nodes.
   When bringing down the peers, use the `splunk stop` command, not `splunk offline`.
4. Upgrade the peer nodes, following the normal procedure for any Splunk Enterprise upgrade, as described in How to upgrade Splunk Enterprise in the *Installation Manual*.
5. Start the peer nodes, if they are not already running.

6. Run `splunk disable maintenance-mode` on the master.

To confirm that the master is not in maintenance mode, `splunk show maintenance-mode` on the master.

You can view the master dashboard to verify that all cluster nodes are up and running.

Upgrade from 5.x to 6.x or later

When you upgrade from a 5.x indexer cluster to a 6.x or later cluster, you must take all cluster nodes offline. You cannot perform a rolling, online upgrade.

Perform the following steps:

1. On the master, run the `safe_restart_cluster_master` script with the `--get_list` option:

```
splunk cmd python safe_restart_cluster_master.py
    <master_uri> --auth <username>:<password> --
                                        get_list
```

Note: `Master uri` parameter, use the URI and port number of the

For the master node. For example: `https://10.152.31.202:8089`

This command puts a list of all cluster bucket copies and their states into the file `$SPLUNK_HOME/var/run/splunk/cluster/buckets.xml`. This list is fed back to the master after the master upgrade.

To obtain a copy of this script, copy it from here: The safe_restart_cluster_master script.

For information on why this step is needed, see Why the safe_restart_cluster_master script is necessary.

1. Stop the master.

2. Stop all the peers and search heads.

When you bring down the peers, use the `splunk stop` command, not `splunk offline`.

3. Upgrade the master node, following the normal procedure for any Splunk Enterprise upgrade, as described in How to upgrade Splunk Enterprise in the *Installation Manual*. **Do not upgrade the peers yet.**

4. Start the master, accepting all prompts, if it is not already running.

5. Run the `splunk apply cluster-bundle` command, using the syntax described in Update cluster peer configurations and apps. (This step is necessary to avoid extra peer restarts, due to a 6.0 change in how the configuration bundle checksum is calculated.)

6. Run `splunk enable maintenance-mode` on the master. To confirm that the master is in maintenance mode, run `splunk show maintenance-mode`. This step prevents unnecessary bucket fix-ups. See Use maintenance mode.

7. Upgrade the peer nodes and search heads, following the normal procedure for any Splunk Enterprise upgrade, as described in How to upgrade Splunk Enterprise in the *Installation Manual*.

8. Start the peer nodes and search heads, if they are not already running.

9. On the master, run the `safe_restart_cluster_master` script again, this time with the `freeze_from` option, specifying the location of the bucket list created in step 1:

```
splunk cmd python safe_restart_cluster_master.py
<master_uri> --auth <username>:<password> --freeze_from
<path_to_buckets_xml>
```

For example:

```
splunk cmd python safe_restart_cluster_master.py
<master_uri> --auth admin:your_password --freeze_from
$SPLUNK_HOME/var/run/splunk/cluster/buckets.xml
```

This step feeds the master the list of frozen buckets obtained in step 1.

10. Run `splunk disable maintenance-mode` on the master. To confirm that the master is not in maintenance mode, run

`splunk show maintenance-mode`.

You can view the master dashboard to verify that all cluster nodes are up and running.

Upgrading search head cluster

To perform a member-by-member upgrade:

Upgrade one member and make it captain:

a. Stop the member.
b. Upgrade the member.
c. Start the member and wait while it joins the cluster.
d. Transfer captaincy to the upgraded member. See Transfer captaincy.
e. **Caution:** During this upgrade procedure, you must run the `splunk transfer shcluster-captain` command from the current captain. In addition, you cannot use the search head clustering dashboard in Settings to transfer captaincy.
    i.

For each additional member, one-by-one:

a. Stop the member.
b. Upgrade the member.
c. Start the member.
d. Upgrade the deployer:
e. Stop the deployer.
f. Upgrade the deployer.
g. Start the deployer.

Ubuntu Upgrade

You can always upgrade from LTS to LTS (Long Term Support):

5.6.3 upgrade from 14.04 to 16.04 and then

5.6.4 upgrade from 16.04 to 18.04
Following this approach there is no need for 16.04 medium

As always, consider creating a backup of critical files before start.

Limit to LTS

The way to limit upgrades to LTS without a GUI is to change the value of `Prompt` in `/etc/update-manager/release-upgrades`.

Set it to

```
[DEFAULT] Prompt=lts
```

The possible values are

5.6.5 never: Never check for a new release.

5.6.6 normal: Check to see if a new release is available. If more than one new release is found, the release upgrader will attempt to upgrade to the release that immediately succeeds the currently-running release.

5.6.7 lts: Check to see if a new LTS release is available. The upgrader will attempt to upgrade to the first LTS release available after the currently-running one. Note that this option should not be used if the currently-running release is not itself an LTS release, since in that case the upgrader won't be able to determine if a newer release is available

How to upgrade

For both upgrade processes, you should always update the current system via

```
sudo apt-get update
sudo apt-get upgrade
```

Then start the upgrade via

```
sudo do-release-upgrade
```

or - SPECIAL CASE - for development versions (which is valid for upgrades from 14.04 to 16.04 until 16.04.1 was released):

```
sudo do-release-upgrade -d
```

If do-release-upgradecommand is not found, install it:

```
sudo apt-get install update-manager-core
```

6.0 TESTING

6.1 TESTING PLAN

The testing technique that is going to be used in the project is black box testing. In black box testing the expected inputs to the system are applied and only the outputs are checked. The testing sub-process includes the following activities in a phase dependent manner: Create Test Plans.

a) Create test Specifications.
b) Review Test Plans and Test Specifications
c) Conduct tests according to the Test Specifications, and log the defects
d) Fix defects, if any.
e) When defects are fixed continue from activity.

