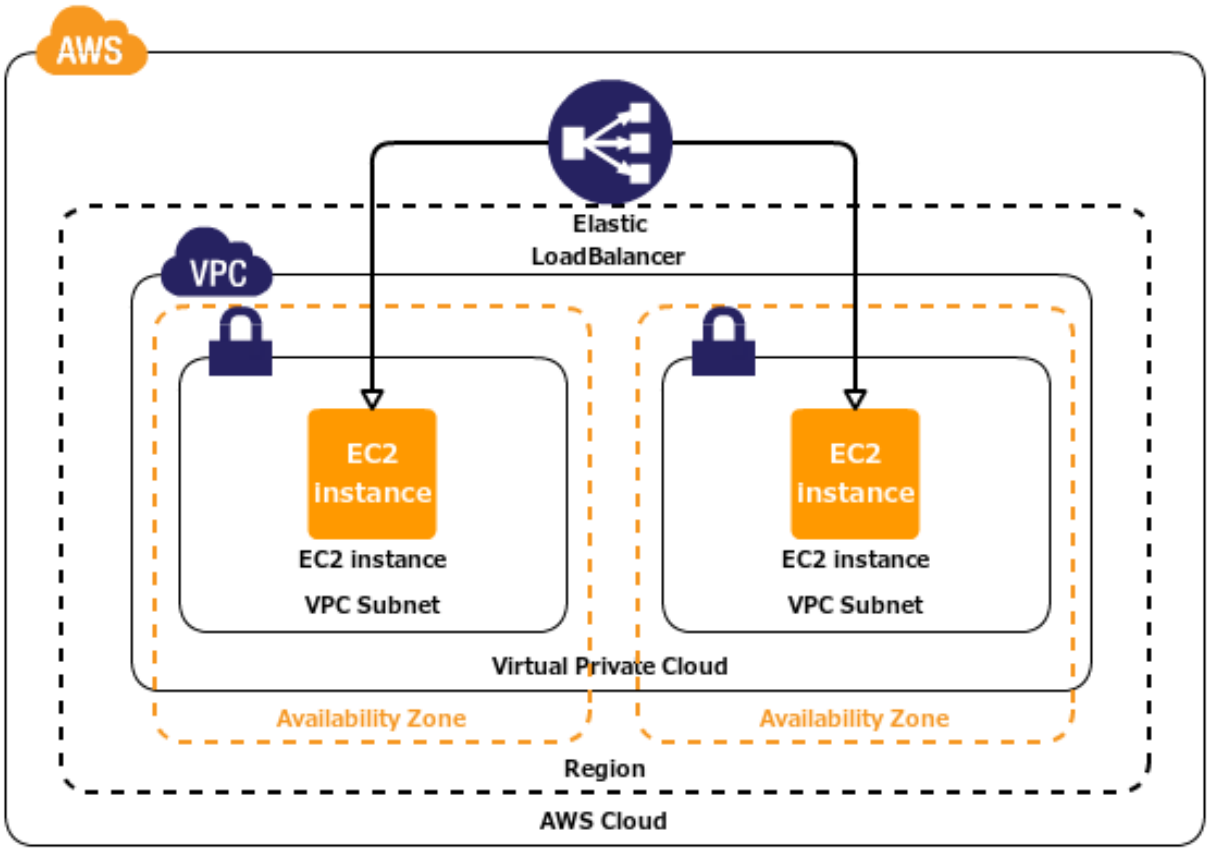
Create a SonarQube stack Ec2 using Terraform.

In this document, we are presenting the deploying the sonarqube stack in AWS using Terraform.

for installing the sonarqube, please follow this link below.

How to - Sonar installation setup steps

AWS ARCHITETURE FOR SONARQUBE STACK:



TERRAFORM code:

provider.tf

```
provider "aws" {
  region     = var.aws_region
  access_key = var.access_key
  secret_key = var.secret_key
}
```

instances.tf

```
resource "aws_instance" "sonar" {
    count = length(var.subnets_cidr)
    ami = var.webservers_ami
    instance_type = var.instance_type
    vpc_security_group_ids = [ "sg-00ec57c74f9cf0cf7" ]
    key_name = "prasanna"
    user_data = file("practiceshel.sh")
    tags ={
      Name = "Sonar-${count.index}"
    }
}
```

elb.tf

```
# Create a new load balancer
resource "aws_elb" "sonarelb" {
  security_groups = [ "sg-00ec57c74f9cf0cf7" ]
  availability_zones = var.azs


  listener     {
      instance_port     = 9000
      instance_protocol = "HTTP"
      lb_port           = 9000
      lb_protocol       = "HTTP"
    }



  health_check  {
    target              = "HTTP:9000/"
    interval            = 30
    healthy_threshold   = 10
    unhealthy_threshold = 2
    timeout             = 20
  }

  instances                   = [for value in aws_instance.sonar: value.
id]
  tags ={
    Name = "sonar-elb"
  }

}

output "elb-dns-name" {
  value = aws_elb.sonarelb.dns_name
}
```

var.tf

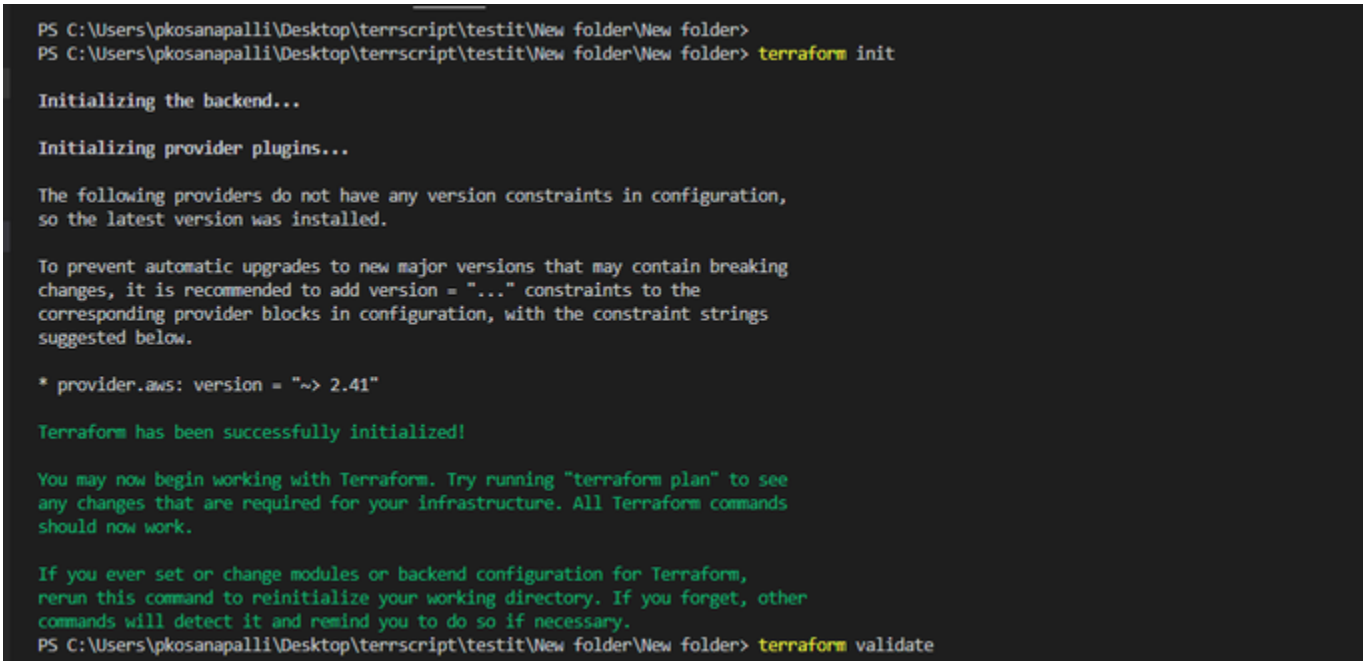
```
variable "aws_region" {
    default = "us-east-1"
}
variable "vpc_cidr" {
    default = "10.20.0.0/16"
}
variable "subnets_cidr" {
    type = list
    default = ["10.20.1.0/28", "10.20.2.0/28"]
}
variable "azs" {
    type = list
    default = ["us-east-1a", "us-east-1b","us-east-1c", "us-east-1d","us
-east-1e", "us-east-1f"]
}
variable "secret_key" {
    type = string
    default = "25r4QLtr56EqfxCVzEk3eHwS1lxhn6bWWEsYH8Ms"
}
variable "access_key" {
    type = string
    default = "AKIAXZXIY4Q4OQX5IE5K"
}
variable "webservers_ami" {
    default = "ami-00eb20669e0990cb4"
}

variable "instance_type" {
    default = "t2.large"
}
```

