

Elasticsearch Assignment

Q.1 What did you choose to automate the provisioning and bootstrapping of the instance? Why?

Solution 1:

Automation: Used Terraform for provisioning the AWS resources.

BootStraping: Used a Shell Scripting for installing & configuring the elasticsearch.

Solution 2:

Automation: Used Terraform for provisioning the AWS resources.

BootStraping: Used an Ansible for installing & configuring the elasticsearch.

Q.2 How did you choose to secure ElasticSearch? Why?

We can secure elasticsearch using SSL Certificate/TLS Connections, Security Groups, NACL, WAF.

Q.3 How would you monitor this instance? What metrics would you monitor?

We can Monitor the Instances using AWS Cloudwatch Service. Also we can configure Prometheus, Grafana for visualization. We can monitor metrics like CPU Utilization, Memory Utilization and Disk Utilization of the instance.

Q.4 Could you extend your solution to launch a secure cluster of ElasticSearch nodes? What would need to change to support this use case?

We can secure our ElasticSearch nodes Using SSL/TLS Communication. We can create cluster in private subnet and use VPN to access elasticsearch cluster. Also we can configure Basic Authentication.

Q.5 Could you extend your solution to replace a running ElasticSearch instance with little or no downtime? How?

We can make a cluster of elasticsearch and enable replication.

Q.6 Was it a priority to make your code well structured, extensible, and reusable?

It is priority to make our code well structured. So later we can modify our code easily with very less efforts.

Q.7 What sacrifices did you make due to time?

SSL Automation connection is pending for DNS. But it's working with default SSL with IP.

Prerequisites

[AWS CLI](#) - (Install AWS CLI & Configure aws profile)

```
aws configure --profile <profile-name>
```

[Terraform](#) - (Install Terraform)

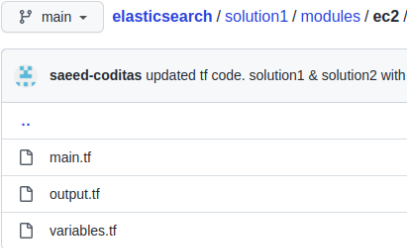
[Ansible](#) - (Install Ansible)

Folder Structure

I have solved assignment using two ways both are in separate folders

There are two folders **solution1** & **solution2**

Solution1:

solution1 folder content	ec2 module folder content	vpc module folder content
 <p>elasticsearch / solution1 /</p> <ul style="list-style-type: none">..modulesbackend-provider.tfmain.tfoutput.tfuserdata.shvariables.tf	 <p>elasticsearch / solution1 / modules / ec2 /</p> <ul style="list-style-type: none">..main.tfoutput.tfvariables.tf	 <p>elasticsearch / solution1 / modules / vpc /</p> <ul style="list-style-type: none">..main.tfoutput.tfvariables.tf


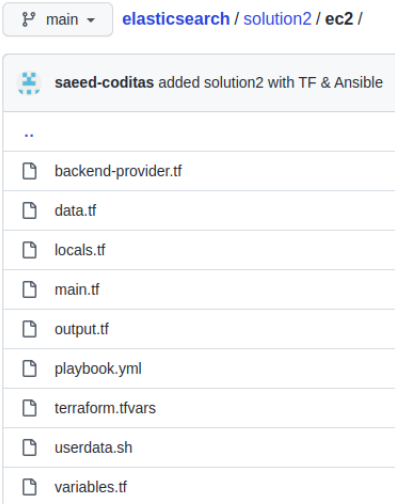
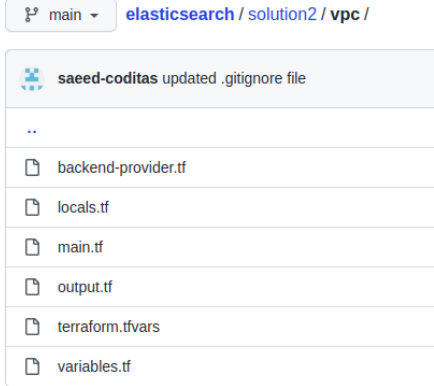
Steps to deploy elasticsearch:

Make proper changes to variables.tf files inside module/ec2 & module/vpc according to your requirement
userdata.sh file is there in solution1 folder for installation & configuration elasticsearch

change bucket name in last line which saves elasticsearch password to S3

```
cd elasticsearch/solution1
terraform init
terraform plan
terraform apply
```

Solution2:

solution2 folder content	ec2 folder content	vpc module folder content
		

Steps to deploy elasticsearch:

Make proper changes to terraform.tfvars & locals.tf files inside ec2 & vpc according to your requirement

playbook.yml is there in ec2 folder for installation & configuration elasticsearch

change bucket name in last line which saves elasticsearch password to S3

Create VPC:

```
cd elasticsearch/solution2/vpc
terraform init
terraform workspace new dev
terraform plan
terraform apply
```

