MONITORING AND LOGGING MANAGEMENT SYSTEM – ONE STOP SOLUTION

By

Vaishal Shah

INTRODUCTION

This is just a demo project I made to showcase my DevOps skills. I integrated different tools and technologies to make things more efficient and less time-consuming.

ELK stack is deployed to make a one-stop solution for monitoring and logging management. ELK stack is deployed using various DevOps technologies. Tech Stack contains; Terraform, Docker, Ansible, GitHub, AWS cloud, and CI/CD. I have used these tools to take the benefits of each technology and make it easier for tech savvies to deploy monitoring and logging infrastructure.

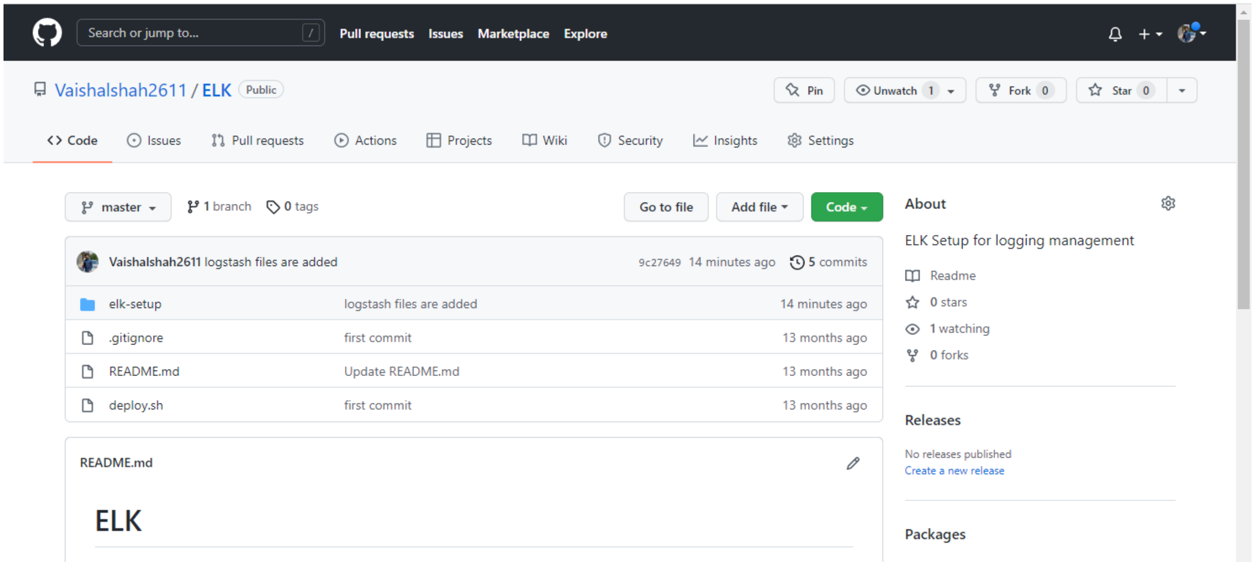
TECHNOLOGY STACK

GitHub - ELK : Setup done.
Terraform - Done. Infrastructure as a code.
Docker - Done. Containerized application.
Ansible - In progress. Configuration management for ELK.
AWS Cloud - Done. Resources are deployed on the AWS cloud.
ELK stack - Done. Tested thoroughly.
CI/CD - In near future (within a week).

GITHUB:

GitHub is the main software I have used as a versioning control system for this project. It consists of an entire repository of the ELK architecture.

It is used for enabling the code version and for continuous integration purposes. People can start contributing to my repository in the future for enhancement.



Vaishalshah2611 / ELK Public

Vaishalshah2611 logstash files are added 9c27649 14 minutes ago 5 commits

elk-setup	logstash files are added	14 minutes ago
.gitignore	first commit	13 months ago
README.md	Update README.md	13 months ago
deploy.sh	first commit	13 months ago