shell script for installing sonar script:

```
#!/bin/bash

sudo su
sudo yum update -y
sudo wget https://d3pxv6yz143wms.cloudfront.net/11.0.5.10.1/java-11-amaz
on-corretto-devel-11.0.5.10-1.x86_64.rpm
sudo yum install -y java-11-amazon-corretto-devel-11.0.5.10-1.x86_64.rpm

sudo yum update -y
sudo java -version
sudo wget -O /etc/yum.repos.d/sonar.repo http://downloads.sourceforge.ne
t/project/sonar-pkg/rpm/sonar.repo
sudo yum install -y sonar
sudo service sonar restart
```

terraform commands:

1. terraform init.



2. terraform validate - to validate the code format.

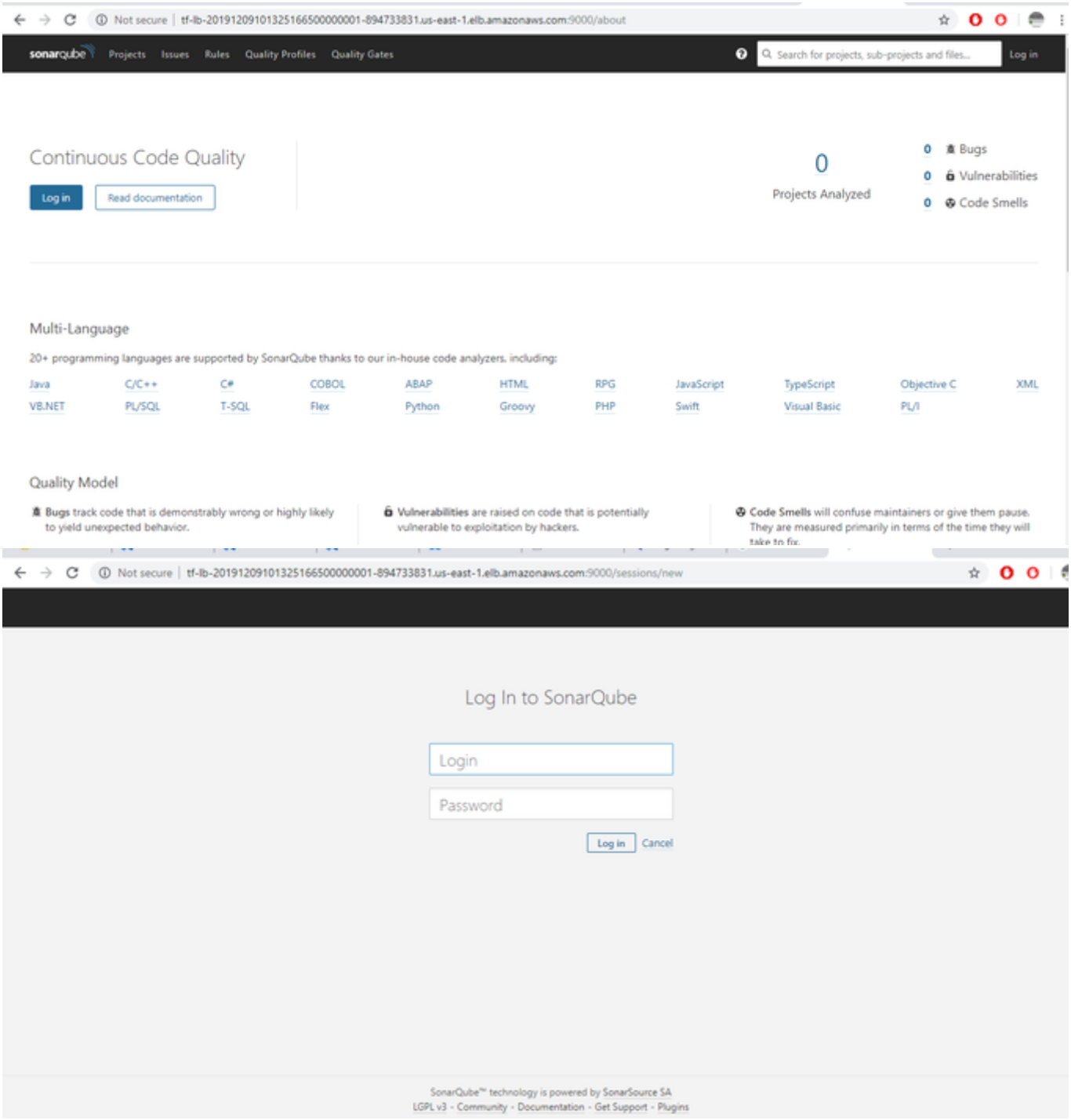
3. terraform plan - to validate the resources in aws.

```
PS C:\Users\pkosanapalli\Desktop\terrscript\testit\New folder\New folder> terraform plan
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.

aws_instance.sonar[1]: Refreshing state... [id=i-08346b0886d49be5c]
aws_instance.sonar[0]: Refreshing state... [id=i-0a036a871029b9cae]
aws_elb.sonarelb: Refreshing state... [id=tf-lb-20191209095255436000000001]

------------------------------------------------------------------------

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_elb.sonarelb will be created
  + resource "aws_elb" "sonarelb" {
      + arn                         = (known after apply)
      + availability_zones          = [
          + "us-east-1a",
          + "us-east-1b",
          + "us-east-1c",
          + "us-east-1d",
          + "us-east-1e",
          + "us-east-1f",
        ]
      + connection_draining         = false
      + connection_draining_timeout = 300
      + cross_zone_load_balancing   = true
      + dns_name                    = (known after apply)
```

