**Three Tire App hosting using Terraform**

**Prerequisites**

* **Install**[**Terraform**](https://learn.hashicorp.com/tutorials/terraform/install-cli)
* **Install the**[**AWS CLI**](https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2.html)

**Install Apache**

|  |
| --- |
|  |
|  | yum update -y  yum install -y httpd |
|  | systemctl start httpd service |
|  | systemctl enable httpd service  For Ubuntu  sudo apt-get update -y  sudo apt install -y httpd  sudo systemctl start httpd service  sudo systemctl enable httpd service |
|  |  |

* **Configure the Providers**
* **Create VPC and Subnets**
* **Create Internet Gateway and RT**
* **Create Web Server in Availability Zone**
* **Create Security Groups**
* **Create an Application Load balancer**
* **Create RDS(DB) instance**
* **Create an ECS Cluster**
* **Create code pipeline CI/CD**

|  |
| --- |
| terraform { |
|  | required\_providers { |
|  | aws = { |
|  | source = "hashicorp/aws" |
|  | version = "~> 3.0" |
|  | } |
|  | } |
|  | } |
|  |  |
|  | # Configure the AWS Provider |
|  | provider "aws" { |
|  | region = "us-east-1" |
|  | } |
|  |  |
|  | # Create a VPC |
|  | resource "aws\_vpc" "my-vpc" { |
|  | cidr\_block = "10.0.0.0/16" |
|  | tags = { |
|  | Name = "Demo VPC" |
|  | } |
|  | } |
|  |  |
|  | # Create Web Public Subnet |
|  | resource "aws\_subnet" "web-subnet-1" { |
|  | vpc\_id = aws\_vpc.my-vpc.id |
|  | cidr\_block = "10.0.1.0/24" |
|  | availability\_zone = "us-east-1a" |
|  | map\_public\_ip\_on\_launch = true |
|  |  |
|  | tags = { |
|  | Name = "Web-1a" |
|  | } |
|  | } |
|  |  |
|  | resource "aws\_subnet" "web-subnet-2" { |
|  | vpc\_id = aws\_vpc.my-vpc.id |
|  | cidr\_block = "10.0.2.0/24" |
|  | availability\_zone = "us-east-1b" |
|  | map\_public\_ip\_on\_launch = true |
|  |  |
|  | tags = { |
|  | Name = "Web-2b" |
|  | } |
|  | } |
|  |  |
|  | # Create Application Public Subnet |
|  | resource "aws\_subnet" "application-subnet-1" { |
|  | vpc\_id = aws\_vpc.my-vpc.id |
|  | cidr\_block = "10.0.11.0/24" |
|  | availability\_zone = "us-east-1a" |
|  | map\_public\_ip\_on\_launch = false |
|  |  |
|  | tags = { |
|  | Name = "Application-1a" |
|  | } |
|  | } |
|  |  |
|  | resource "aws\_subnet" "application-subnet-2" { |
|  | vpc\_id = aws\_vpc.my-vpc.id |
|  | cidr\_block = "10.0.12.0/24" |
|  | availability\_zone = "us-east-1b" |
|  | map\_public\_ip\_on\_launch = false |
|  |  |
|  | tags = { |
|  | Name = "Application-2b" |
|  | } |
|  | } |
|  |  |
|  | # Create Database Private Subnet |
|  | resource "aws\_subnet" "database-subnet-1" { |
|  | vpc\_id = aws\_vpc.my-vpc.id |
|  | cidr\_block = "10.0.21.0/24" |
|  | availability\_zone = "us-east-1a" |
|  |  |
|  | tags = { |
|  | Name = "Database-1a" |
|  | } |
|  | } |
|  |  |
|  | resource "aws\_subnet" "database-subnet-2" { |
|  | vpc\_id = aws\_vpc.my-vpc.id |
|  | cidr\_block = "10.0.22.0/24" |
|  | availability\_zone = "us-east-1b" |
|  |  |
|  | tags = { |
|  | Name = "Database-2b" |