6.2 TESTING STRATEGY

The development process repeats this testing sub-process a number of times for the following phases:

- Unit Testing
- Integration Testing

Unit Testing tests a unit of code (module or program) after coding of that unit is completed. Integration Testing tests whether the various programs that make up a system, interface with each other as desired, fit together and whether the interfaces between the programs are correct. System Testing ensures that the system meets its stated design specifications. Acceptance Testing is testing by the users to ascertain whether the system developed is a correct implementation of the Software Requirements Specification.

Testing is carried out in such a hierarchical manner to ensure that each component is correct and the assembly/combination of components is correct. Merely testing a whole system at the end would most likely throw up errors in components that would be very costly to trace and fix. We have performed both Unit Testing and System Testing to detect and fix errors.

6.3 TESTING METHODS

- **White Box Testing**

Also known as glass box, structural, clear box and open box testing. A software testing technique whereby explicit knowledge of the internal workings of the item being tested are

used to select the test data. Unlike black box testing, white box testing uses specific knowledge of programming code to examine outputs. The test is accurate only if the tester knows what the program is supposed to do; it means that it must be completely aware that for particular input a particular output must be obtained. The main benefit of this type of testing is Tester can see if the program diverges from its intended goal. This test concentrates on the examination of the code rather than the specification. We have included three different forms of white box testing.

- **Statement Coverage Criteria:**
  This is the simplest coverage criteria. We are checking in it that each statement of the program was executed "at least once".

- **Branch Coverage Criteria:**
  An improvement over statement is Branch Coverage. In which we are running a series of test to ensure that all branches must tested at least once.

- **Path Coverage Criterion:**

  There was many errors which are not detected by statement or branch testing. The reason is that some errors are related to some combination of branches and it may be not check in other test. In this test, we are checking that all path of programs are executed or not.

➢ **Black Box Testing**

Black Box and White Box are test design methods. Black Box test design treats the system as a black box, so it doesn't explicitly use knowledge of the internal structure. Black Box test design is usually described as focusing on testing functional requirements. Also known as behavioral, functional, opaque-box and closed-box. Its useful to find error such as:

- Interface error
- Incorrect or missing function
- Performance error

➢ **Unit Testing**

Unit testing is a method of testing the correctness of a particular module of source code. The idea is to write test cases for every non-trivial function or method in the module so that each test case is separate from the others if possible. The developers mostly do this type of testing. In this method of testing we test all individual components to ensure that they operate correctly. Each component is tested independently without other system components.

➢ **Integration Testing**

It is the phase of software testing in which individual software modules we are combined and tested as a group. It follows unit testing and precedes system testing. The purpose of Integration testing is to verify functional, performance and reliability requirements placed on major design items. It takes as its input modules that have been checked out by unit testing, groups them in larger aggregates, applies tests defined in an Integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

- **Regression Testing**

  After we made some changes in one module, we had to check whether older modules were working perfectly or not.

6.4 Testing Plan

The testing technique that is going to be used in the project is white box testing, that is expected inputs to the system are applied and only the outputs are checked.

6.5 Testing cases

In all the cases we have checked script is properly working or not in terminal. If in any error occurs it will be displayed in terminal. If some tasks are not required then they are skipped or if any error is ignorable then it is ignored as it doesn't cause hurdle to subsequent tasks .

Table 6.5.1 Pre-checks for existing cluster

Test Case ID	Test Steps	Test Data	Expected Result	Actual Result	Pass /Fail
T1	1)Open AWS Console 2.) Determine ID of the instance 3.)Pass the same ID in the play without any changes 4.)SSH to instance through play and determine splunk version and Ubuntu version 5.)Make sure for checking ubuntu version login as root 6.)For splunk version login as splunk user		Splunk version and ubuntu version	As Expected	Pass

Test Case ID	Test Steps	Test Data	Expected Result	Actual Result	Pass/Fail
T2	1.) Open AWS console 2.) Determine ID of the instance3.) Check root volume and EBS		EBS volume	As Expected	Pass

Table 6.5.2 Backing up existing system

Test Case ID	Test Steps	Test Data	Expected Result	Actual Result	Pass/Fail
T3	1.) Determine directories whose backup is needed 2.) Exclude directories whose data is not needed or have very much big size 3.)Make tar ball of all directories 4.) Upload it to S3		Compressed Form of data(.tar)	As Expected	Pass

Test Case ID	Test Steps	Test Data	Expected Result	Actual Result	Pass/Fail
T4	1.) Determine directories whose backup is needed 2.) Take backup of /opt/splunk/etc It contains all app data 3.)Make tarball /opt directory 4.) Upload it to S3		Compressed Form of data(.tar)	As Expected	Pass

