MAJOR-2 PROJECT

SYNOPSIS REPORT on

“Real-Time Log Visualization using DevOps”

Submitted by:

Name	**Enrollment No.**	**Branch**
Manik Khurana	R110216092	B.Tech CSE CCVT
Babanjot Singh	R110216052	B.Tech CSE CCVT
Ekanshu Dargan	R110216063	B.Tech CSE CCVT
Chhavi Sharma	R110216055	B.Tech CSE CCVT

Under the guidance of:

Mr. Harvinder Singh

Assistant Professor
Department of the Virtualization,
School of Computer Science,
University Of Petroleum and Energy Studies.
Dehradun- 248007

UPES

Approved By

(Mr. Harvinder Singh) (Dr. Deepshikha Bhargava)

Project Guide **Department Head**

School of Computer Science
University of Petroleum & Energy Studies, Dehradun
Synopsis Report (2019-20)

1. Project Title

Real-time Log Visualization using DevOps.

2. Abstract

In this rapidly developing world, an ample amount of professionalism is required in every work environment. Therefore, software developers do not have enough time for everything that they are supposed to do. Earlier a simple task used to take weeks to be completed and now it happens in mere seconds. With this project we aim to present a software solution with the assistance of the latest tools used in agile environments like DevOps and reduce the amount of time and efforts to view the logs of any particular website. Crucial details of websites will be taken care of by the ELK (Elasticsearch, Logstash, and Kibana) collectively. The ELK stack will be up and running with the help of an ansible-playbook. This project will be done using only an ansible-playbook. This task, if not done this way requires a lot of work to be done each and every time. We will be using Docker and Docker Compose for containerization. This makes this software solution generic and portable. A curator will be set-up for indices cleaning.

Keywords: Ansible , Docker, Compose, FileBeats, Elasticsearch, Logstash, Kibana, Git

3. Introduction

This project serves as an application - a future solution - to a time-costly problem of getting logs manually generated with the help of the outdated applications. Using our solution the developers will be able to save a lot of time and they can then focus on more important modules and grow at a greater rate. The solution is an ansible playbook that performs the functions of setting up docker and docker-compose on the remote host with ELK stash tools through which we will perform all the tasks.There are various Devops tools through which we will create an interface from taking the logs and processing it , applying sorting on it , getting specific or desired outputs as per our requirements. Following are the tools that we will use

- **Ansible** — 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.

  Ansible's main goals are simplicity and ease-of-use. It also has a strong focus on security and reliability, featuring a minimum of moving parts, usage of OpenSSH for transport (with other transports and pull modes as alternatives), and a language that is designed around auditability by humans–even those not familiar with the program.

- **Docker** — Docker provides the ability to package and run an application in a loosely isolated environment called a container. The isolation and security allow you to run many containers simultaneously on a given host. Containers are lightweight because they don't need the extra load of a hypervisor, but run directly within the host machine's kernel. This means you can run more containers on a given hardware combination than if you were using virtual machines. You can even run Docker containers within host machines that are actually virtual machines!

  Docker provides tools and a platform to manage the lifecycle of your containers,Develop your application and its supporting components using containers. The container becomes the unit for distributing and testing your application.

  When you're ready, deploy your application into your production environment, as a container or an orchestrated service. This works the same whether your production environment is a local data center, a cloud provider, or a hybrid of the two.

- **Filebeat** — Filebeat is a lightweight shipper for forwarding and centralizing log data. Installed as an agent on your servers, Filebeat monitors the log files or locations that you specify, collects log events, and forwards them to either Elasticsearch or Logstash for indexing.

Here’s how Filebeat works: When you start Filebeat, it starts one or more inputs that look in the locations you’ve specified for log data. For each log that Filebeat locates, Filebeat starts a harvester. Each harvester reads a single log for new content and sends the new log data to libbeat, which aggregates the events and sends the aggregated data to the output that you’ve configured for Filebeat.