About

ELK Setup for logging management

Releases

No releases published
Create a new release

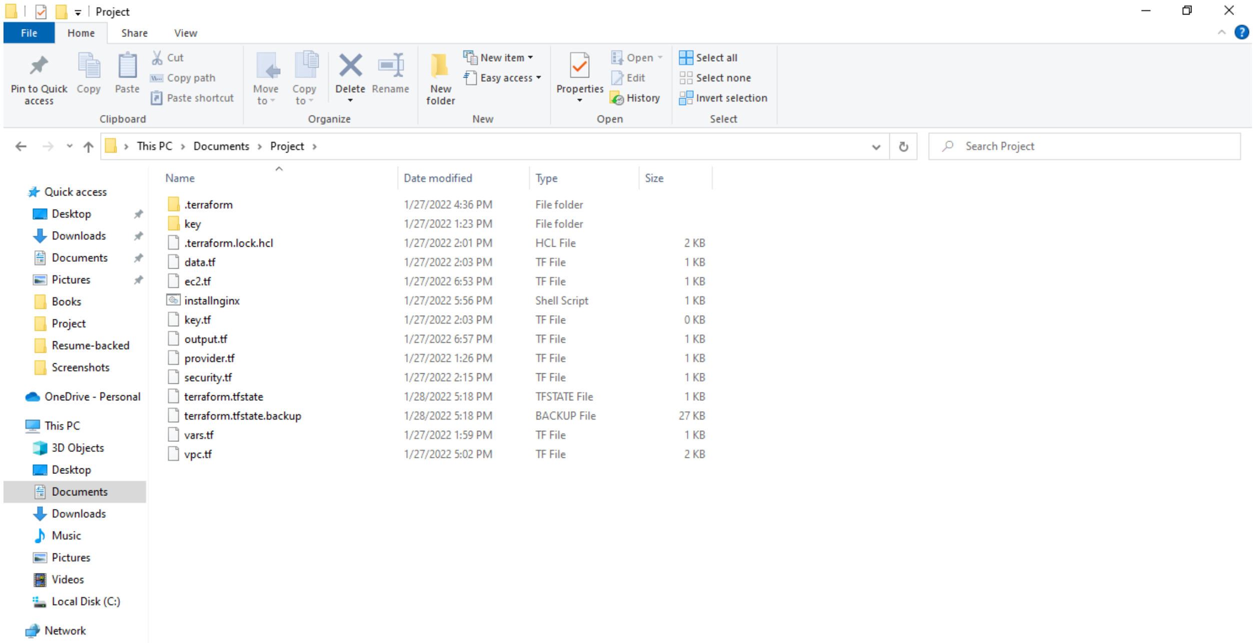
README.md

ELK

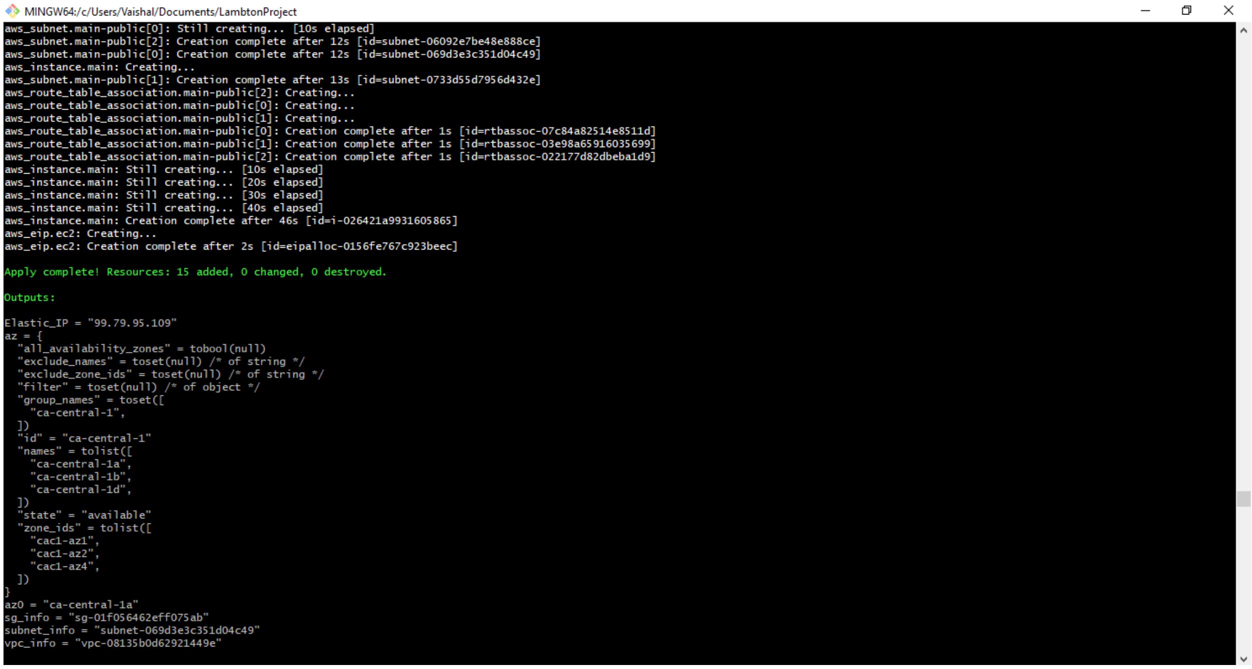
TERRAFORM:

Terraform is used to enable Infrastructure as Code. The infrastructure deployed on AWS using Terraform. Currently, terraform modules are not being used as it becomes easy to pull the defined resources. So, as a beginner, I insisted on writing the code from scratch so that it will help me to clear concepts.

