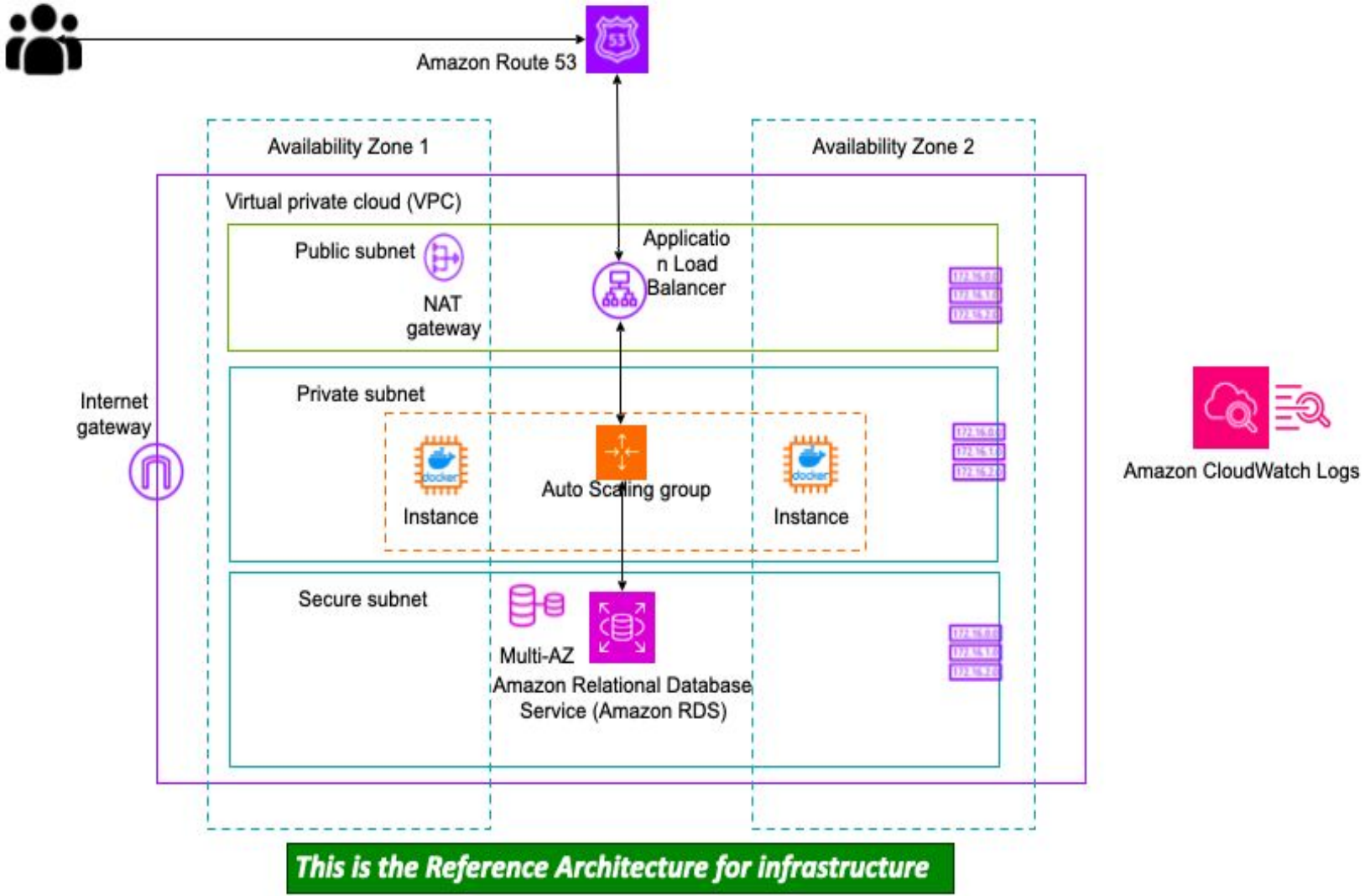
AWS Infrastructure provisioning using terraform

Creation of Infrastructure using Terraform

- Terraform is a mandatory skill. Used across the industry and in multi cloud scenario.
- It’s easy to get started but can get really complicated later to manage.
- Best practices are a MUST.
- State must be in Cloud using S3 or Azure Blob Storage.
- For State file security enable S3 cross region replication, versioning and encryption
- Apply state locking using DynamoDB
- The CICD Pipeline to have three steps and checks:
  - **On commit to any Branch** perform terraform init, validate, fmt and plan operations, *checkov scan is optional*
  - **On pull request creation** — perform terraform init, validate fmt and plan so that reviewer can see what is the change this pipeline will perform
  - **On merge** the feature is merged to master/ main and terraform apply happens using GitOps approach.

