**1. Use Terraform to deploy the following things:**

- VPC

- Public Subnets

- Private Subnets

- Launch Template

- Auto Scaling Group of EC2 Machine with the Latest ubuntu image

- Application Load Balancer for the above auto-scaling group

- MySQL RDS

- SNS Topic and subscription

- Cloudwatch CPU and RAM utilization metric and Alert (Alerts should be sent to SNS Topic subscribers)

**Ans:-**

provider "aws" {

region = "ap-south-1"

access\_key = "AKIAWL3SPHGY5PCPQNS4"

secret\_key = "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"

}

**# create vpc**

resource "aws\_vpc" "cloud" {

cidr\_block = "10.0.0.0/16"

instance\_tenancy = "default"

tags = {

Name = "cloudvpc"

}

}

**# create subnets**

resource "aws\_subnet" "public-subnet" {

vpc\_id = aws\_vpc.cloud.id

cidr\_block = "10.0.1.0/24"

tags = {

Name = "public-subnet"

}

}

resource "aws\_subnet" "private-subnet" {

vpc\_id = aws\_vpc.cloud.id

cidr\_block = "10.0.2.0/24"

tags = {

Name = "private-subnet"

}

}

**# create security group**

resource "aws\_security\_group" "cloudsg" {

name = "cloudsg"

description = "Allow TLS inbound traffic"

vpc\_id = aws\_vpc.cloud.id

ingress {

description = "TLS from VPC"

from\_port = 22

to\_port = 22

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

egress {

from\_port = 0

to\_port = 0

protocol = "-1"

cidr\_blocks = ["0.0.0.0/0"]

}

tags = {

Name = "cloud-sg"

}

}

**# create internet gateway**

resource "aws\_internet\_gateway" "cloud-igw" {

vpc\_id = aws\_vpc.cloud.id

tags = {

Name = "cloud-igw"

}

}

**# create route table**

resource "aws\_route\_table" "public-rt" {

vpc\_id = aws\_vpc.cloud.id

route {

cidr\_block = "0.0.0.0/0"

gateway\_id = aws\_internet\_gateway.cloud-igw.id

}

tags = {

Name = "public-rt"

}

}

resource "aws\_route\_table" "private-rt" {

vpc\_id = aws\_vpc.cloud.id

route {

cidr\_block = "0.0.0.0/0"

gateway\_id = aws\_nat\_gateway.cloud-nat.id

}

tags = {

Name = "private-rt"

}

}

**# create route table association**

resource "aws\_route\_table\_association" "private-asso" {

subnet\_id = aws\_subnet.private-subnet.id

route\_table\_id = aws\_route\_table.private-rt.id

}

resource "aws\_route\_table\_association" "public-asso" {

subnet\_id = aws\_subnet.public-subnet.id

route\_table\_id = aws\_route\_table.public-rt.id

}

**# create key-pair**

resource "aws\_key\_pair" "yudizkey" {

key\_name = "yudizkey"

public\_key = "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC8SAK+dcmSdu0y9/V0MSW+XWttLf2ok5nge8Zhg+zu0CtgKL8XNjxHjmc0qixlix7QGKhw7gxNIE4Sti6ybK13H/2CRU8ZnITAZ185t0PC3xMtnyNgxLpW72KB8Z1unSUcIPvLP50NdRG0Ksm/b0bXDqv5JEctvqYda7JaZT7/AVBJH9u8Ftbxa5T1FvbWg2nfxVG9u1S8dRc8pukVNLtyvHj+G/D8Fa+ld/B1Q8m2cMnwA1Xs0F2u1rOeUPcl/hIGArlP49K0FmOBxhFA9MfAzwjGoPmIQSah2dvxQWPg3Wl9BWNw1iQTY2bDhksGD36rlnWDHE0CqOZC7XgI6fRj root@ip-172-31-12-31.ap-south-1.compute.internal"

}

**#create instance**

resource "aws\_instance" "instance" {

ami = "ami-01a4f99c4ac11b03c"

instance\_type = "t3.micro"

subnet\_id = aws\_subnet.public-subnet.id

vpc\_security\_group\_ids = [aws\_security\_group.cloudsg.id]

key\_name = "yudizkey"

tags = {

Name = "instance"

}

}

resource "aws\_instance" "db-instance" {

ami = "ami-01a4f99c4ac11b03c"

instance\_type = "t3.micro"

subnet\_id = aws\_subnet.private-subnet.id

vpc\_security\_group\_ids = [aws\_security\_group.cloudsg.id]

key\_name = "yudizkey"

tags = {

Name = "db-instance"

}

}

resource "aws\_eip" "cloud-natip" {

vpc = true

}

resource "aws\_nat\_gateway" "cloud-nat" {

allocation\_id = aws\_eip.cloud-natip.id

subnet\_id = aws\_subnet.public-subnet.id

}

resource "aws\_security\_group" "elb" {

name = "teraform-example-elb"

egress {

from\_port = 0

to\_port = 0

protocol = "-1"

cidr\_blocks = ["0.0.0.0/0"]