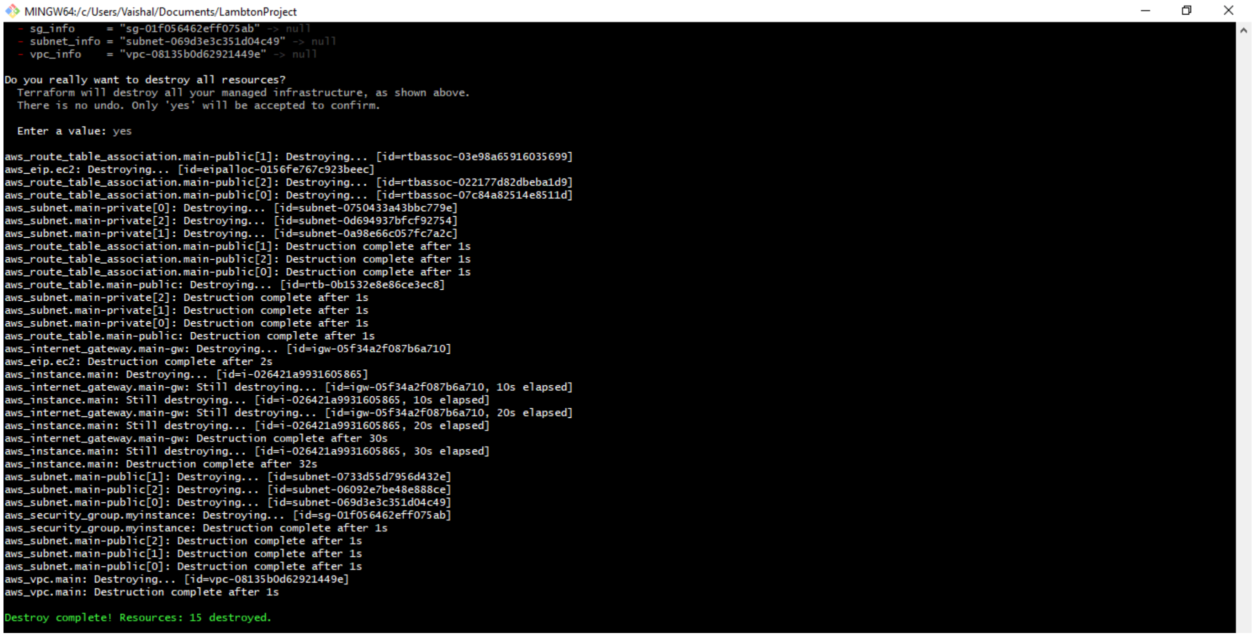
The below screenshot shows the code directory, creation, and deletion of the resources using terraform:



Name	Date modified	Type	Size
.terraform	1/27/2022 4:36 PM	File folder	
key	1/27/2022 1:23 PM	File folder	
.terraform.lock.hcl	1/27/2022 2:01 PM	HCL File	2 KB
data.tf	1/27/2022 2:03 PM	TF File	1 KB
ec2.tf	1/27/2022 6:53 PM	TF File	1 KB
installnginx	1/27/2022 5:56 PM	Shell Script	1 KB
key.tf	1/27/2022 2:03 PM	TF File	0 KB
output.tf	1/27/2022 6:57 PM	TF File	1 KB
provider.tf	1/27/2022 1:26 PM	TF File	1 KB
security.tf	1/27/2022 2:15 PM	TF File	1 KB
terraform.tfstate	1/28/2022 5:18 PM	TFSTATE File	1 KB
terraform.tfstate.backup	1/28/2022 5:18 PM	BACKUP File	27 KB
vars.tf	1/27/2022 1:59 PM	TF File	1 KB
vpc.tf	1/27/2022 5:02 PM	TF File	2 KB