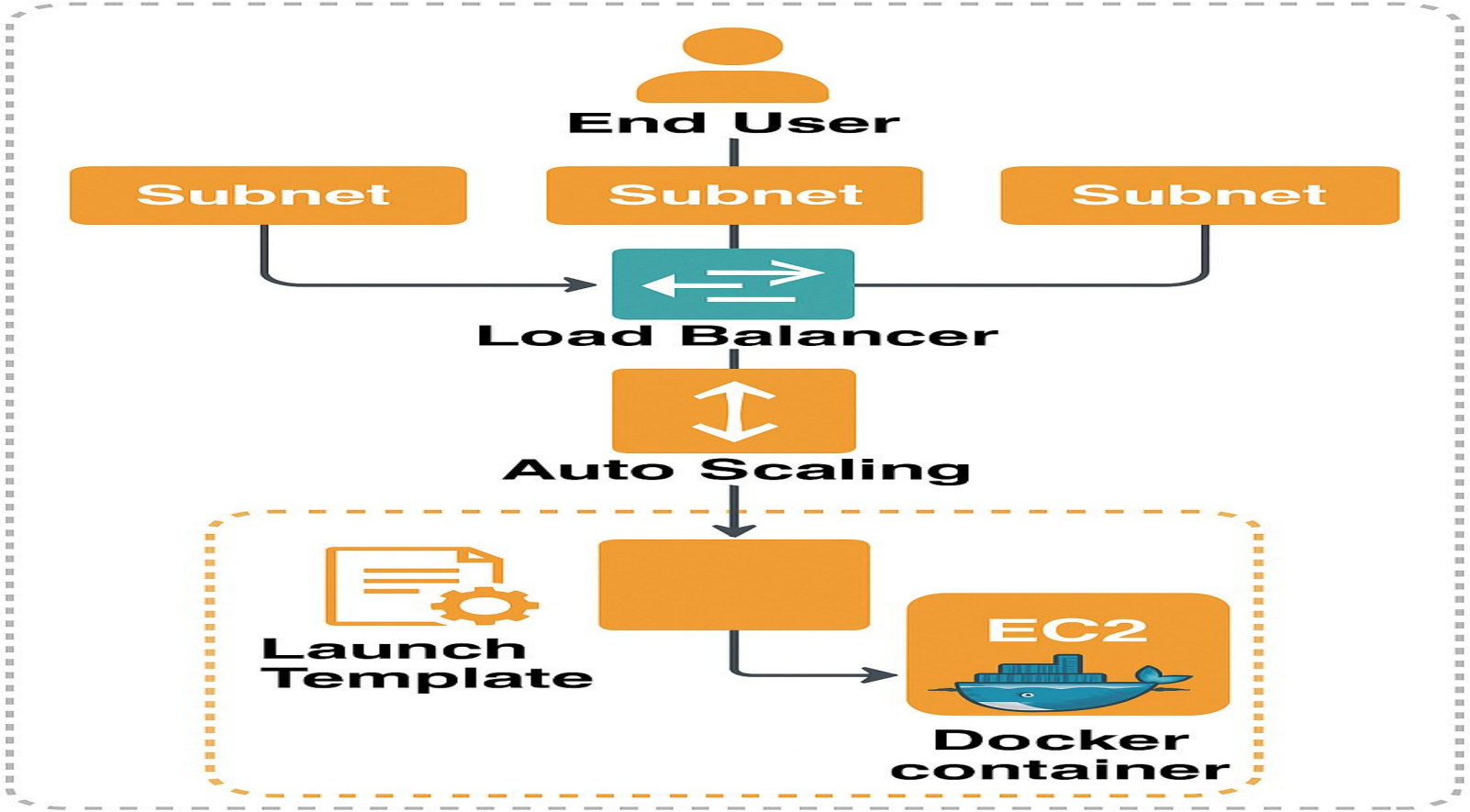

Amazon Route 53
Availability Zone 1
Availability Zone 2
Virtual private cloud (VPC)
Public subnet
NAT gateway
Applicatio n Load Balancer
Private subnet
Internet gateway
Instance
Auto Scaling group
Instance
Secure subnet
Multi-AZ
Amazon Relational Database Service (Amazon RDS)
Amazon CloudWatch Logs
This is the Reference Architecture for infrastructure

Key requirements for Infrastructure

1- 3 Subnets ArchitecturePublic, Private and Secure Subnets.
Public Subnets should have routes to Internet Gateway. Private Subnets should have route to NAT Gateway. Secure subnets should not have route to IGW or NAT GW.