|  | } |
|  | } |
|  |  |
|  | resource "aws\_subnet" "database-subnet" { |
|  | vpc\_id = aws\_vpc.my-vpc.id |
|  | cidr\_block = "10.0.3.0/24" |
|  | availability\_zone = "us-east-1a" |
|  |  |
|  | tags = { |
|  | Name = "Database" |
|  | } |
|  | } |
|  |  |
|  | # Create Internet Gateway |
|  | resource "aws\_internet\_gateway" "igw" { |
|  | vpc\_id = aws\_vpc.my-vpc.id |
|  |  |
|  | tags = { |
|  | Name = "Demo IGW" |
|  | } |
|  | } |
|  |  |
|  | # Create Web layer route table |
|  | resource "aws\_route\_table" "web-rt" { |
|  | vpc\_id = aws\_vpc.my-vpc.id |
|  |  |
|  |  |
|  | route { |
|  | cidr\_block = "0.0.0.0/0" |
|  | gateway\_id = aws\_internet\_gateway.igw.id |
|  | } |
|  |  |
|  | tags = { |
|  | Name = "WebRT" |
|  | } |
|  | } |
|  |  |
|  | # Create Web Subnet association with Web route table |
|  | resource "aws\_route\_table\_association" "a" { |
|  | subnet\_id = aws\_subnet.web-subnet-1.id |
|  | route\_table\_id = aws\_route\_table.web-rt.id |
|  | } |
|  |  |
|  | resource "aws\_route\_table\_association" "b" { |
|  | subnet\_id = aws\_subnet.web-subnet-2.id |
|  | route\_table\_id = aws\_route\_table.web-rt.id |
|  | } |
|  |  |
|  | #Create EC2 Instance |
|  | resource "aws\_instance" "webserver1" { |
|  | ami = "ami-0d5eff06f840b45e9" |
|  | instance\_type = "t2.micro" |
|  | availability\_zone = "us-east-1a" |
|  | vpc\_security\_group\_ids = [aws\_security\_group.webserver-sg.id] |
|  | subnet\_id = aws\_subnet.web-subnet-1.id |
|  | user\_data = file("install\_apache.sh") |
|  |  |
|  | tags = { |
|  | Name = "Web Server" |
|  | } |
|  |  |
|  | } |
|  |  |
|  | resource "aws\_instance" "webserver2" { |
|  | ami = "ami-0d5eff06f840b45e9" |
|  | instance\_type = "t2.micro" |
|  | availability\_zone = "us-east-1b" |
|  | vpc\_security\_group\_ids = [aws\_security\_group.webserver-sg.id] |
|  | subnet\_id = aws\_subnet.web-subnet-2.id |
|  | user\_data = file("install\_apache.sh") |
|  |  |
|  | tags = { |
|  | Name = "Web Server" |
|  | } |
|  |  |
|  | } |
|  |  |
|  | # Create Web Security Group |
|  | resource "aws\_security\_group" "web-sg" { |
|  | name = "Web-SG" |
|  | description = "Allow HTTP inbound traffic" |
|  | vpc\_id = aws\_vpc.my-vpc.id |
|  |  |
|  | ingress { |
|  | description = "HTTP from VPC" |
|  | from\_port = 80 |
|  | to\_port = 80 |
|  | 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 = "Web-SG" |
|  | } |
|  | } |
|  |  |
|  | # Create Application Security Group |
|  | resource "aws\_security\_group" "webserver-sg" { |
|  | name = "Webserver-SG" |
|  | description = "Allow inbound traffic from ALB" |
|  | vpc\_id = aws\_vpc.my-vpc.id |
|  |  |
|  | ingress { |
|  | description = "Allow traffic from web layer" |
|  | from\_port = 80 |
|  | to\_port = 80 |
|  | protocol = "tcp" |
|  | security\_groups = [aws\_security\_group.web-sg.id] |
|  | } |
|  |  |
|  | egress { |
|  | from\_port = 0 |
|  | to\_port = 0 |
|  | protocol = "-1" |
|  | cidr\_blocks = ["0.0.0.0/0"] |