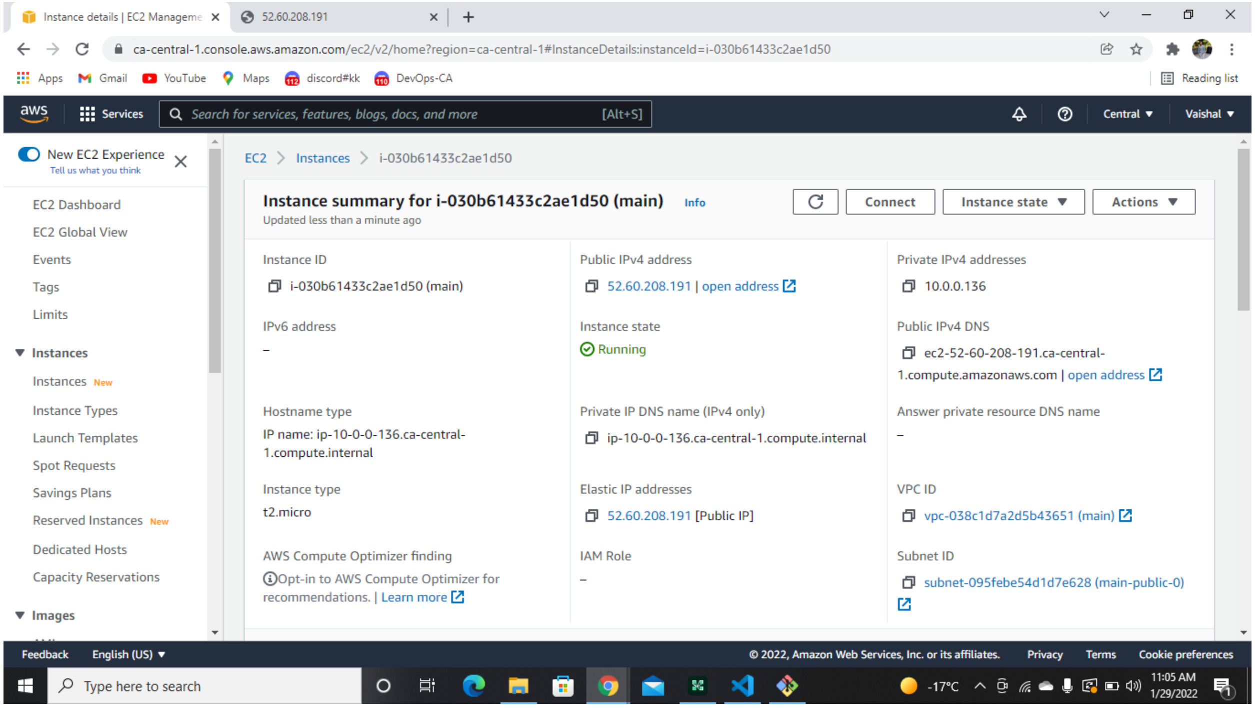
```
aws_subnet.main-public[0]: Still creating... [10s elapsed]
aws_subnet.main-public[2]: Creation complete after 12s [id=subnet-06092e7be48e888ce]
aws_subnet.main-public[0]: Creation complete after 12s [id=subnet-069d3e3c351d04c49]
aws_instance.main: Creating...
aws_subnet.main-public[1]: Creation complete after 13s [id=subnet-0733d55d7956d432e]
aws_route_table_association.main-public[2]: Creating...
aws_route_table_association.main-public[0]: Creating...
aws_route_table_association.main-public[1]: Creating...
aws_route_table_association.main-public[0]: Creation complete after 1s [id=rtbassoc-07c84a82514e8511d]
aws_route_table_association.main-public[1]: Creation complete after 1s [id=rtbassoc-03e98a65916035699]
aws_route_table_association.main-public[2]: Creation complete after 1s [id=rtbassoc-022177d82dbeba1d9]
aws_instance.main: Still creating... [10s elapsed]
aws_instance.main: Still creating... [20s elapsed]
aws_instance.main: Still creating... [30s elapsed]
aws_instance.main: Still creating... [40s elapsed]
aws_instance.main: Creation complete after 46s [id=i-026421a9931605865]
aws_eip.ec2: Creating...
aws_eip.ec2: Creation complete after 2s [id=eipalloc-0156fe767c923beec]

Apply complete! Resources: 15 added, 0 changed, 0 destroyed.

Outputs:

Elastic_IP = "99.79.95.109"
az = {
  "all_availability_zones" = tobool(null)
  "exclude_names" = toset(null) /* of string */
  "exclude_zone_ids" = toset(null) /* of string */
  "filter" = toset(null) /* of object */
  "group_names" = toset([
    "ca-central-1",
  ])
  "id" = "ca-central-1"
  "names" = tolist([
    "ca-central-1a",
    "ca-central-1b",
    "ca-central-1d",
  ])
  "state" = "available"
  "zone_ids" = tolist([
    "cac1-az1",
    "cac1-az2",
    "cac1-az4",
  ])
}
az0 = "ca-central-1a"
sg_info = "sg-01f056462eff075ab"
subnet_info = "subnet-069d3e3c351d04c49"
vpc_info = "vpc-08135b0d62921449e"
```



```
  - sg_info     = "sg-01f056462eff075ab" -> null
  - subnet_info = "subnet-069d3e3c351d04c49" -> null
  - vpc_info    = "vpc-08135b0d62921449e" -> null

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

aws_route_table_association.main-public[1]: Destroying... [id=rtbassoc-03e98a65916035699]
aws_eip.ec2: Destroying... [id=eipalloc-0156fe767c923beec]
aws_route_table_association.main-public[2]: Destroying... [id=rtbassoc-022177d82dbeba1d9]
aws_route_table_association.main-public[0]: Destroying... [id=rtbassoc-07c84a82514e8511d]
aws_subnet.main-private[0]: Destroying... [id=subnet-0750433a43bbc779e]
aws_subnet.main-private[2]: Destroying... [id=subnet-0d694937bfcf92754]
aws_subnet.main-private[1]: Destroying... [id=subnet-0a98e66c057fc7a2c]
aws_route_table_association.main-public[1]: Destruction complete after 1s
aws_route_table_association.main-public[2]: Destruction complete after 1s
aws_route_table_association.main-public[0]: Destruction complete after 1s
aws_route_table.main-public: Destroying... [id=rtb-0b1532e8e86ce3ec8]
aws_subnet.main-private[2]: Destruction complete after 1s
aws_subnet.main-private[1]: Destruction complete after 1s
aws_subnet.main-private[0]: Destruction complete after 1s
aws_route_table.main-public: Destruction complete after 1s
aws_internet_gateway.main-gw: Destroying... [id=igw-05f34a2f087b6a710]
aws_eip.ec2: Destruction complete after 2s
aws_instance.main: Destroying... [id=i-026421a9931605865]
aws_internet_gateway.main-gw: Still destroying... [id=igw-05f34a2f087b6a710, 10s elapsed]
aws_instance.main: Still destroying... [id=i-026421a9931605865, 10s elapsed]
aws_internet_gateway.main-gw: Still destroying... [id=igw-05f34a2f087b6a710, 20s elapsed]
aws_instance.main: Still destroying... [id=i-026421a9931605865, 20s elapsed]
aws_internet_gateway.main-gw: Destruction complete after 30s
aws_instance.main: Still destroying... [id=i-026421a9931605865, 30s elapsed]
aws_instance.main: Destruction complete after 32s
aws_subnet.main-public[1]: Destroying... [id=subnet-0733d55d7956d432e]
aws_subnet.main-public[2]: Destroying... [id=subnet-06092e7be48e888ce]
aws_subnet.main-public[0]: Destroying... [id=subnet-069d3e3c351d04c49]
aws_security_group.myinstance: Destroying... [id=sg-01f056462eff075ab]
aws_security_group.myinstance: Destruction complete after 1s
aws_subnet.main-public[2]: Destruction complete after 1s
aws_subnet.main-public[1]: Destruction complete after 1s
aws_subnet.main-public[0]: Destruction complete after 1s
aws_vpc.main: Destroying... [id=vpc-08135b0d62921449e]
aws_vpc.main: Destruction complete after 1s

Destroy complete! Resources: 15 destroyed.
```

AWS CLOUD:

All resources are deployed on the AWS cloud via Terraform. Resources used in this demo are: VPC, Subnets public/private, Route-table, Internet Gateway, Security Group, NACL, Elastic IP, EBS, Key Pair, EC2 instance. EC2 instance of type T3.medium is used and the flavor is Ubuntu 20.04.