2- Load balancer in Public SubnetCreate Load Balancer in Public Subnet across 2 AZs. Create corresponding Listener and Target Groups.

3- Create Auto Scaling GroupCreate the ASG across 2 AZs in private subnet. Attach the ASG to ALB. Desired=1, Min=1, Max=1
The instances should not have public IPs.
Instances should be connecting using SSM or EC2 Instance Connect Endpoint.
Security group should not open port 22.
EC2 should be using a user data script at startup and install the application [Docker Image or WAR JAR File]. This should be in sync with the ALB Target groups.


End User
Subnet
Subnet
Subnet
Load Balancer
Auto Scaling
Launch
Template
EC2
Docker
container

Step1: clone/fork this repository for infrastructure provisioning

1- https://github.com/ankit20000/springapp_terraform.git

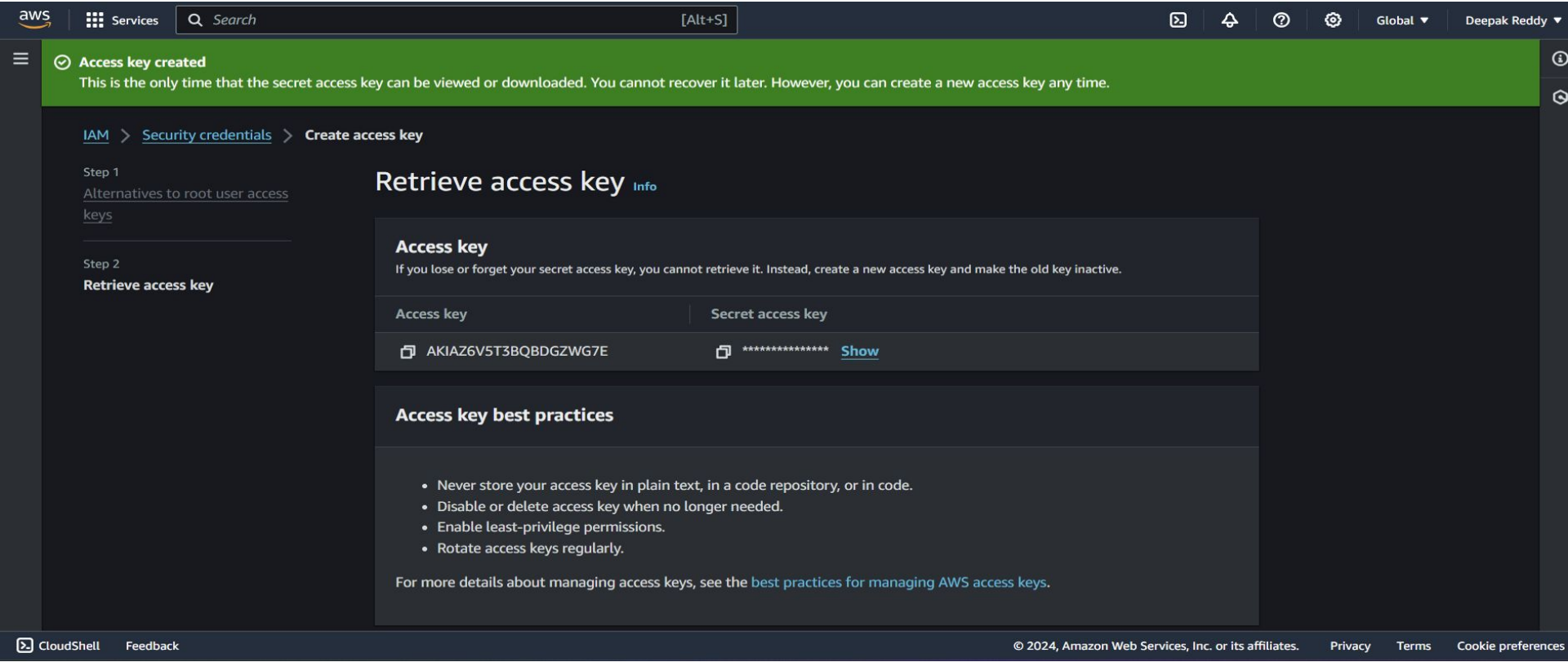
2- Install terraform –

https://developer.hashicorp.com/terraform/install

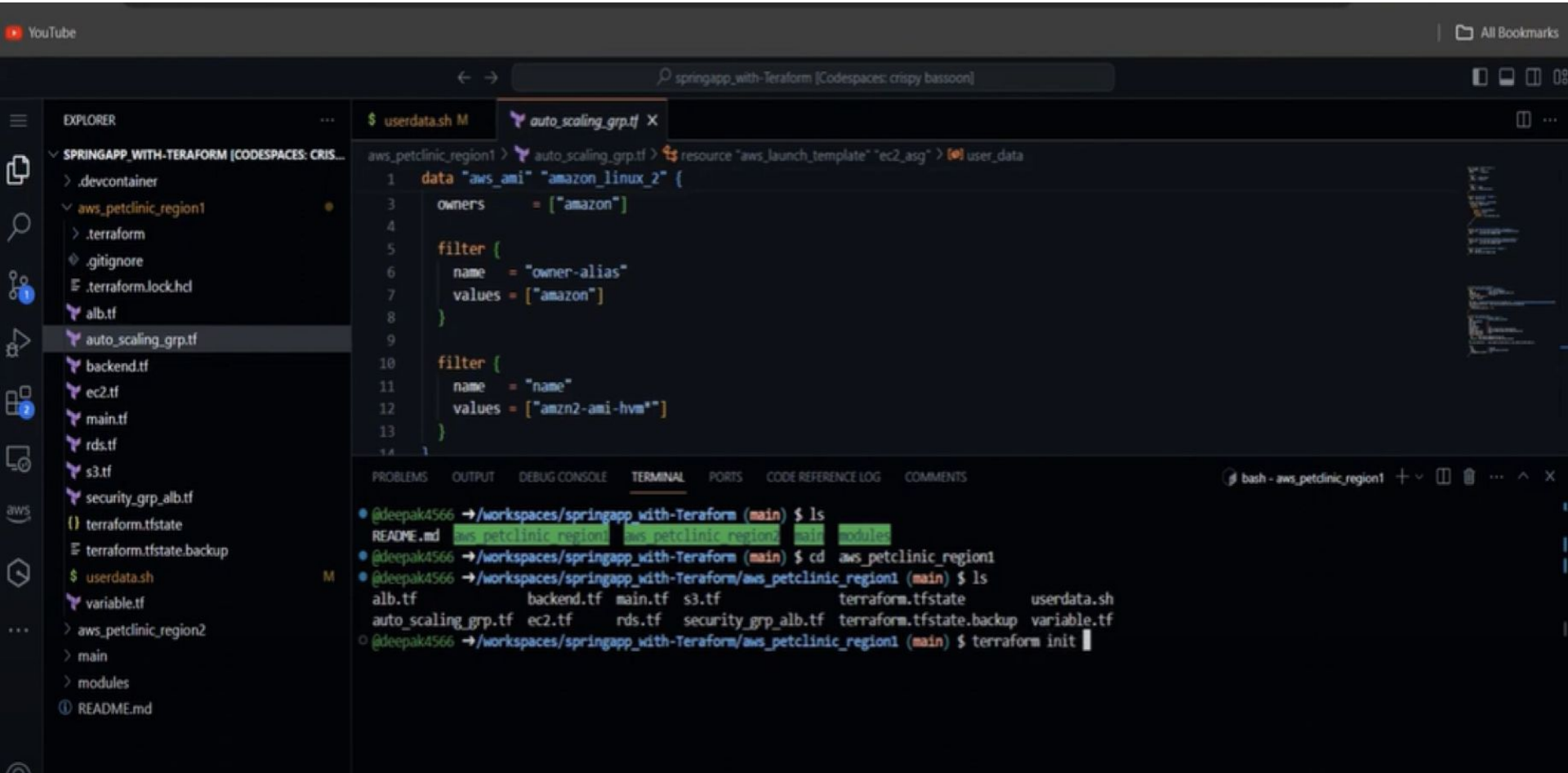
3- Install aws cli –

https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html

Step2 :configure aws cli with access keys



Step3 : apply command `Terraform init` to initialize terraform backend



YouTube

springapp_with-Teraform [Codespaces: crispy bassoon]

EXPLORER

SPRINGAPP_WITH-TERAFORM [CODESPACES: CRIS...