|  | } |
|  |  |
|  | tags = { |
|  | Name = "Webserver-SG" |
|  | } |
|  | } |
|  |  |
|  | # Create Database Security Group |
|  | resource "aws\_security\_group" "database-sg" { |
|  | name = "Database-SG" |
|  | description = "Allow inbound traffic from application layer" |
|  | vpc\_id = aws\_vpc.my-vpc.id |
|  |  |
|  | ingress { |
|  | description = "Allow traffic from application layer" |
|  | from\_port = 3306 |
|  | to\_port = 3306 |
|  | protocol = "tcp" |
|  | security\_groups = [aws\_security\_group.webserver-sg.id] |
|  | } |
|  |  |
|  | egress { |
|  | from\_port = 32768 |
|  | to\_port = 65535 |
|  | protocol = "tcp" |
|  | cidr\_blocks = ["0.0.0.0/0"] |
|  | } |
|  |  |
|  | tags = { |
|  | Name = "Database-SG" |
|  | } |
|  | } |
|  |  |
|  | resource "aws\_lb" "external-elb" { |
|  | name = "External-LB" |
|  | internal = false |
|  | load\_balancer\_type = "application" |
|  | security\_groups = [aws\_security\_group.web-sg.id] |
|  | subnets = [aws\_subnet.web-subnet-1.id, aws\_subnet.web-subnet-2.id] |
|  | } |
|  |  |
|  | resource "aws\_lb\_target\_group" "external-elb" { |
|  | name = "ALB-TG" |
|  | port = 80 |
|  | protocol = "HTTP" |
|  | vpc\_id = aws\_vpc.my-vpc.id |
|  | } |
|  |  |
|  | resource "aws\_lb\_target\_group\_attachment" "external-elb1" { |
|  | target\_group\_arn = aws\_lb\_target\_group.external-elb.arn |
|  | target\_id = aws\_instance.webserver1.id |
|  | port = 80 |
|  |  |
|  | depends\_on = [ |
|  | aws\_instance.webserver1, |
|  | ] |
|  | } |
|  |  |
|  | resource "aws\_lb\_target\_group\_attachment" "external-elb2" { |
|  | target\_group\_arn = aws\_lb\_target\_group.external-elb.arn |
|  | target\_id = aws\_instance.webserver2.id |
|  | port = 80 |
|  |  |
|  | depends\_on = [ |
|  | aws\_instance.webserver2, |
|  | ] |
|  | } |
|  |  |
|  | resource "aws\_lb\_listener" "external-elb" { |
|  | load\_balancer\_arn = aws\_lb.external-elb.arn |
|  | port = "80" |
|  | protocol = "HTTP" |
|  |  |
|  | default\_action { |
|  | type = "forward" |
|  | target\_group\_arn = aws\_lb\_target\_group.external-elb.arn |
|  | } |
|  | } |
|  |  |
|  | resource "aws\_db\_instance" "default" { |
|  | allocated\_storage = 10 |
|  | db\_subnet\_group\_name = aws\_db\_subnet\_group.default.id |
|  | engine = "mysql" |
|  | engine\_version = "8.0.20" |
|  | instance\_class = "db.t2.micro" |
|  | multi\_az = true |
|  | name = "mydb" |
|  | username = "username" |
|  | password = "password" |
|  | skip\_final\_snapshot = true |
|  | vpc\_security\_group\_ids = [aws\_security\_group.database-sg.id] |
|  | } |
|  |  |
|  | resource "aws\_db\_subnet\_group" "default" { |
|  | name = "main" |
|  | subnet\_ids = [aws\_subnet.database-subnet-1.id, aws\_subnet.database-subnet-2.id] |
|  |  |
|  | tags = { |
|  | Name = "My DB subnet group" |
|  | } |
|  | } |
|  |  |
|  | output "lb\_dns\_name" { |
|  | description = "The DNS name of the load balancer" |
|  | value = aws\_lb.external-elb.dns\_name |
|  | } |