- **Logstash** — Logstash is an open source data collection engine with real-time pipelining capabilities. Logstash can dynamically unify data from disparate sources and normalize the data into destinations of your choice. Cleanse and democratize all your data for diverse advanced downstream analytics and visualization use cases.

  While Logstash originally drove innovation in log collection, its capabilities extend well beyond that use case. Any type of event can be enriched and transformed with a broad array of input, filter, and output plugins, with many native codecs further simplifying the ingestion process. Logstash accelerates your insights by harnessing a greater volume and variety of data.

- **Elasticsearch** — Elasticsearch is the distributed search and analytics engine at the heart of the Elastic Stack. Logstash and Beats facilitate collecting, aggregating, and enriching your data and storing it in Elasticsearch. Kibana enables you to interactively explore, visualize, and share insights into your data and manage and monitor the stack. Elasticsearch is where the indexing, search, and analysis magic happens.

  Elasticsearch provides real-time search and analytics for all types of data. Whether you have structured or unstructured text, numerical data, or geospatial data, Elasticsearch can efficiently store and index it in a way that supports fast searches. You can go far beyond simple data retrieval and aggregate information to discover trends and patterns in your data. And as your data and query volume grows, the distributed nature of Elasticsearch enables your deployment to grow seamlessly right along with it.

- **Kibana** — Kibana is an open source analytics and visualization platform designed to work with Elasticsearch. You use Kibana to search, view, and interact with data stored in Elasticsearch indices. You can easily perform advanced data analysis and visualize your data in a variety of charts, tables, and maps.

  Kibana makes it easy to understand large volumes of data. It's simple, browser-based interface enables you to quickly create and share dynamic dashboards that display changes to Elasticsearch queries in real time.

4. Problem Statement

Through log analytics we can collect and analyze the data that is generated by resources. As the log data grows the operations team starts facing issues to consolidate and manage the large data of log files. The issue with log data is that the logs are unstructured that are generated from various layers produce different types of logs. The challenge lies in consolidating the log data, process it to generate the Business and technical insights.

5. Objective

The primary objective of the project is to analyze and visualize log data in real-time. ELK helps achieve this goal. It will provide a centralized logging that will be useful when attempting to identify problems with the servers or applications and solve them. The whole process is automated and can be done with a single click thereby saving a lot of workload and downtime every time an error is encountered.