- .devcontainer
- aws_petclinic_region1
  - .terraform
  - .gitignore
  - .terraform.lock.hcl
  - alb.tf
  - auto_scaling_grp.tf
  - backend.tf
  - ec2.tf
  - main.tf
  - rds.tf
  - s3.tf
  - security_grp_alb.tf
  - terraform.tfstate
  - terraform.tfstate.backup
  - userdata.sh M
  - variable.tf
- aws_petclinic_region2
- main
- modules
- README.md

$ userdata.sh M | auto_scaling_grp.tf

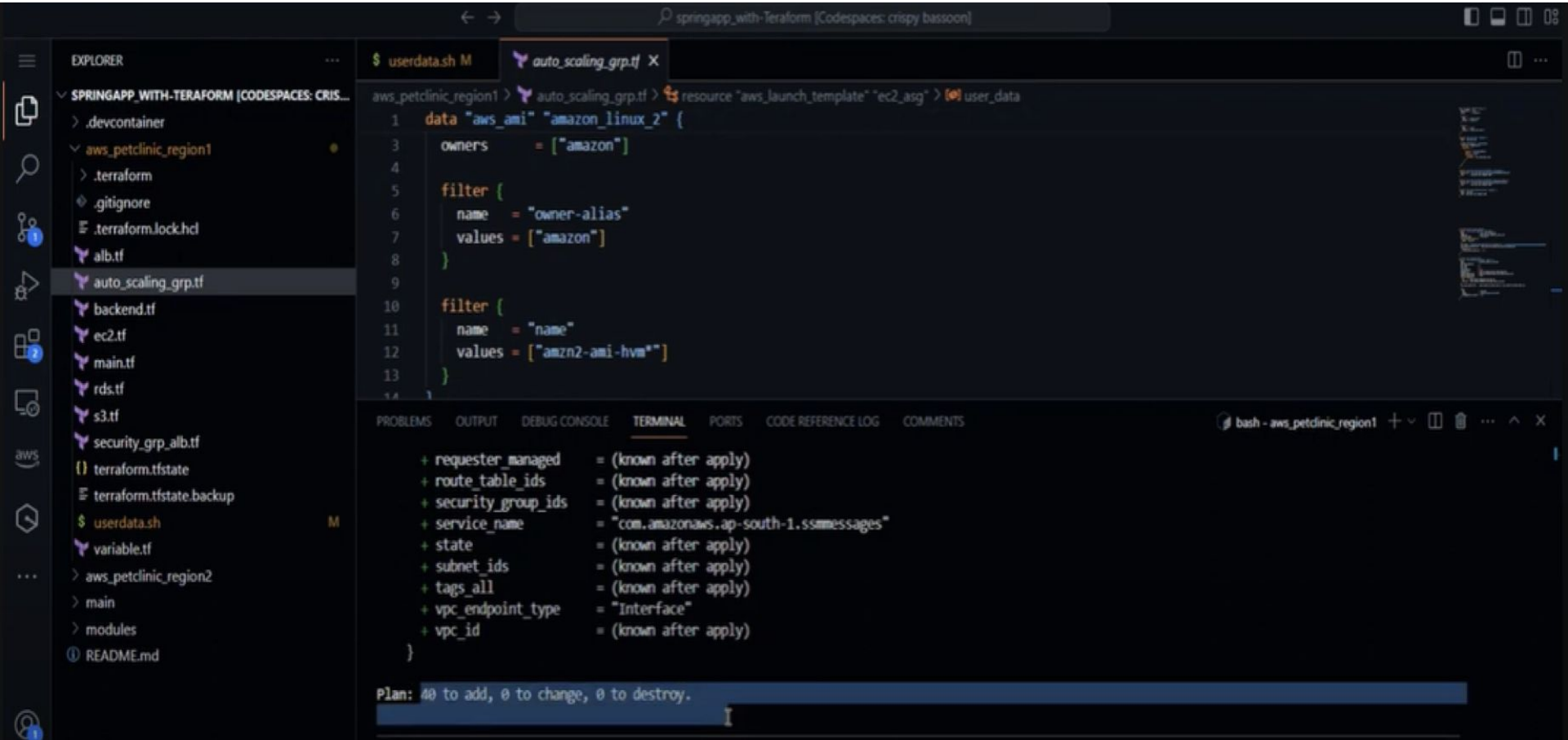
aws_petclinic_region1 > auto_scaling_grp.tf > resource "aws_launch_template" "ec2_asg" > user_data

```
data "aws_ami" "amazon_linux_2" {
  owners      = ["amazon"]

  filter {
    name   = "owner-alias"
    values = ["amazon"]
  }

  filter {
    name   = "name"
    values = ["amzn2-ami-hvm*"]
  }
}
```