Table 6.5.3 Determine instance type and upgrade accordingly

Test Case ID	Test Steps	Test Data	Expected Result	Actual Result	Pass/Fail
T5	1.) Determine Cluster-master of clustering 2.) Untar latest package 3.) Check for data loss		Upgrade to latest version	As Expected	Pass

Test Case ID	Test Steps	Test Data	Expected Result	Actual Result	Pass/Fail
T6	1.) Determine Search head of clustering 2.) Untar latest package 3.) Check for data loss		Upgrade to latest version	As Expected	Pass

Test Case ID	Test Steps	Test Data	Expected Result	Actual Result	Pass/Fail
T7	1.) Determine Indexers of clustering 2.) Untar latest package 3.) Check for data loss		Upgrade to latest version	As Expected	Pass

Table 6.5.4 Perform splunk upgrade incrementally

Test Case ID	Test Steps	Test Data	Expected Result	Actual Result	Pass/Fail
T8	1.) Determine Search Head Captain 2.) Upgrade Search haed captain 3.) Perform Rolling Upgrade		All nodes Upgraded simultaneously	As Expected	Pass

Test Case ID	Test Steps	Test Data	Expected Result	Actual Result	Pass/Fail
T9	1.) Determine Search Head Captain 2.) Upgrade Search haed captain 3.) Perform non-Rolling Upgrade		All nodes Upgraded One by one	As Expected	Pass

Table 6.5.5 Perform Ubuntu upgrade incrementally

Test Case ID	Test Steps	Test Data	Expected Result	Actual Result	Pass/Fail
T10	1.) Determine Ubuntu version 2.) Take backup of required directories and put in /tmp directory 3.) Fire do-release command		Ubuntu upgraded to version 18.04	As Expected	Pass

Test Case ID	Test Steps	Test Data	Expected Result	Actual Result	Pass/Fail
T11	1.) Determine Ubuntu version 2.) Take backup of required directories and put in /tmp directory 3.) Fire do-release command		Ubuntu upgraded to version 18.04	As Expected	Pass

Table 6.5.6 Restoring backup

Test Case ID	Test Steps	Test Data	Expected Result	Actual Result	Pass/Fail
T12.1	1.) Open AWS panel 2.) Open S3 terminal 3.) Determine required backup of specific date and time 4.) Download OS level backup		Backup of OS binaries	As Expected	Pass
T12.2	1.) Open AWS panel 2.) Open S3 terminal 3.) Determine required backup of specific date and time 4.) Download splunk backup		Backup of splunk binaries	As Expected	Pass

Table 6.5.7 Create EBS volume

Test Case ID	Test Steps	Test Data	Expected Result	Actual Result	Pass/Fail
T13	1.) Open AWS Panel 2.) Create New EBS volume 3.) Specify required AMI 4.) Specify volume type 5.) Specify size – Make sure the size must be enough to accommodate all splunk data 6,) Keep volume as non-encrypted as no encryption is required		New EBS volume	As Expected	Pass

Table 6.5.8 Attaching EBS volume

Test Case ID	Test Steps	Test Data	Expected Result	Actual Result	Pass/Fail
T14	1.) Take instance ID to which volume is to be attached. 2.) Ensure Instance is stopped and disable termination protection		Instance stopped	As Expected	Pass

Test Case ID	Test Steps	Test Data	Expected Result	Actual Result	Pass/Fail
T15	1.) Take instance ID to which volume is to be attached. 2.) Create new Mount point		Creating new Mount point	As Expected	Pass

Test Case ID	Test Steps	Test Data	Expected Result	Actual Result	Pass/Fail
T16	1.) Run fsync utility On instance 2.) Take backup of data to EBSvolume		Data synchronized Between root volume and EBS volume	As Expected	Pass

7.0 User Manual

The below figures shows the version of Splunk and Ubuntu with older version and then their upgraded version.