4. terraform apply - to apply the change of resources in aws.

```
          + timeout             = 5
          + unhealthy_threshold = 2
        }

      + listener {
          + instance_port     = 9000
          + instance_protocol = "HTTP"
          + lb_port           = 9000
          + lb_protocol       = "HTTP"
        }
    }

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_elb.sonarelb: Creating...
aws_elb.sonarelb: Still creating... [10s elapsed]
aws_elb.sonarelb: Creation complete after 15s [id=tf-lb-20191209100544553100000001]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

Outputs:

elb-dns-name = tf-lb-20191209100544553100000001-486545270.us-east-1.elb.amazonaws.com
```

5. Go to url and try the url with port number. you can see the sonar instance.



6. terraform destroy - to destroy the resource in aws.

```
elb-dns-name = tf-lb-20191209100544553100000001-486545270.us-east-1.elb.amazonaws.com
PS C:\Users\pkosanapalli\Desktop\terrscript\testit\New folder\New folder> terraform destroy
aws_instance.sonar[0]: Refreshing state... [id=i-0a036a871029b9cae]
aws_instance.sonar[1]: Refreshing state... [id=i-08346b0886d49be5c]
aws_elb.sonarelb: Refreshing state... [id=tf-lb-20191209100544553100000001]

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  # aws_elb.sonarelb will be destroyed
  - resource "aws_elb" "sonarelb" {
      - arn                       = "arn:aws:elasticloadbalancing:us-east-1:536285340728:loadbalancer/tf-lb-20191209100544553100000001" -> nu
      - availability_zones        = [
          - "us-east-1a",
          - "us-east-1b",
          - "us-east-1c",
          - "us-east-1d",
          - "us-east-1e",
          - "us-east-1f",
        ] -> null
      - connection_draining       = false -> null
      - connection_draining_timeout = 300 -> null
      - cross_zone_load_balancing = true -> null
      - dns_name                  = "tf-lb-20191209100544553100000001-486545270.us-east-1.elb.amazonaws.com" -> null
```

```
PROBLEMS 22   OUTPUT   DEBUG CONSOLE   TERMINAL                    1: powershell

          - iops                = 100 -> null
          - volume_id           = "vol-0e146f93ddb90c2dc" -> null
          - volume_size         = 8 -> null
          - volume_type         = "gp2" -> null
        }
    }

Plan: 0 to add, 0 to change, 3 to destroy.

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_elb.sonarelb: Destroying... [id=tf-lb-20191209100544553100000001]
aws_elb.sonarelb: Destruction complete after 3s
aws_instance.sonar[1]: Destroying... [id=i-08346b0886d49be5c]
aws_instance.sonar[0]: Destroying... [id=i-0a036a871029b9cae]
aws_instance.sonar[1]: Still destroying... [id=i-08346b0886d49be5c, 10s elapsed]
aws_instance.sonar[0]: Still destroying... [id=i-0a036a871029b9cae, 10s elapsed]
aws_instance.sonar[1]: Still destroying... [id=i-08346b0886d49be5c, 20s elapsed]
aws_instance.sonar[0]: Still destroying... [id=i-0a036a871029b9cae, 20s elapsed]
aws_instance.sonar[1]: Still destroying... [id=i-08346b0886d49be5c, 30s elapsed]
aws_instance.sonar[0]: Still destroying... [id=i-0a036a871029b9cae, 30s elapsed]
aws_instance.sonar[0]: Destruction complete after 35s
aws_instance.sonar[1]: Destruction complete after 35s

Destroy complete! Resources: 3 destroyed.
PS C:\Users\pkosanapalli\Desktop\terrscript\testit\New folder\New folder>
```