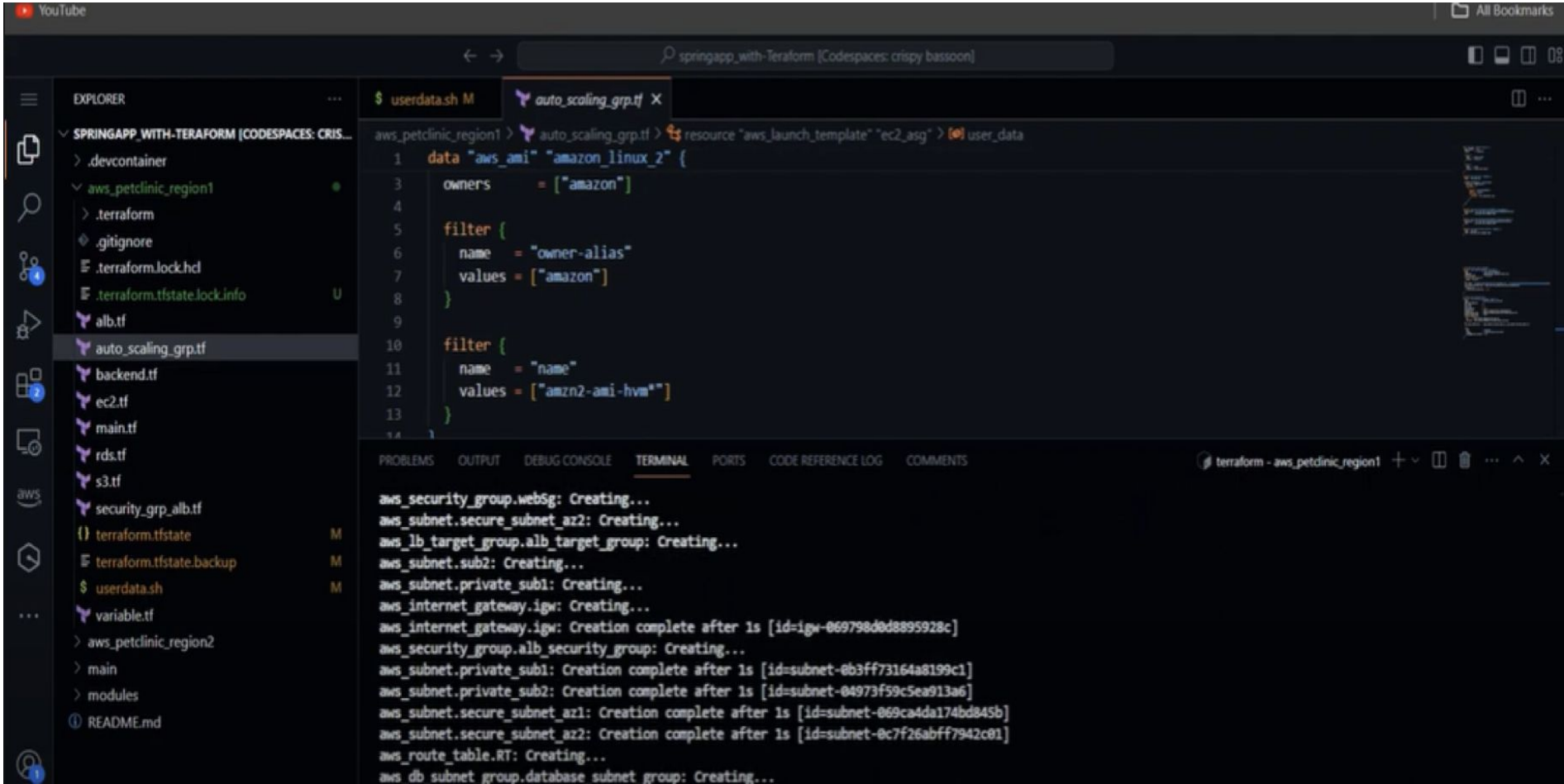
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS CODE REFERENCE LOG COMMENTS

```
Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.41.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
```