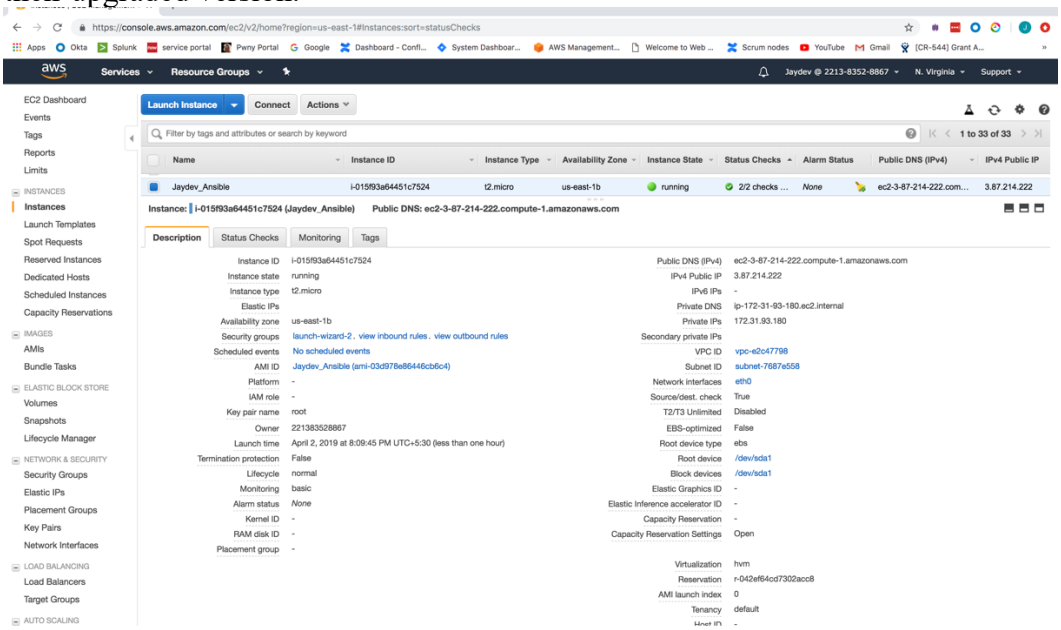


Fig 7.1 AWS instance for running play

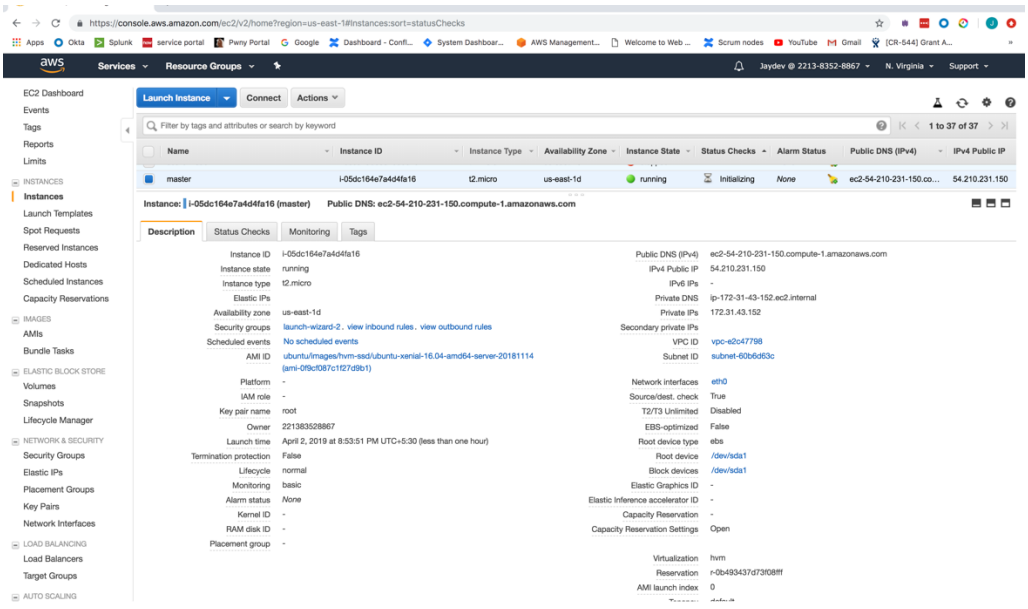


Fig 7.2 Cluster-master on AWS

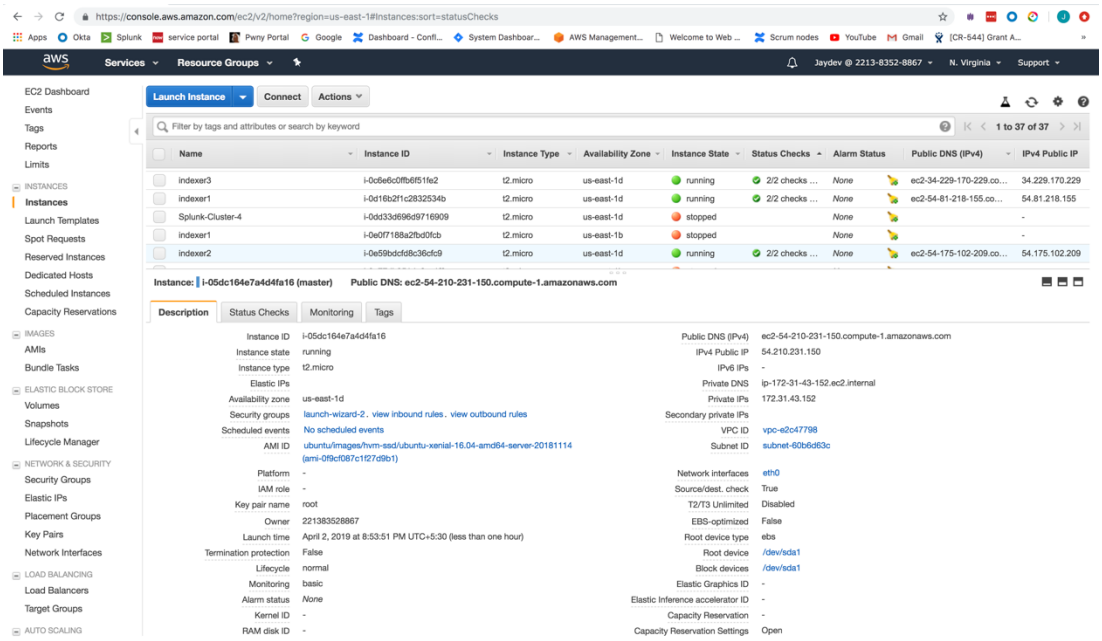


Fig 7.3 Indexers on AWS

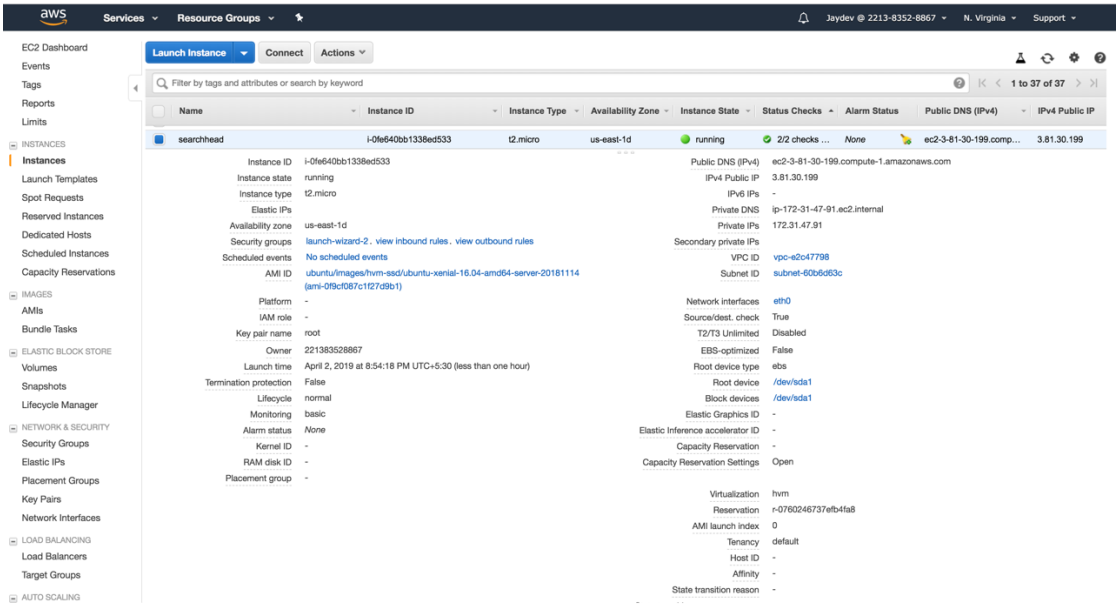