Instance summary for i-030b61433c2ae1d50 (main) Info

Updated less than a minute ago

Instance ID
i-030b61433c2ae1d50 (main)

Public IPv4 address
52.60.208.191 | open address

Private IPv4 addresses
10.0.0.136

IPv6 address
–

Instance state
Running

Public IPv4 DNS
ec2-52-60-208-191.ca-central-1.compute.amazonaws.com | open address

Hostname type
IP name: ip-10-0-0-136.ca-central-1.compute.internal

Private IP DNS name (IPv4 only)
ip-10-0-0-136.ca-central-1.compute.internal

Answer private resource DNS name
–

Instance type
t2.micro

Elastic IP addresses
52.60.208.191 [Public IP]

VPC ID
vpc-038c1d7a2d5b43651 (main)

AWS Compute Optimizer finding
Opt-in to AWS Compute Optimizer for recommendations. | Learn more

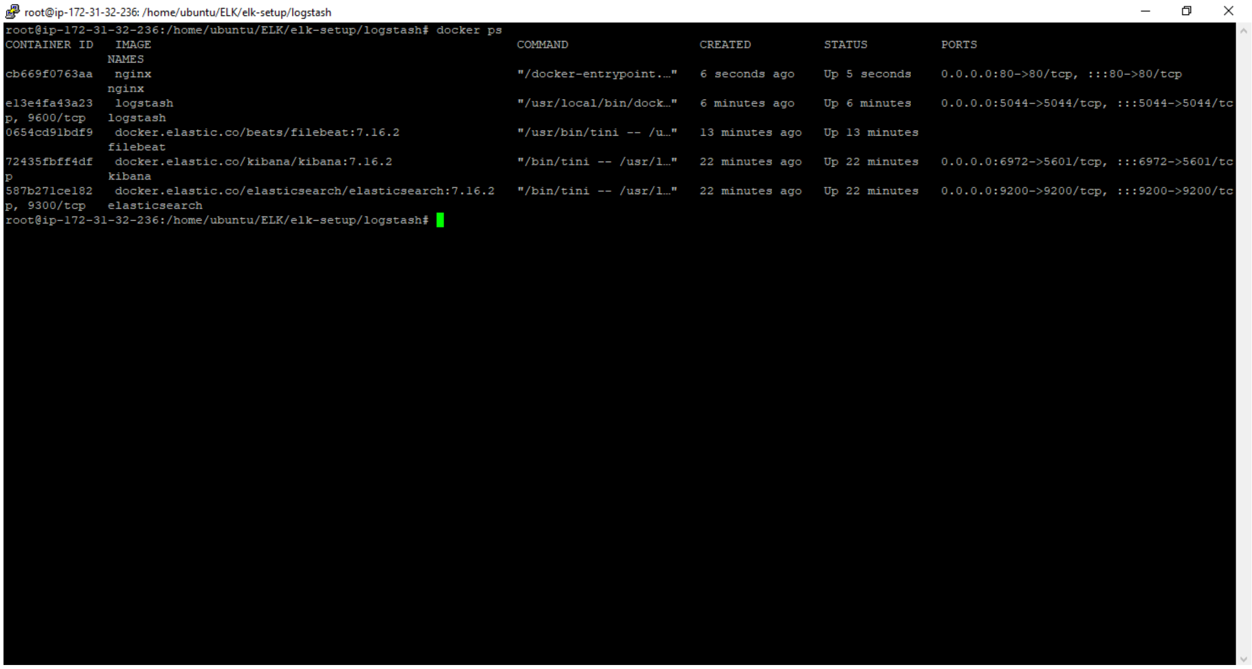
IAM Role
–

Subnet ID
subnet-095febe54d1d7e628 (main-public-0)

DOCKER:

Docker, being a containerization technology, I have used it to containerize the whole ELK stack along with Nginx. Nginx is a web server whose logs are collected and stored on Elasticsearch. These logs are procured by Kibana which in turn helps us to visualize the data.

The stack is deployed using docker and docker-compose. Docker is used to make the application/ELK stack portable and platform independent.