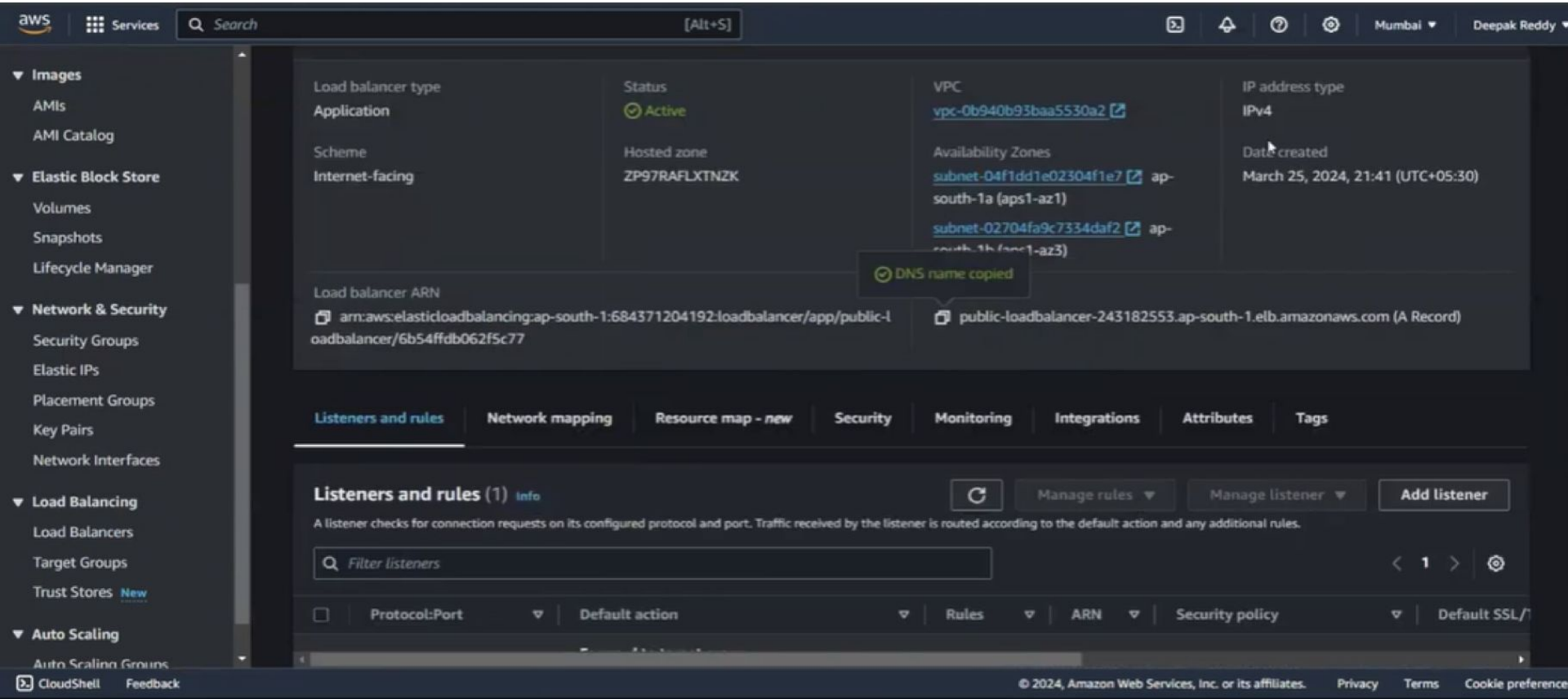
Step4 : apply command `Terraform plan` to check what resources are created



Step5 : Terraform apply –auto approve` to create infra



Step6: after creation, go load balancers in aws console and check whether application is running or not with dns name.



We can successfully see that application is running