Fig 7.4 Search-head on AWS

After all the above instances are shown up in AWS , it shows that clustering has been done and splunk has been installed with v6.x.

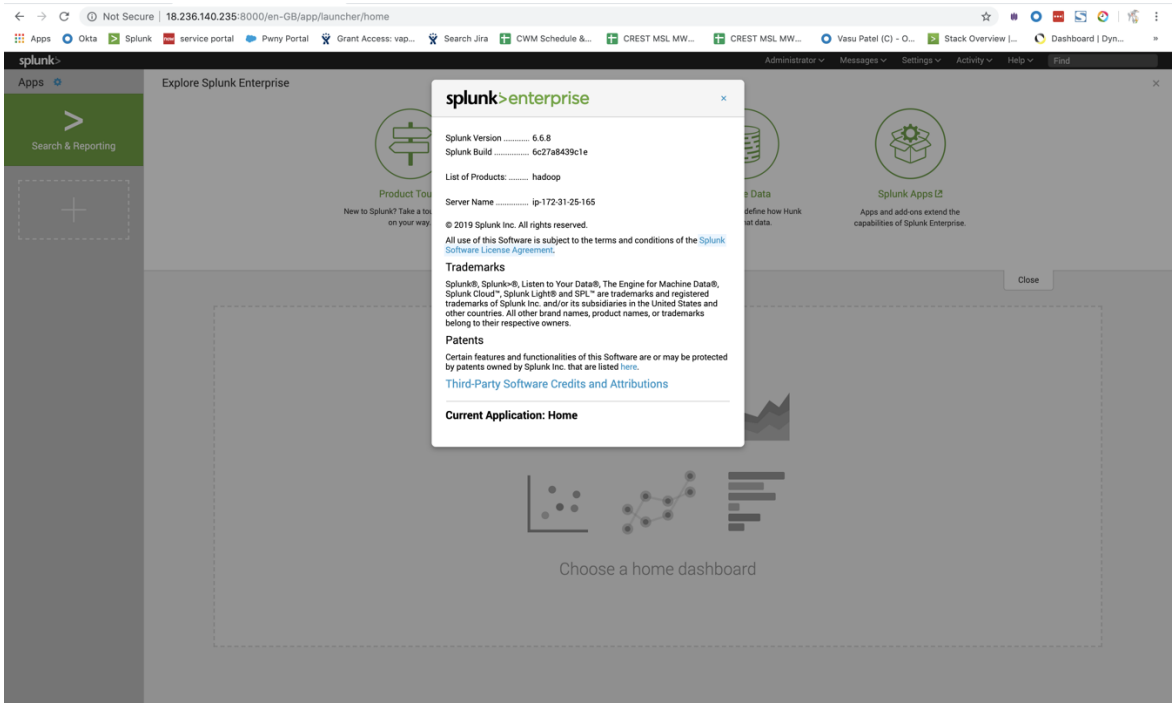


Fig 7.4 Cluster-master after clustering(with v6.6.8)