resource "aws\_ecs\_service" "webapp" {

name = "webappb"

cluster = aws\_ecs\_cluster.foo.id

task\_definition = aws\_ecs\_task\_definition.webapp.arn

desired\_count = 3

iam\_role = aws\_iam\_role.foo.arn

depends\_on = [aws\_iam\_role\_policy.foo]

ordered\_placement\_strategy {

type = "binpack"

field = "cpu"

}

placement\_constraints {

type = "memberOf"

expression = "attribute:ecs.availability-zone in [us-east-2, us-east-2]"

}

}

resource "aws\_codepipeline" "codepipeline" {

name = "tf-test-pipeline"

role\_arn = aws\_iam\_role.codepipeline\_role.arn

artifact\_store {

location = aws\_s3\_bucket.codepipeline\_bucket.bucket

type = "S3"

}

stage {

name = "Source"

action {

name = "Source"

category = "Source"

owner = "AWS"

provider = "CodeStarSourceConnection"

version = "1"

output\_artifacts = ["source\_output"]

configuration = {

ConnectionArn = aws\_codestarconnections\_connection.example.arn

FullRepositoryId = "my-organization/example"

BranchName = "main"

}

}

}

stage {

name = "Build"

action {

name = "Build"

category = "Build"

owner = "AWS"

provider = "CodeBuild"

input\_artifacts = ["source\_output"]

output\_artifacts = ["build\_output"]

version = "1"

configuration = {

ProjectName = "test"

}

}

}

stage {

name = "Deploy"

action {

name = "Deploy"

category = "Deploy"

owner = "AWS"

provider = "CloudFormation"

input\_artifacts = ["build\_output"]

version = "1"

configuration = {

ActionMode = "REPLACE\_ON\_FAILURE"

Capabilities = "CAPABILITY\_AUTO\_EXPAND,CAPABILITY\_IAM"

OutputFileName = "CreateStackOutput.json"

StackName = "MyStack"

TemplatePath = "build\_output::sam-templated.yaml"

}

}

}

}

resource "aws\_codestarconnections\_connection" "example" {

name = "example-connection"

provider\_type = "GitHub"

}

resource "aws\_s3\_bucket" "codepipeline\_bucket" {

bucket = "test-bucket"

acl = "private"

}

resource "aws\_iam\_role" "codepipeline\_role" {

name = "test-role"

assume\_role\_policy = <<EOF

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": {

"Service": "codepipeline.amazonaws.com"

},

"Action": "sts:AssumeRole"

}

]

}

EOF

}

resource "aws\_iam\_role\_policy" "codepipeline\_policy" {

name = "codepipeline\_policy"

role = aws\_iam\_role.codepipeline\_role.id

policy = <<EOF

{

"Version": "2012-10-17",

"Statement": [

{

"Effect":"Allow",

"Action": [

"s3:GetObject",

"s3:GetObjectVersion",

"s3:GetBucketVersioning",

"s3:PutObject"

],

"Resource": [

"${aws\_s3\_bucket.codepipeline\_bucket.arn}",

"${aws\_s3\_bucket.codepipeline\_bucket.arn}/\*"

]

},

{

"Effect": "Allow",

"Action": [

"codebuild:BatchGetBuilds",

"codebuild:StartBuild"

],

"Resource": "\*"

}

]

}

EOF

}

CICI pipeline launched in terraform, directly pushing to s3 or GitHub this is the platform they pushing it will be automatically coded pipeline connect to the deploy to the continuous integration then continue deployment or continue delivery to the EKS or any instances