6. Methodology

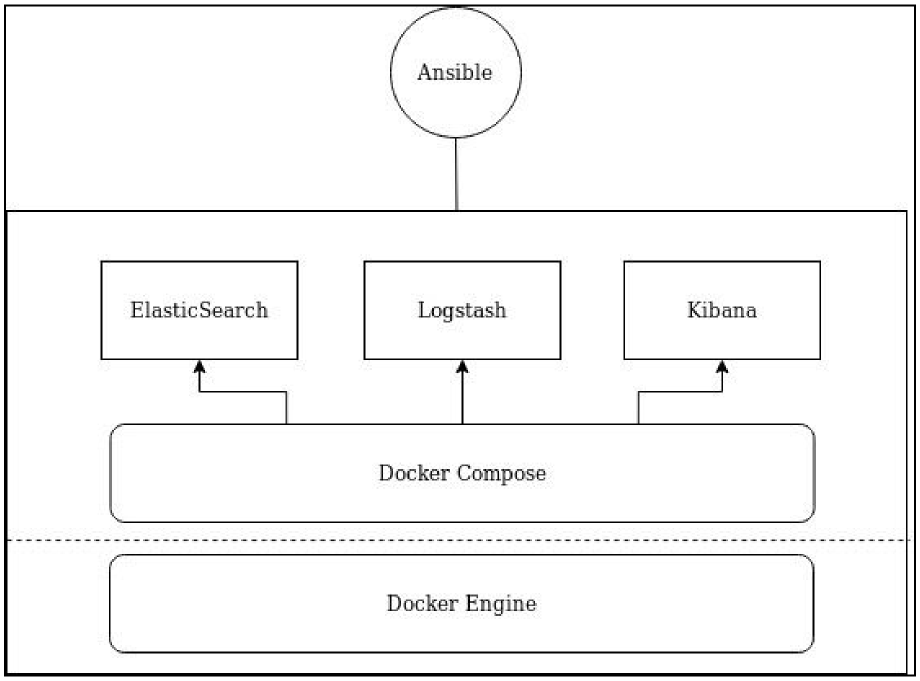


Figure 1: Ansible and Docker

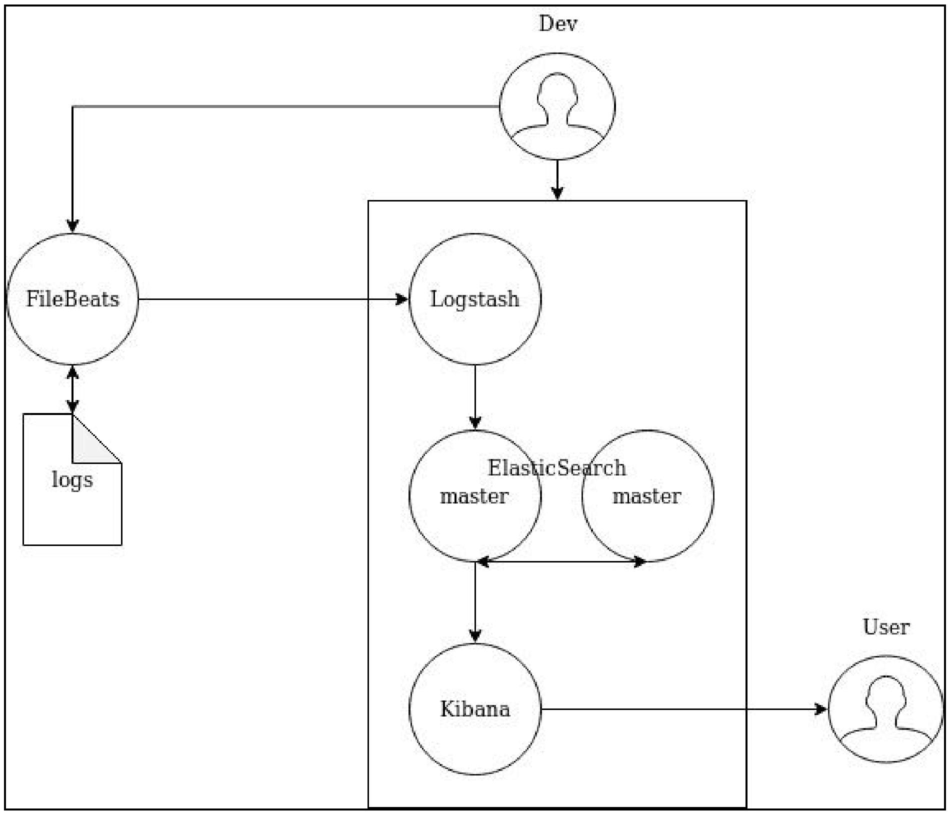


Figure 2: Elastic Search

Module 1: Configuring ELK setup

Setting up our logstash's .yml + .conf files

These are two files where we have to setup the configurations.

.yml — things which are related to logstash's system own settings. It resides in ./config

.conf — things that are related to logstash's parsing of received logs (which we call as an event for each line of the log) from filebeat.

logstash.yml — The contents inside the record must be self-explanatory. Define our ports, form of queue for incoming filebeat triggers, or other customisation here.

pipelines.yml — Logstash will automatically recognise this file as a definition of the different pipelines you want to configure just by the filename. Over here you define the path of the .conf file of each pipeline.

pipeline/*.conf — Define the filtering needed to parse the events coming through for each pipeline.

patterns — Define any custom grok patterns here. Reference this in our pipeline/*.conf file.