7. the resources details can be see in terraform.tfstate file.


```
{
  "version": 4,
  "terraform_version": "0.12.17",
  "serial": 44,
  "lineage": "1bed19af-cb8e-1c7f-b769-d6801f30c0b7",
  "outputs": {
    "elb-dns-name": {
      "value": "tf-lb-20191209101325166500000001-894733831.us-east-1.elb
.amazonaws.com",
      "type": "string"
    }
```

```
  },
  "resources": [
    {
      "mode": "managed",
      "type": "aws_elb",
      "name": "sonarelb",
      "provider": "provider.aws",
      "instances": [
        {
          "schema_version": 0,
          "attributes": {
            "access_logs": [],
            "arn": "arn:aws:elasticloadbalancing:us-east-1:536285340728:
loadbalancer/tf-lb-20191209101325166500000001",
            "availability_zones": [
              "us-east-1a",
              "us-east-1b",
              "us-east-1c",
              "us-east-1d",
              "us-east-1e",
              "us-east-1f"
            ],
            "connection_draining": false,
            "connection_draining_timeout": 300,
            "cross_zone_load_balancing": true,
            "dns_name": "tf-lb-20191209101325166500000001-894733831.us-e
ast-1.elb.amazonaws.com",
            "health_check": [
              {
                "healthy_threshold": 10,
                "interval": 30,
                "target": "HTTP:9000/",
                "timeout": 20,
                "unhealthy_threshold": 2
              }
            ],
            "id": "tf-lb-20191209101325166500000001",
            "idle_timeout": 60,
            "instances": [
              "i-06a0d6738c5e30389",
              "i-0f60cd4820052f3b0"
            ],
            "internal": false,
            "listener": [
              {
                "instance_port": 9000,
                "instance_protocol": "HTTP",
                "lb_port": 9000,
                "lb_protocol": "HTTP",
                "ssl_certificate_id": ""
```

```
              }
            ],
            "name": "tf-lb-20191209101325166500000001",
            "name_prefix": null,
            "security_groups": [
              "sg-00ec57c74f9cf0cf7"
            ],
            "source_security_group": "536285340728/launch-wizard-8",
            "source_security_group_id": "sg-00ec57c74f9cf0cf7",
            "subnets": [
              "subnet-2e331811",
              "subnet-36f3c919",
              "subnet-5a615a07",
              "subnet-5cff5053",
              "subnet-96ecf3f2",
              "subnet-f24589b8"
            ],
            "tags": {
              "Name": "sonar-elb"
            },
            "zone_id": "Z35SXDOTRQ7X7K"
          },
          "private": "bnVsbA==",
          "dependencies": [
            "aws_instance.sonar"
          ]
        }
      ]
    },
    {
      "mode": "managed",
      "type": "aws_instance",
      "name": "sonar",
      "each": "list",
      "provider": "provider.aws",
      "instances": [
        {
          "index_key": 0,
          "schema_version": 1,
          "attributes": {
            "ami": "ami-00eb20669e0990cb4",
            "arn": "arn:aws:ec2:us-east-1:536285340728:instance/i-06a0d6
738c5e30389",
            "associate_public_ip_address": true,
            "availability_zone": "us-east-1c",
            "cpu_core_count": 2,
            "cpu_threads_per_core": 1,
            "credit_specification": [
              {
                "cpu_credits": "standard"
```

```
    }
  ],
  "disable_api_termination": false,
  "ebs_block_device": [],
  "ebs_optimized": false,
  "ephemeral_block_device": [],
  "get_password_data": false,
  "host_id": null,
  "iam_instance_profile": "",
  "id": "i-06a0d6738c5e30389",
  "instance_initiated_shutdown_behavior": null,
  "instance_state": "running",
  "instance_type": "t2.large",
  "ipv6_address_count": 0,
  "ipv6_addresses": [],
  "key_name": "prasanna",
  "monitoring": false,
  "network_interface": [],