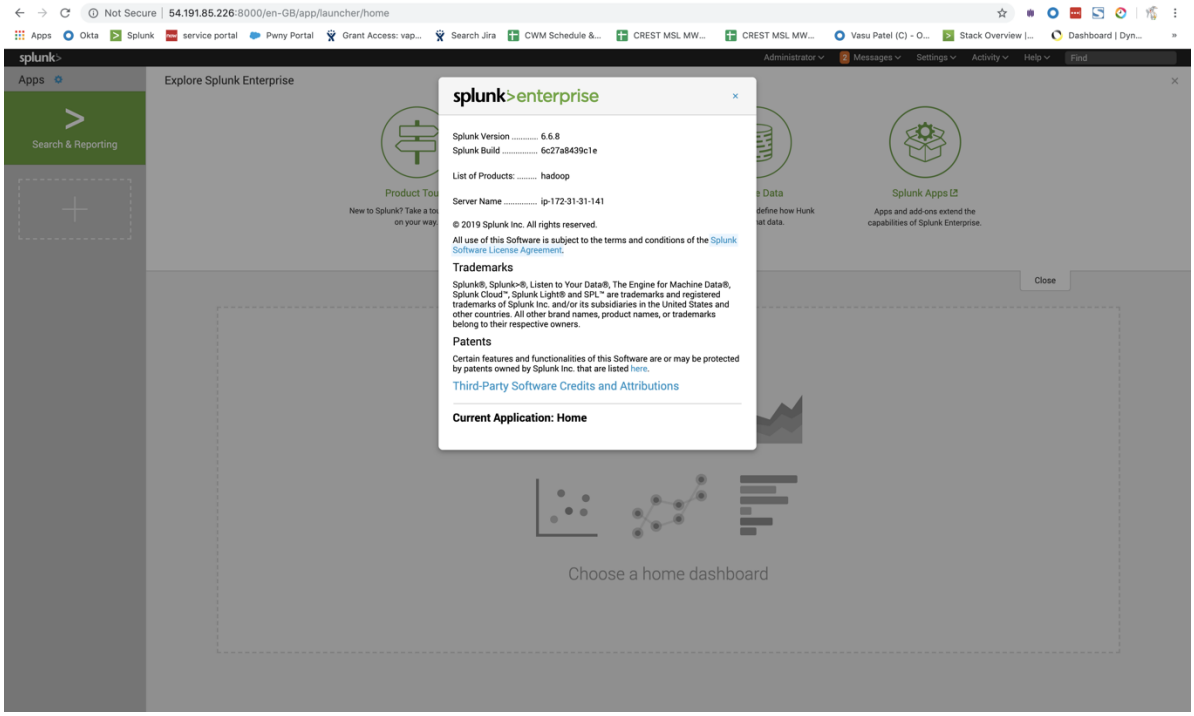


Fig 7.5 Search-head after clustering(with v6.6.8)

```
root@ip-172-31-28-31:/# cd opt/splunk/bin/
root@ip-172-31-28-31:/opt/splunk/bin# ./splunk --version
Splunk 6.6.8 (build 6c27a8439c1e)
root@ip-172-31-28-31:/opt/splunk/bin# hostname
ip-172-31-28-31
root@ip-172-31-28-31:/opt/splunk/bin#
```

Indexer 1

```
root@ip-172-31-29-145:/# cd opt/splunk/bin/
root@ip-172-31-29-145:/opt/splunk/bin# ./splunk --version
Splunk 6.6.8 (build 6c27a8439c1e)
root@ip-172-31-29-145:/opt/splunk/bin# hostname
ip-172-31-29-145
root@ip-172-31-29-145:/opt/splunk/bin#
```

Indexer 2

```
root@ip-172-31-16-159:/# cd opt/splunk/bin/
root@ip-172-31-16-159:/opt/splunk/bin# ./splunk version
Splunk 6.6.8 (build 6c27a8439c1e)
root@ip-172-31-16-159:/opt/splunk/bin# hostname
ip-172-31-16-159
root@ip-172-31-16-159:/opt/splunk/bin#
```

Indexer 3

Fig 7.6 Indexers after clustering(with v6.6.8)

- **Now we will run the script of Upgrading the Splunk Cluster to version 7.2.3**