Setting up our elasticsearch's .yml file

elasticsearch.yml — Anything that is related to elasticsearch's system own settings. Resides in ./config (default location)

Setting up our kibana's .yml file

kibana.yml — Anything that is related to kibana's system own settings. Resides in ./config (default location)

Module 2: Setting up Filebeat

Setting up filebeat's .yml file.

All logs (on the host machine) that filebeat needs to ship will be mounted to filebeat's container, at path /usr/share/filebeat/dockerlogs/* within the container

Module 3: Setting up X-Pack Monitoring

To set this up, we need to ensure that xpack.monitoring configs are set up properly in:

filebeat.yml

logstash.yml

elasticsearch.yml

kibana.yml

They should already be configured in the .yml files in the repo.

Module 4: ANSIBLE Environment and ssh keys

Ansible lets us automate the advent, configuration and management of machines. Instead of manually retaining servers up to date, making configurations, transferring documents, and so on., you could use Ansible to automate this for groups of servers from one manipulating system.

For development and testing purposes, but, you might locate yourself by putting it in the stack repeatedly. While the installation method is straightforward sufficient and should take you no extra than 5 mins, a one-liner solution for installing and configuring the numerous components might be even higher.

It will allow us to configure by allowing users to write the scripts in yaml files with fixed or unchanged execution. The script will perform tasks and set up servers with direct use of any ssh service with each other.

- **Playbooks** — Execution entrypoint of ansible scripts

  Setting up Ansible s might be even higher.

  It will allow us to configure by allowing users to write the scripts in yaml files with fixed or unchanged execution. The script will perform tasks and set up servers with direct use of any ssh service with each other.

- **Playbooks — Execution entrypoint of ansible scripts**

  sudo apt update

  sudo apt install software-properties-common

  sudo apt-add-repository --yes --update ppa:ansible/ansible

  sudo apt install ansible

Ensure our SSH keys are set up

Try to ssh into our target server using ssh **<user>@<host_ip>** to check if ssh-agent has got our keys first. This is because ansible will try to SSH into the target servers specified in the ./hosts file with the keys ssh-agent has.

our ssh-user should have sudo privileges to allow creations of user and group ids that we will be using in the images we are going to build.

Configure our Docker credentials in .bashrc file.

Create new file .bashrc in the project dir with the vars.

This will be used to login to docker before attempting to push the images to the specified DOCKER_URL.

Module 5: Deploying the ELK Stack using Ansible

1. Override the variables you see fit in ./group_vars/elk.yml with our own variables
2. Execute the ansible script to build and push the image
3. Execute the ansible script to deploy the elk containers

Module 6 : Deploying the Filebeat clients using Ansible

1. Setup the variables in ./group_vars/elk-clients.yml with our own variables
2. Configure our environment IPs our filebeat containers will be residing in, in the ./hosts file.

3. Execute the ansible script to build and push the image
4. Execute the ansible script to deploy the filebeat containers

Module 7 : Setting up Curator to clean Elasticsearch indices regularly

We will be using a role based on an ansible-galaxy role developed by geerlingguy to setup our elasticsearch curator.

1. Configure curator cron settings in the ./group_vars/elk.yml file
2. Configure the curator's housekeeping configurations
3. Setup Curator

Module 8 : Check that all services are up and running

Use docker-compose ps or docker stats to check that all docker containers are working.

Module 9 : Setting up our index patterns and browsing logs

Lets test run our log shipper deployed in our Step 6 previously.

cd <path of where our app logs are at>

docker run -v $PWD/hello.log:/hello.log ubuntu echo hello world >> hello.log

Now hit our kibana URL with the <kibana_ip>:<port>, and go to the Management section.