  "network_interface_id": null,
  "password_data": "",
  "placement_group": "",
  "primary_network_interface_id": "eni-08a6635f07673e98e",
  "private_dns": "ip-172-31-83-167.ec2.internal",
  "private_ip": "172.31.83.167",
  "public_dns": "ec2-3-87-57-183.compute-1.amazonaws.com",
  "public_ip": "3.87.57.183",
  "root_block_device": [
    {
      "delete_on_termination": true,
      "encrypted": false,
      "iops": 100,
      "kms_key_id": "",
      "volume_id": "vol-0abceebd7bfdc0aae",
      "volume_size": 8,
      "volume_type": "gp2"
    }
  ],
  "security_groups": [
    "launch-wizard-8"
  ],
  "source_dest_check": true,
  "subnet_id": "subnet-36f3c919",
  "tags": {
    "Name": "Sonar-0"
  },
  "tenancy": "default",
  "timeouts": null,
  "user_data": "c5b9683801a654956201b7975a415707302691da",
  "user_data_base64": null,
  "volume_tags": {},
```

```
            "vpc_security_group_ids": [
              "sg-00ec57c74f9cf0cf7"
            ]
          },
          "private": "eyJlMmJmYjczMC1lY2FhLTExZTYtOGY4OC0zNDM2M2JjN2M0YzAiOnsiY3JlYXRlIjo2MDAwMDAwMDAwMDAsImRlbGV0ZSI6MTIwMDAwMDAwMDAwMCwidXBkYXRlIjo2MDAwMDAwMDAwMDB9LCJzY2hlbWFfdmVyc2lvbiI6IjEifQ=="
        },
        {
          "index_key": 1,
          "schema_version": 1,
          "attributes": {
            "ami": "ami-00eb20669e0990cb4",
            "arn": "arn:aws:ec2:us-east-1:536285340728:instance/i-0f60cd4820052f3b0",
            "associate_public_ip_address": true,
            "availability_zone": "us-east-1c",
            "cpu_core_count": 2,
            "cpu_threads_per_core": 1,
            "credit_specification": [
              {
                "cpu_credits": "standard"
              }
            ],
            "disable_api_termination": false,
            "ebs_block_device": [],
            "ebs_optimized": false,
            "ephemeral_block_device": [],
            "get_password_data": false,
            "host_id": null,
            "iam_instance_profile": "",
            "id": "i-0f60cd4820052f3b0",
            "instance_initiated_shutdown_behavior": null,
            "instance_state": "running",
            "instance_type": "t2.large",
            "ipv6_address_count": 0,
            "ipv6_addresses": [],
            "key_name": "prasanna",
            "monitoring": false,
            "network_interface": [],
            "network_interface_id": null,
            "password_data": "",
            "placement_group": "",
            "primary_network_interface_id": "eni-0ee126e958cb6a854",
            "private_dns": "ip-172-31-94-220.ec2.internal",
            "private_ip": "172.31.94.220",
            "public_dns": "ec2-3-86-113-188.compute-1.amazonaws.com",
            "public_ip": "3.86.113.188",
            "root_block_device": [
              {
```

```
                "delete_on_termination": true,
                "encrypted": false,
                "iops": 100,
                "kms_key_id": "",
                "volume_id": "vol-0259608bdd5213d1d",
                "volume_size": 8,
                "volume_type": "gp2"
              }
            ],
            "security_groups": [
              "launch-wizard-8"
            ],
            "source_dest_check": true,
            "subnet_id": "subnet-36f3c919",
            "tags": {
              "Name": "Sonar-1"
            },
            "tenancy": "default",
            "timeouts": null,
            "user_data": "c5b9683801a654956201b7975a415707302691da",
            "user_data_base64": null,
            "volume_tags": {},
            "vpc_security_group_ids": [
              "sg-00ec57c74f9cf0cf7"
            ]
          },
          "private": "eyJlMmJmYjczMC1lY2FhLTExZTYtOGY4OC0zNDM2M2JjN2M0Yz
AiOnsiY3JlYXRlIjo2MDAwMDAwMDAwMDAsImRlbGV0ZSI6MTIwMDAwMDAwMDAwMCwidXBkYX
RlIjo2MDAwMDAwMDAwMDB9LCJzY2hlbWFfdmVyc2lvbiI6IjEifQ=="
        }
      ]
    }
```

```
    ]
  }
```