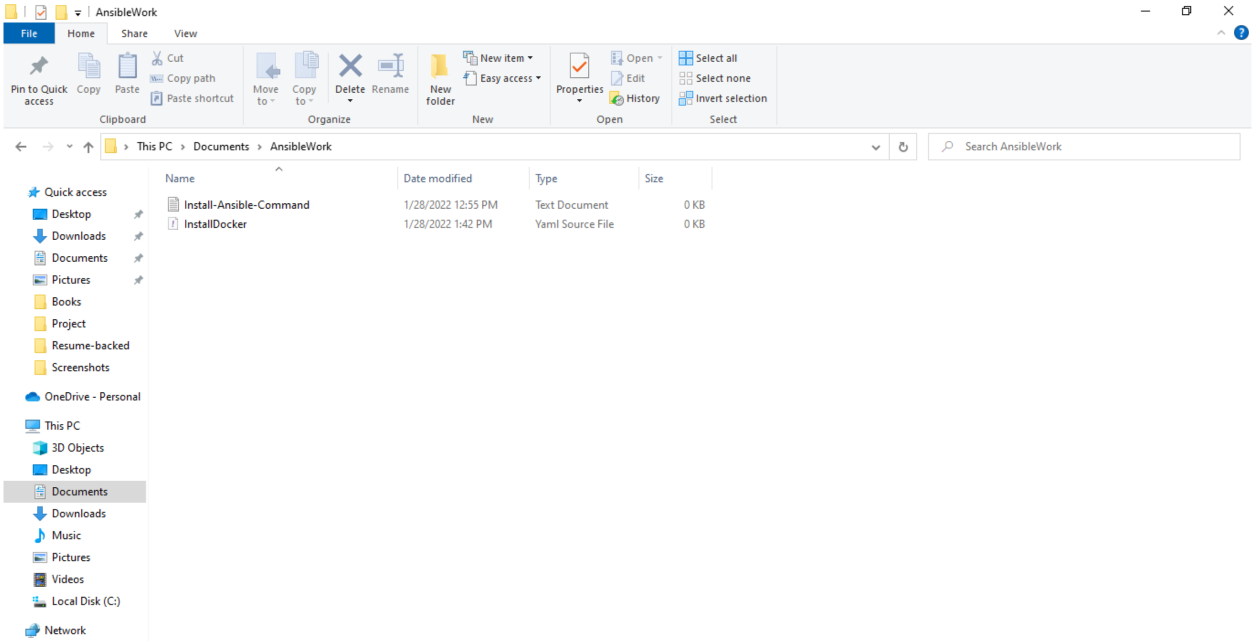


root@ip-172-31-32-236: /home/ubuntu/ELK/elk-setup/logstash

```
root@ip-172-31-32-236:/home/ubuntu/ELK/elk-setup/logstash# docker ps
CONTAINER ID   IMAGE                                                  COMMAND                  CREATED          STATUS          PORTS
              NAMES
cb669f0763aa   nginx                                                  "/docker-entrypoint.…"   6 seconds ago    Up 5 seconds    0.0.0.0:80->80/tcp, :::80->80/tcp
              nginx
e13e4fa43a23   logstash                                               "/usr/local/bin/dock…"   6 minutes ago    Up 6 minutes    0.0.0.0:5044->5044/tcp, :::5044->5044/tc
p, 9600/tcp   logstash
0654cd91bdf9   docker.elastic.co/beats/filebeat:7.16.2                "/usr/bin/tini -- /u…"   13 minutes ago   Up 13 minutes
              filebeat
72435fbff4df   docker.elastic.co/kibana/kibana:7.16.2                 "/bin/tini -- /usr/l…"   22 minutes ago   Up 22 minutes   0.0.0.0:6972->5601/tcp, :::6972->5601/tc
p             kibana
587b271ce182   docker.elastic.co/elasticsearch/elasticsearch:7.16.2   "/bin/tini -- /usr/l…"   22 minutes ago   Up 22 minutes   0.0.0.0:9200->9200/tcp, :::9200->9200/tc
p, 9300/tcp   elasticsearch
root@ip-172-31-32-236:/home/ubuntu/ELK/elk-setup/logstash#
```

ANSIBLE:

Ansible is used to automate configurations of the ELK stack. I have written an Ansible playbook using YAML. So, we can run playbooks to multiple servers if needed, without any intervention into the virtual machine remotely. For now, I have written a playbook for installing docker on a remote machine. In near future, I will write an entire ELK configuration playbook that can be deployed remotely using the Ansible playbook.



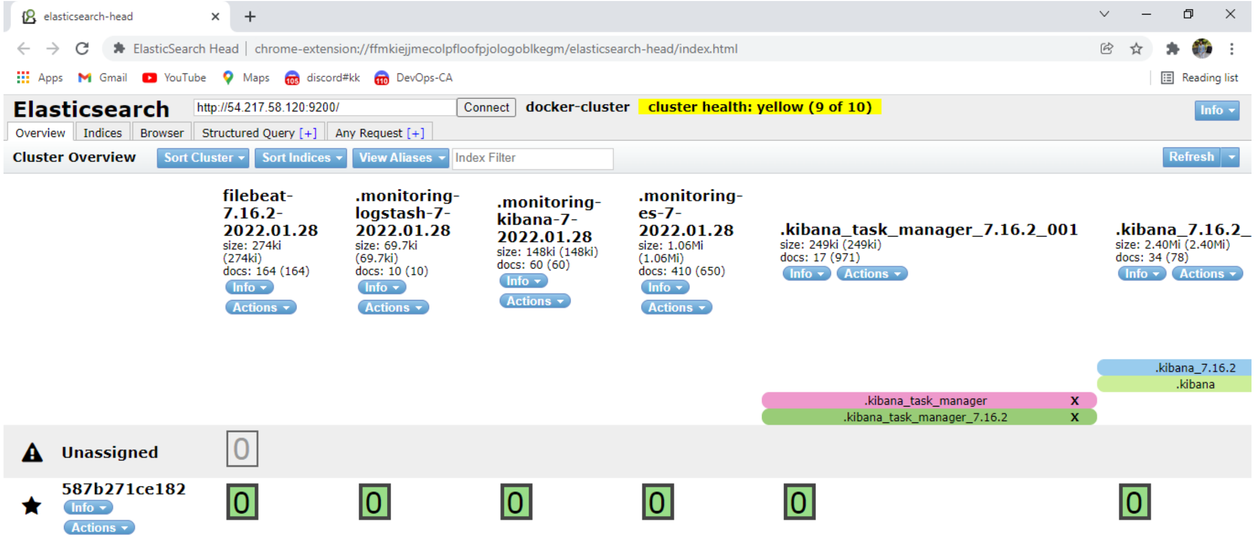
Name	Date modified	Type	Size
Install-Ansible-Command	1/28/2022 12:55 PM	Text Document	0 KB
InstallDocker	1/28/2022 1:42 PM	Yaml Source File	0 KB