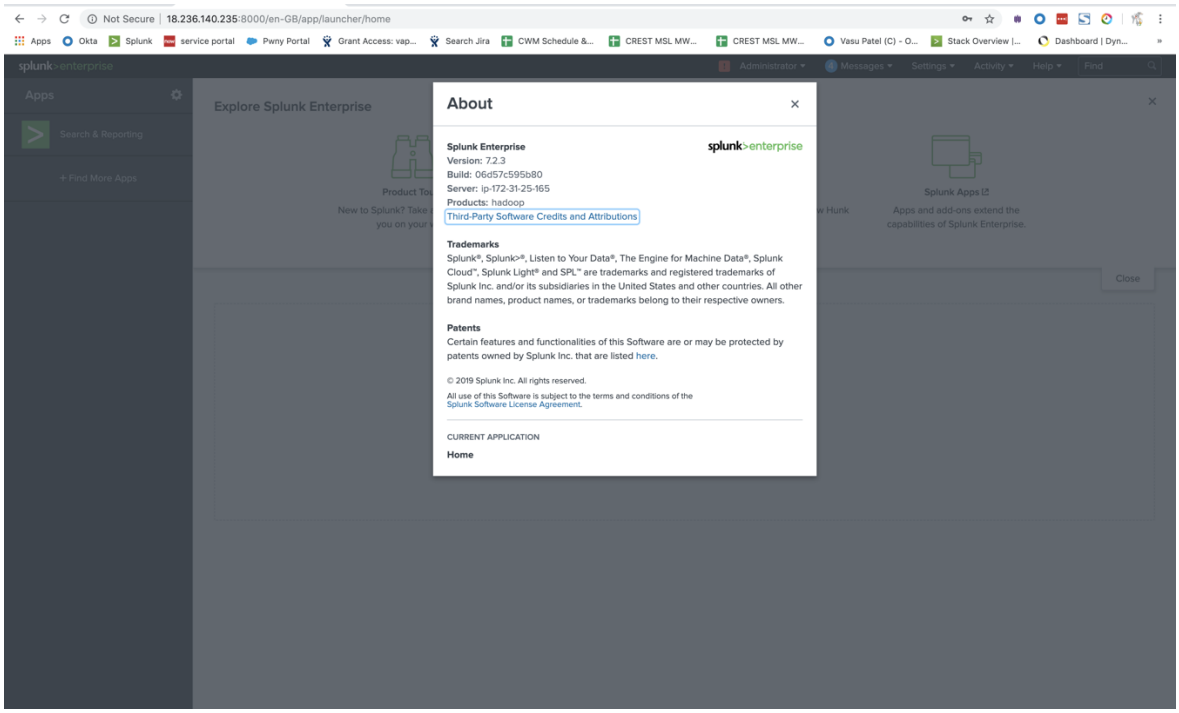


Fig 7.7 Cluster-master Upgraded (with v7.2.3)

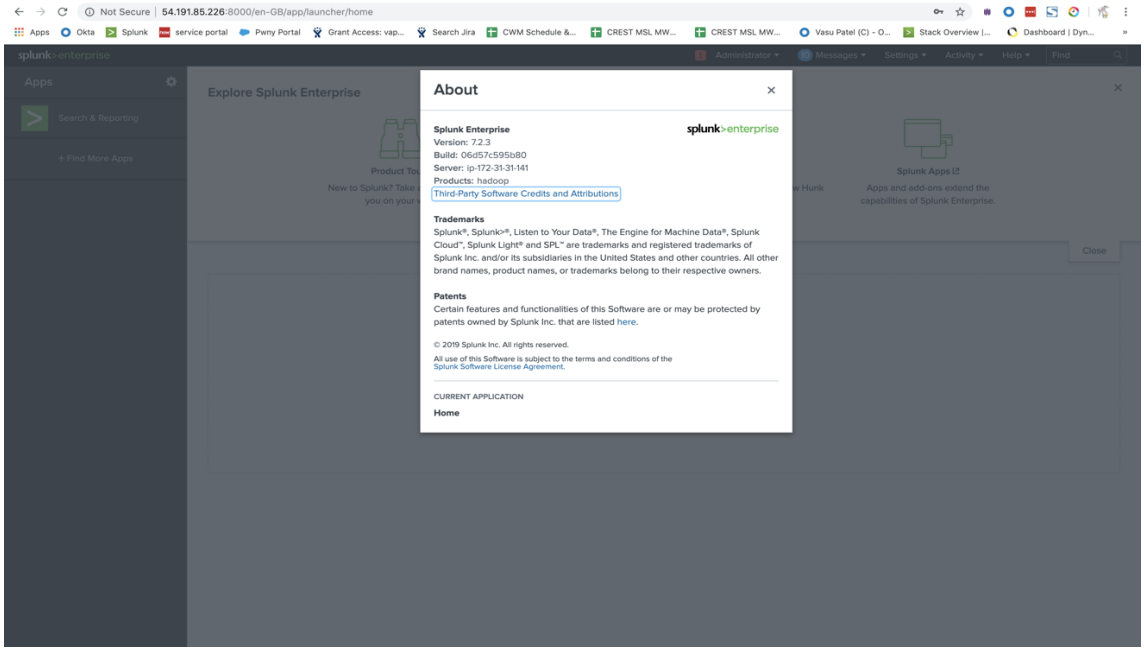


Fig 7.8 Search-head Upgraded (with v7.2.3)

```
root@ip-172-31-16-159:/opt/splunk/bin# ./splunk version
Splunk 6.6.8 (build 6c27a8439c1e)
root@ip-172-31-16-159:/opt/splunk/bin# hostname
ip-172-31-16-159
root@ip-172-31-16-159:/opt/splunk/bin# ./splunk version
Splunk 7.2.3 (build 06d57c595b80)
root@ip-172-31-16-159:/opt/splunk/bin#
```

Indexer 1

```
root@ip-172-31-28-31:/# cd opt/splunk/bin/
root@ip-172-31-28-31:/opt/splunk/bin# ./splunk --version
Splunk 6.6.8 (build 6c27a8439c1e)
root@ip-172-31-28-31:/opt/splunk/bin# hostname
ip-172-31-28-31
root@ip-172-31-28-31:/opt/splunk/bin# ./splunk --version
Splunk 7.2.3 (build 06d57c595b80)
root@ip-172-31-28-31:/opt/splunk/bin#
```