How to save tfstate file while multiple developers working on the same:

we need to use the s3 as backend to store terraform.tfstate files.

many bankends are there i.e

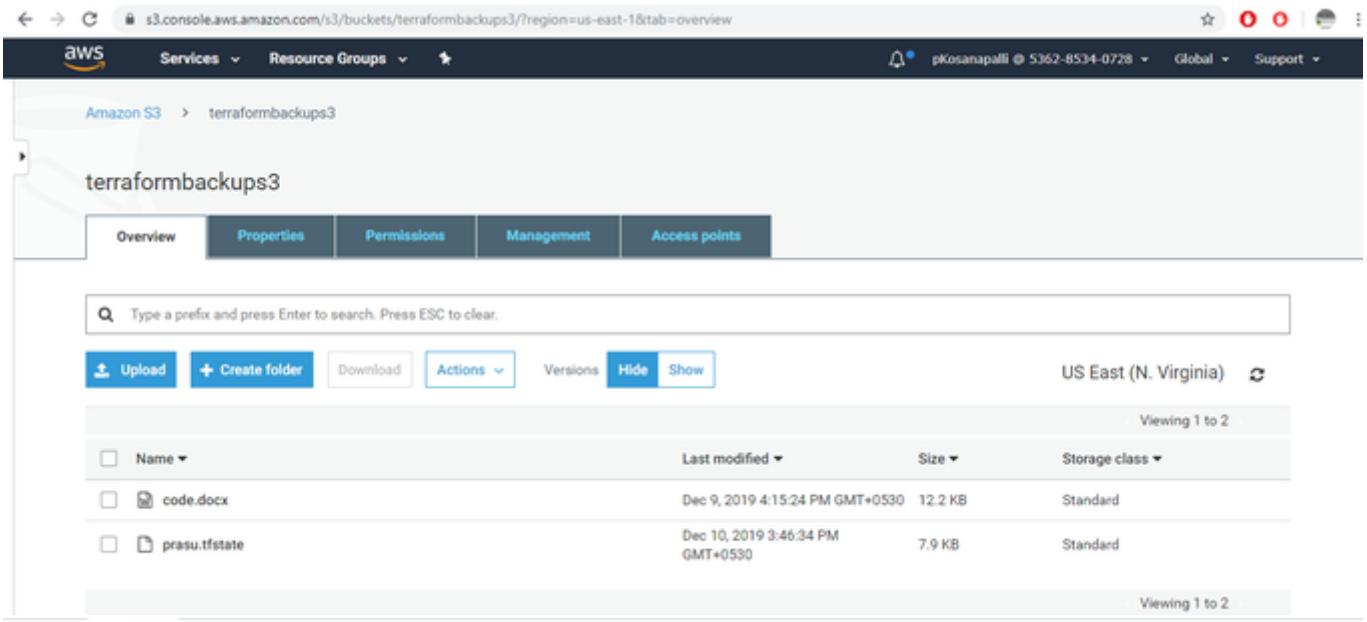
- artifactory
- azurerm
- consul
- etcd
- etcdv3
- gcs
- http
- manta
- oss
- pg
- s3
- swift
- terraform enterprise.

example with s3.

```
terraform {
  backend "s3" {
    bucket         = "terraformbackups3"
    key            = "prasu.tfstate"
    region         = "us-east-1"
    encrypt        = true
    profile        = "default"

  }
}
```

```
#sampleone
terraform {
  backend "artifactory" {
    username = "SheldonCooper"
    password = "AmyFarrahFowler"
    url      = "https://custom.artifactoryonline.com/artifactory"
    repo     = "foo"
    subpath  = "terraform-bar"
  }
}
```

- terraform init
- terraform apply



that's all done !!!