ELK STACK:

It stands for Elastic, Logstash, Kibana. The stack is deployed to make a one-stop self-hosted solution for monitoring, and logging management systems.

In this demo, Filebeat collects the logs of various docker containers and sends them out to Logstash, where filtering/indexing of the logs takes place and then Logstash sends it to Elasticsearch. All the data gets stored in Elasticsearch. From Elasticsearch, Kibana reads the data and makes it available for visualizing the logs as per our needs.

The attached screenshots show the Elasticsearch workspace, and the index management enabled on Kibana to process and read the logs from Elasticsearch. Lastly, the Kibana dashboard shows where data/logs can be visualized.


elasticsearch-head
ElasticSearch Head | chrome-extension://ffmkiejjmecolpfloofpjologoblkegm/elasticsearch-head/index.html
Apps Gmail YouTube Maps discord#kk DevOps-CA Reading list
Connect
Info
Overview Indices Browser Structured Query [+] Any Request [+]
Cluster Overview Sort Cluster Sort Indices View Aliases Index Filter Refresh

Elasticsearch http://54.217.58.120:9200/ docker-cluster cluster health: yellow (9 of 10)

filebeat-7.16.2-2022.01.28
size: 274ki (274ki)
docs: 164 (164)

.monitoring-logstash-7-2022.01.28
size: 69.7ki (69.7ki)
docs: 10 (10)

.monitoring-kibana-7-2022.01.28
size: 148ki (148ki)
docs: 60 (60)

.monitoring-es-7-2022.01.28
size: 1.06Mi (1.06Mi)
docs: 410 (650)

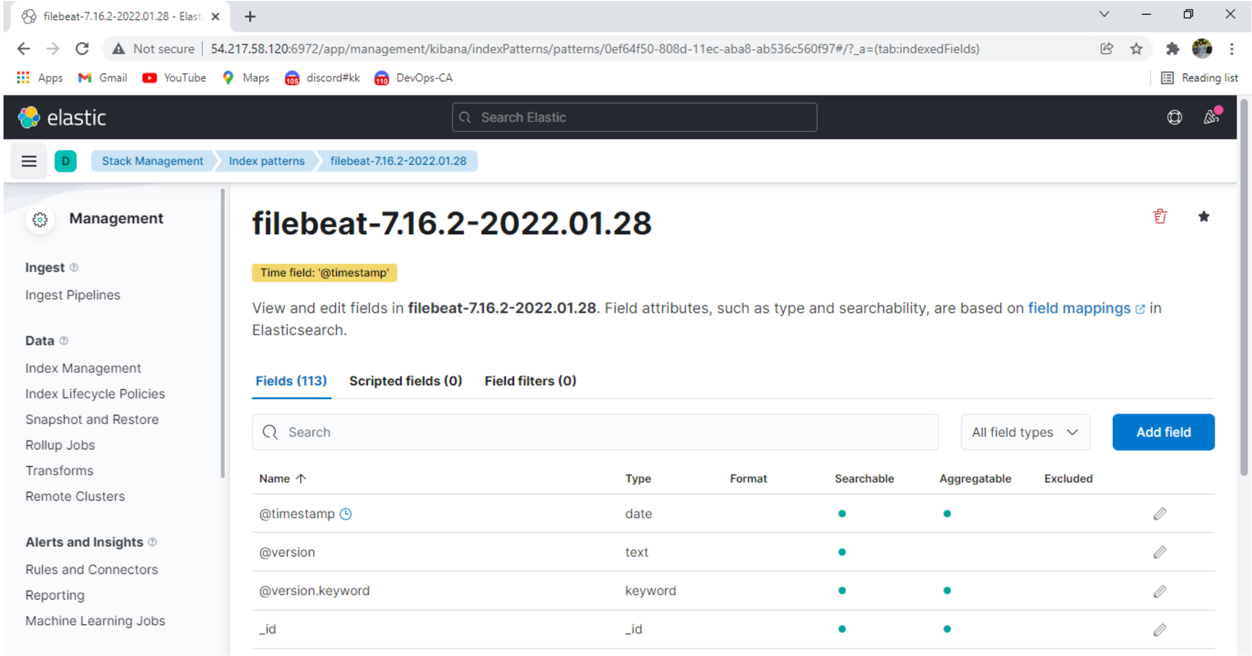
.kibana_task_manager_7.16.2_001
size: 249ki (249ki)
docs: 17 (971)

.kibana_7.16.2_
size: 2.40Mi (2.40Mi)
docs: 34 (78)