We should see a new index created → dev.2020.xxxx. This index is created based on the <env>-<YYYY.MM.dd>. We can always change the way the index is created ./roles/elk/files/logstash/pipeline/app1.conf

1. Define your index pattern with dev-* . Index pattern will define the set of indexes that falls under that wildcard expression.
2. Tell Kibana to take the time reference from @timestamp field.
3. Create the Index Pattern.
4. Go the the Discover tab, and you should see the default dev-* index pattern selected. Select the message field, and you should see a 'hello world' which was the log message we used as the test run previously.

7. Design

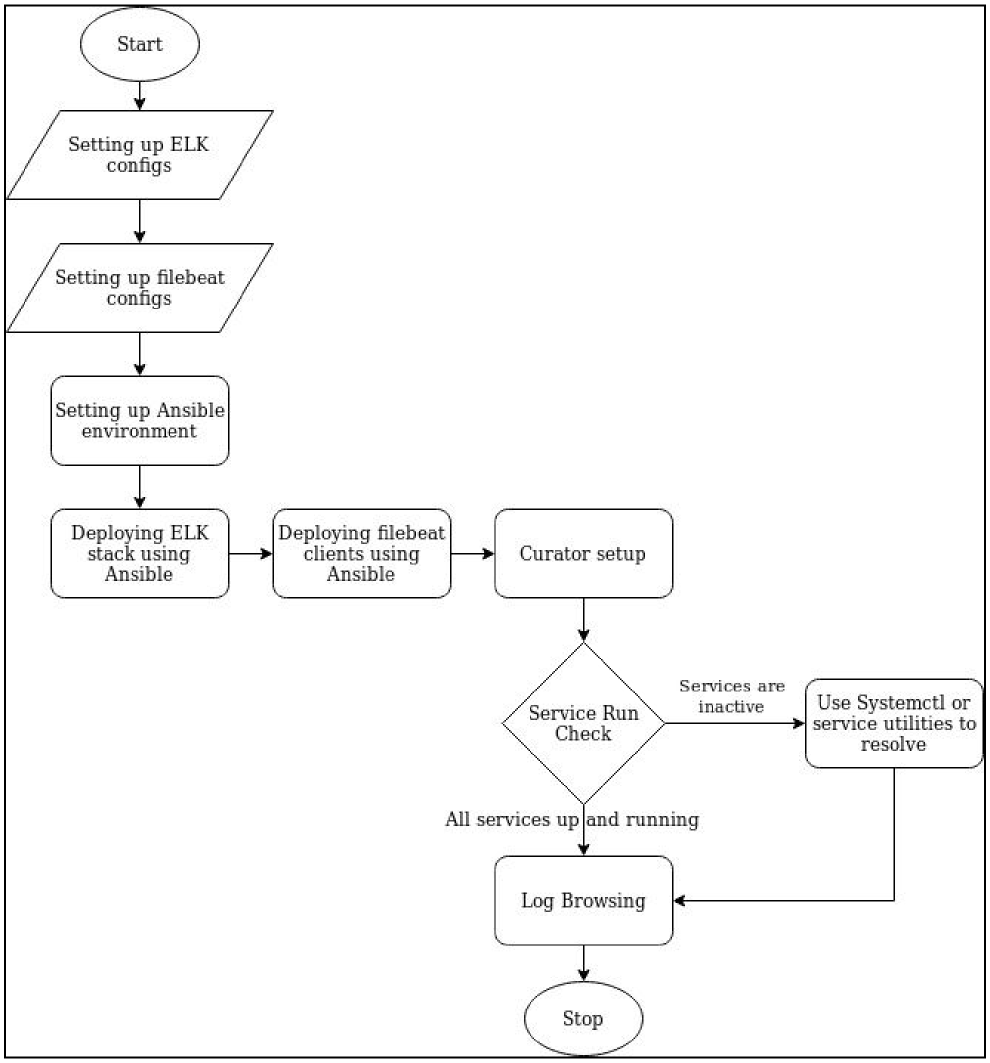
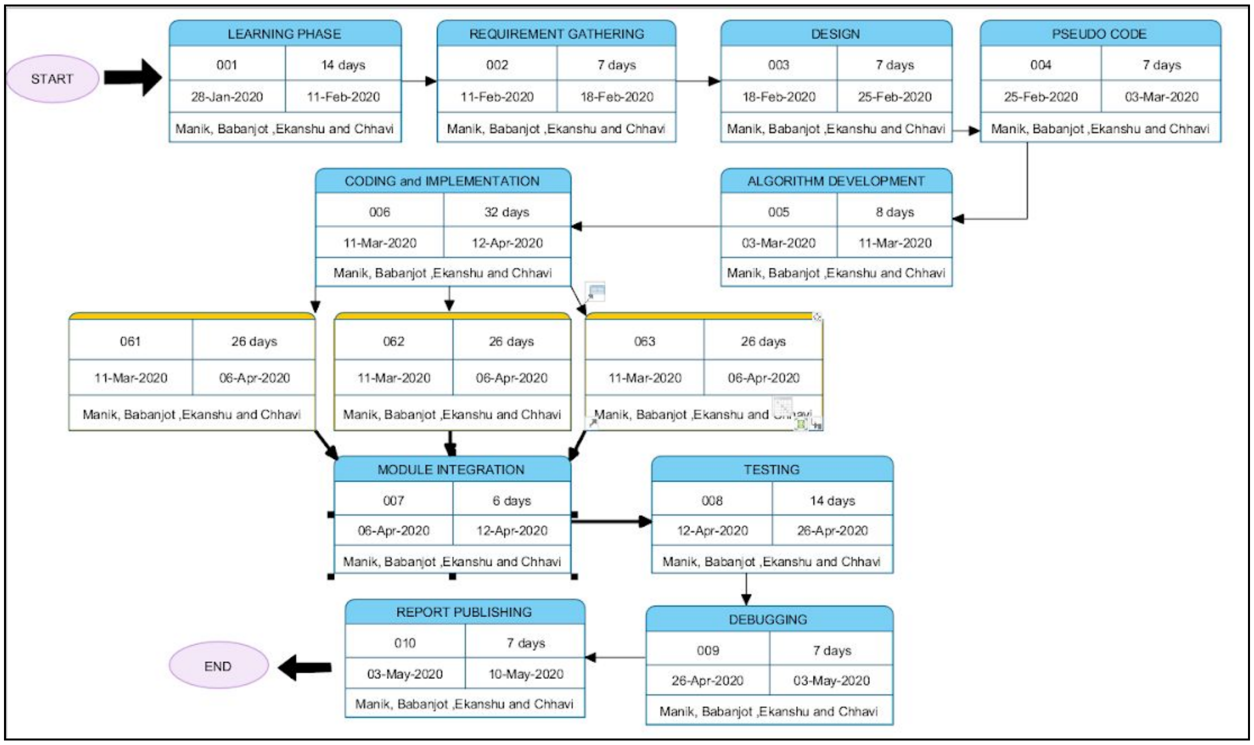

Start
Setting up ELK configs
Setting up filebeat configs
Setting up Ansible environment
Deploying ELK stack using Ansible
Deploying filebeat clients using Ansible
Curator setup
Service Run Check
Services are inactive
Use Systemctl or service utilities to resolve
All services up and running
Log Browsing
Stop

Figure 3: Design Philosophy

8. System Requirements (Software/Hardware)

- **Hardware:**
  - Intel(R) Core(TM) i3-3200 CPU @ 1.5 GHz
  - 2 GB RAM
  - System type is 32/64-bit Operating System
- **Software:**
  - Linux based operating System
  - Docker
  - Ansible

9. **Schedule** (**P**roject **E**valuation and **R**eview **T**echnique Chart**)**



10. Reference Links

[a].https://docs.ansible.com/

[b].https://www.elastic.co/guide

[c].https://jenkins.io/doc/

[d].https://docs.docker.com

[e].https://docs.aws.amazon.com

[f]. https://guides.github.com