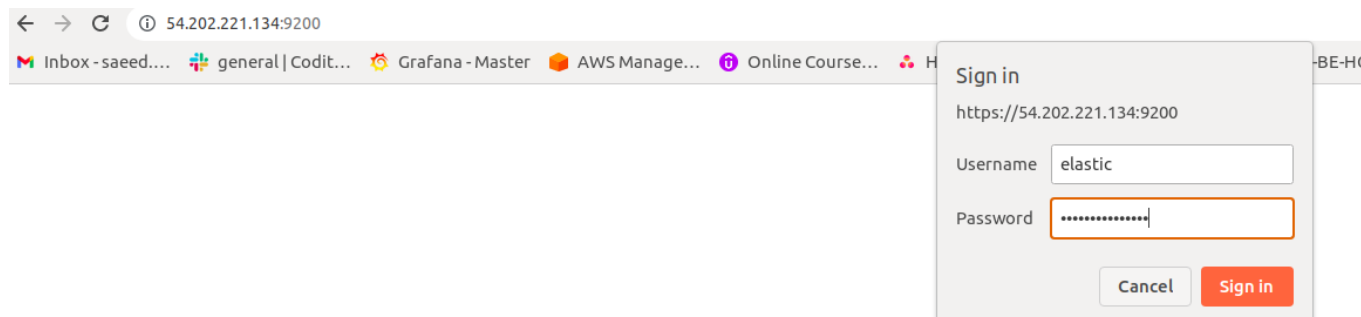
Create Elasticsearch Instance:

```
cd elasticsearch/solution2/ec2
terraform init
terraform workspace new dev
```

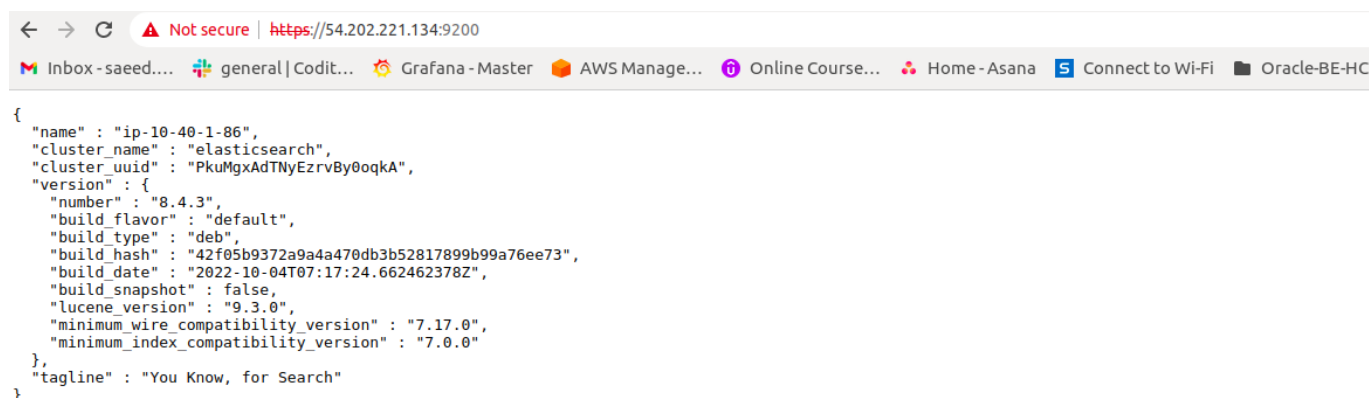
```
terraform plan
terraform apply
```

Screenshots

Login with Basic Auth UserName & Password:



Screen after login:



Created an Elasticsearch Index Using cURL & View:



The screenshot shows a web browser window with a REST client interface. The address bar displays the URL `https://54.202.221.134:9200/body_index?pretty`. The browser's tab bar includes several tabs: "Inbox - saeed...", "general | Codit...", "Grafana - Master", "AWS Manage...", "Online Course...", "Home - Asana", "Connect to Wi-Fi", and "Oracle-BE-HC". The main content area displays a JSON response, which is a mapping of the "body_index" field. The JSON structure is as follows:

```
{
  "body_index" : {
    "aliases" : { },
    "mappings" : {
      "properties" : {
        "created" : {
          "type" : "date",
          "format" : "epoch_second"
        },
        "field_1" : {
          "type" : "text"
        },
        "field_2" : {
          "type" : "integer"
        },
        "field_3" : {
          "type" : "boolean"
        }
      }
    }
  },
  "settings" : {
    "index" : {
      "routing" : {
        "allocation" : {
          "include" : {
            "_tier_preference" : "data_content"
          }
        }
      },
      "number_of_shards" : "3",
      "provided_name" : "body_index",
      "creation_date" : "1667332001942",
      "number_of_replicas" : "1",
      "uuid" : "tfKP2-2xRuCtndqnUJVJLw",
      "version" : {
        "created" : "8040399"
      }
    }
  }
}
```