.kibana_7.16.2
.kibana
.kibana_task_manager x
.kibana_task_manager_7.16.2 x

Unassigned 0

587b271ce182 0 0 0 0 0 0


filebeat-7.16.2-2022.01.28 - Elast
Not secure | 54.217.58.120:6972/app/management/kibana/indexPatterns/patterns/0ef64f50-808d-11ec-aba8-ab536c560f97#/?_a=(tab:indexedFields)
Apps Gmail YouTube Maps discord#kk DevOps-CA Reading list
elastic
Search Elastic
D Stack Management Index patterns filebeat-7.16.2-2022.01.28
Management
Ingest
Ingest Pipelines
Data
Index Management
Index Lifecycle Policies
Snapshot and Restore
Rollup Jobs
Transforms
Remote Clusters
Alerts and Insights
Rules and Connectors
Reporting
Machine Learning Jobs
Search
All field types
Add field

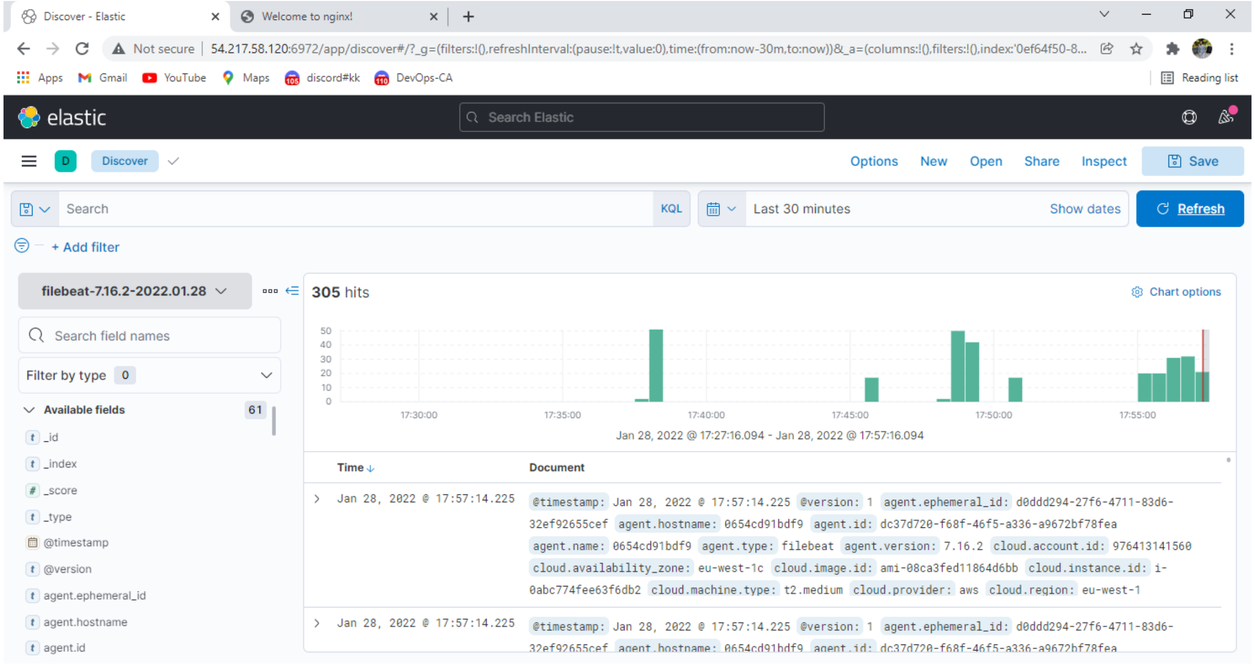
filebeat-7.16.2-2022.01.28

Time field: '@timestamp'

View and edit fields in **filebeat-7.16.2-2022.01.28**. Field attributes, such as type and searchability, are based on field mappings in Elasticsearch.

Fields (113) Scripted fields (0) Field filters (0)

Name ↑	Type	Format	Searchable	Aggregatable	Excluded
@timestamp	date		●	●	
@version	text		●		
@version.keyword	keyword		●	●	
_id	_id		●	●	



```
Jan 28, 2022 @ 17:57:14.225   @timestamp: Jan 28, 2022 @ 17:57:14.225 @version: 1 agent.ephemeral_id: d0ddd294-27f6-4711-83d6-
                              32ef92655cef agent.hostname: 0654cd91bdf9 agent.id: dc37d720-f68f-46f5-a336-a9672bf78fea
                              agent.name: 0654cd91bdf9 agent.type: filebeat agent.version: 7.16.2 cloud.account.id: 976413141560
                              cloud.availability_zone: eu-west-1c cloud.image.id: ami-08ca3fed11864d6bb cloud.instance.id: i-
                              0abc774fee63f6db2 cloud.machine.type: t2.medium cloud.provider: aws cloud.region: eu-west-1

Jan 28, 2022 @ 17:57:14.225   @timestamp: Jan 28, 2022 @ 17:57:14.225 @version: 1 agent.ephemeral_id: d0ddd294-27f6-4711-83d6-
                              32ef92655cef agent.hostname: 0654cd91bdf9 agent.id: dc37d720-f68f-46f5-a336-a9672bf78fea
```

CI/CD:

I will make it CI/CD enabled using either GitHub actions or Jenkins. It is the future plan, which will be ready soon. Which will hold the terraform code. So, infrastructure will be automatically deployed on AWS with just a simple git push command.

REFERENCE:

Blog written by me: https://faun.pub/elk-elasticsearch-logstash-kibana-conceptual-tutorial-for-beginners-2a7a827305b8