}

ingress {

from\_port = 80

to\_port = 80

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

}

**# Create a Launch Configuration**

resource "aws\_launch\_configuration" "ubuntu" {

image\_id = "ami-0f8ca728008ff5af4"

instance\_type = "t2.micro"

key\_name = "yudizkey"

security\_groups = ["sg-03726efbae8345350"]

}

**# Create an Auto Scaling Group**

resource "aws\_autoscaling\_group" "ubuntu" {

name = "ubuntu"

launch\_configuration = aws\_launch\_configuration.ubuntu.id

vpc\_zone\_identifier = ["subnet-00ba1f77c313f378a","subnet-0eda9fa948d0e80fa"]

min\_size = 1

max\_size = 3

desired\_capacity = 2

}

# Create a scaling policy

resource "aws\_autoscaling\_policy" "my-cpu-scaling-policy" {

name = "my-cpu-scaling-policy"

adjustment\_type = "ChangeInCapacity"

scaling\_adjustment = 1

cooldown = 300

autoscaling\_group\_name = aws\_autoscaling\_group.ubuntu.name

}

**#create MySQL RDS**

terraform {

required\_providers {

aws = {

source = "hashicorp/aws"

version = "~> 3.27"

}

}

required\_version = ">= 0.14.9"

}

resource "aws\_db\_instance" "rds\_instance" {

allocated\_storage = 20

identifier = "rds-terraform"

storage\_type = "gp2"

engine = "mysql"

engine\_version = "8.0.27"

instance\_class = "db.t2.micro"

name = "mysqldb"

username = "admin"

password = "admin123"

publicly\_accessible = true

skip\_final\_snapshot = true

tags = {

Name = "ExampleRDSServerInstance"

}

}

**# create a target group**

resource "aws\_lb\_target\_group" "target\_group" {

health\_check {

interval = 10

path = "/"

protocol = "HTTP"

timeout = 5

healthy\_threshold = 5

unhealthy\_threshold = 2

}

name = "whiz-tg"

port = 80

protocol = "HTTP"

target\_type = "instance"

vpc\_id = aws\_vpc.cloud.id

}

**#create cloud watch**

resource "aws\_cloudwatch\_dashboard" "EC2-Dashboard" {

dashboard\_name = "EC2-Dashboard"

dashboard\_body = <<EOF

{

"widgets": [

{

"type": "explorer",

"width": 24,

"height": 15,

"x": 0,

"y": 0,

"properties": {

"metrics": [

{

"metricName": "CPUUtilization",

"resourceType": "AWS::EC2::Instance",

"stat": "Maximum"

}

],

"aggregateBy": {

"key": "InstanceType",

"func": "MAX"

},

"labels": [

{

"key": "State",

"value": "running"

}

],

"widgetOptions": {

"legend": {

"position": "bottom"

},

"view": "timeSeries",

"rowsPerPage": 8,

"widgetsPerRow": 2

},

"period": 60,

"title": "Running EC2 Instances CPUUtilization"

}

}

]

}

EOF

}

**# create cloudwatch alarm**

resource "aws\_cloudwatch\_composite\_alarm" "EC2" {

alarm\_description = "Composite alarm that monitors CPUUtilization "

alarm\_name = "EC2\_Composite\_Alarm"

alarm\_actions = [aws\_sns\_topic.EC2\_topic.arn]

alarm\_rule = "ALARM(${aws\_cloudwatch\_metric\_alarm.EC2\_CPU\_Usage\_Alarm.alarm\_name})"

depends\_on = [

aws\_cloudwatch\_metric\_alarm.EC2\_CPU\_Usage\_Alarm,

aws\_sns\_topic.EC2\_topic,

aws\_sns\_topic\_subscription.EC2\_Subscription

]

}

**# Creating the AWS CLoudwatch Alarm that will autoscale the AWS EC2 instance based on CPU utilization.**

resource "aws\_cloudwatch\_metric\_alarm" "EC2\_CPU\_Usage\_Alarm" {

alarm\_name = "EC2\_CPU\_Usage\_Alarm"

comparison\_operator = "GreaterThanOrEqualToThreshold"

evaluation\_periods = "2"

metric\_name = "CPUUtilization"

namespace = "AWS/EC2"

period = "60"

statistic = "Average"

threshold = "70"

alarm\_description = "This metric monitors ec2 cpu utilization exceeding 70%"

}

**# create a cloud watch log group**

resource "aws\_cloudwatch\_log\_group" "ebs\_log\_group" {

name = "ebs\_log\_group"

retention\_in\_days = 30

}

resource "aws\_cloudwatch\_log\_stream" "ebs\_log\_stream" {

name = "ebs\_log\_stream"

log\_group\_name = aws\_cloudwatch\_log\_group.ebs\_log\_group.name

}

**# create sns topic and subscription**

resource "aws\_sns\_topic" "EC2\_topic" {

name = "EC2\_topic"

}

resource "aws\_sns\_topic\_subscription" "EC2\_Subscription" {

topic\_arn = aws\_sns\_topic.EC2\_topic.arn

protocol = "email"

endpoint = "prashantdhole7620@gmail.com"

depends\_on = [

aws\_sns\_topic.EC2\_topic

]

}