Indexer 2

```
root@ip-172-31-29-145:/opt/splunk/bin# ./splunk --version
Splunk 6.6.8 (build 6c27a8439c1e)
root@ip-172-31-29-145:/opt/splunk/bin# hostname
ip-172-31-29-145
root@ip-172-31-29-145:/opt/splunk/bin# ./splunk --version
Splunk 7.2.3 (build 06d57c595b80)
root@ip-172-31-29-145:/opt/splunk/bin#
```

Indexer 3

Fig 7.9 Indexers Upgraded (with v7.2.3)

- **After Splunk has been upgraded successfully, we will upgrade the Ubuntu from v16.04 to 18.04**

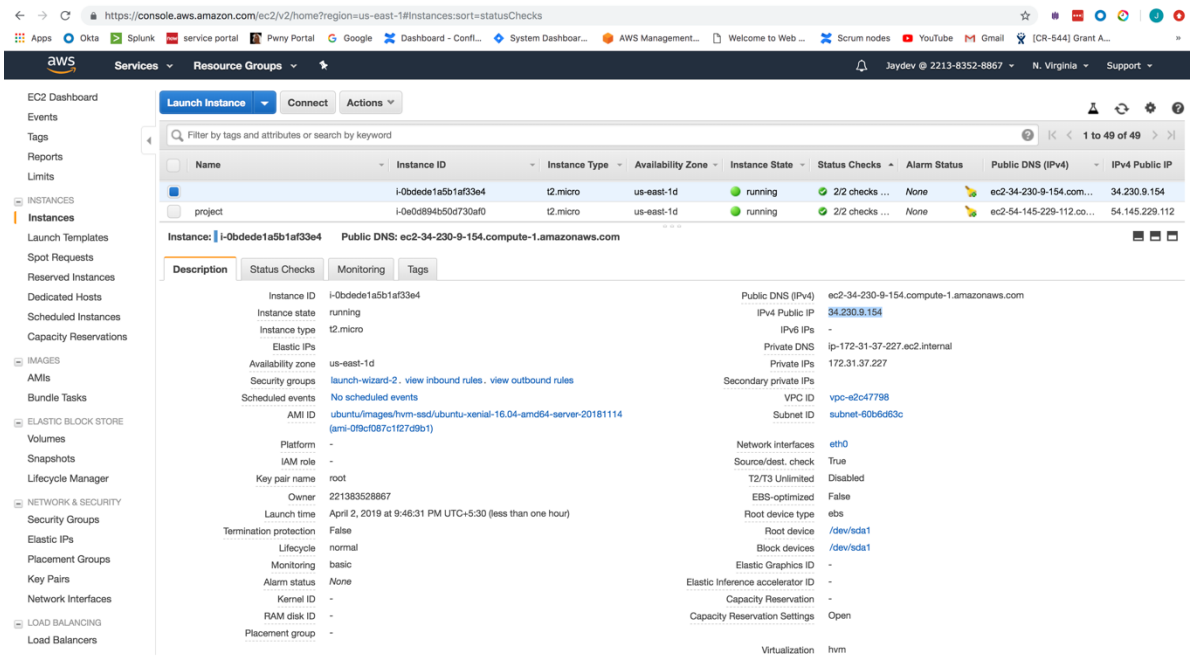


Fig 7.5 AWS instances with Ubuntu OS

```
exit
ubuntu@ip-172-31-93-180:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 16.04.5 LTS
Release:        16.04
Codename:       xenial
ubuntu@ip-172-31-93-180:~$
```

Fig 7.6 AWS instance with Ubuntu OS v16.04

After the Ubuntu Upgradation play run successfully following output will be shown:

```
*** System restart required ***
Last login: Tue Apr  2 16:21:10 2019 from 3.87.214.222
ubuntu@ip-172-31-34-109:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 18.04.2 LTS
Release:        18.04
Codename:       bionic
ubuntu@ip-172-31-34-109:~$
```

Fig 7.7 AWS instance with Ubuntu OS v18.04

8.0 CONCLUSION AND FUTURE EXTENSION

8.1 CONCLUSION

The proposed system will work efficiently for Perform Upgrades on Existing Splunk Cloud Infrastructure. This app is focused for DevOps users .The goal for upgradation was to deliver the latest technology with good processing power and maintain splunk up to date.

User can have better experience in managing docker containers by this app. The system offers various advantages to DevOps.This app was created with the thought to provide better and user-friendly way for monitoring performance parameters of docker.

The software provides the flexibility to IT admin for monitoring docker, its memory usage, performance parameters etc in a easy and an accurate way.

The system provides more precise, accurate and instant monitoring of the docker instances which can help the IT admin to monitor performance parameters in a better way.

8.2 FUTURE EXTENSION

8.2.1 These scripts are useful for upgrading Ubuntu on any cloud machine, it just need extension

8.2.2 EBS volumes will attach to any eligible EC2 instances.

8.2.3 These scripts currently use shell script which is supported by Linux machines only so in future I would like to expand this app for another OS also.

BIBLIOGRAPHY

→ Splunk Documentation:

https://docs.splunk.com/Documentation

→ Ansible Documentation:

http://docs.ansible.com/ansible/latest/index.html

→ AWS Documentation:

https://aws.amazon.com/documentation/

→ Ansible with AWS

https://www.ansible.com/integrations/cloud/amazon-web-services

→ Splunk troubleshooting:

https://answers.splunk.com/