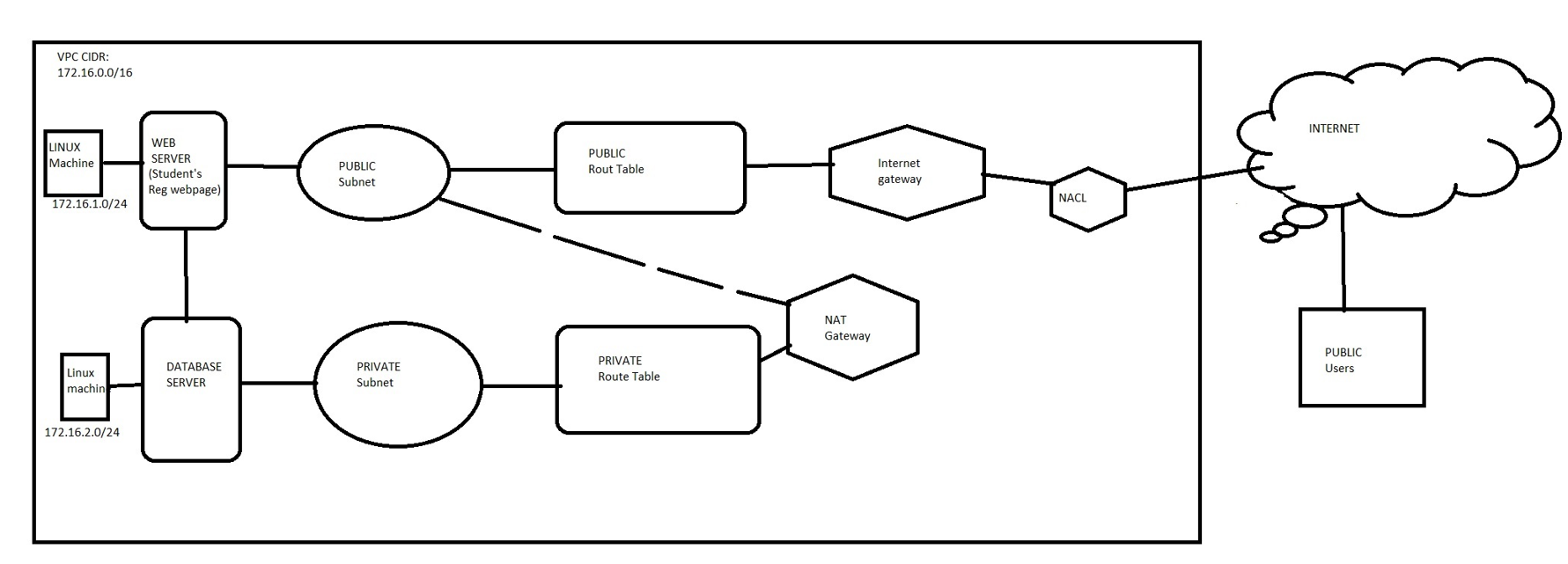
Creation of LAMP(LINUX,APACHE,MYSQL,PHP) through TERRAFORM (IAC)

Architecture:



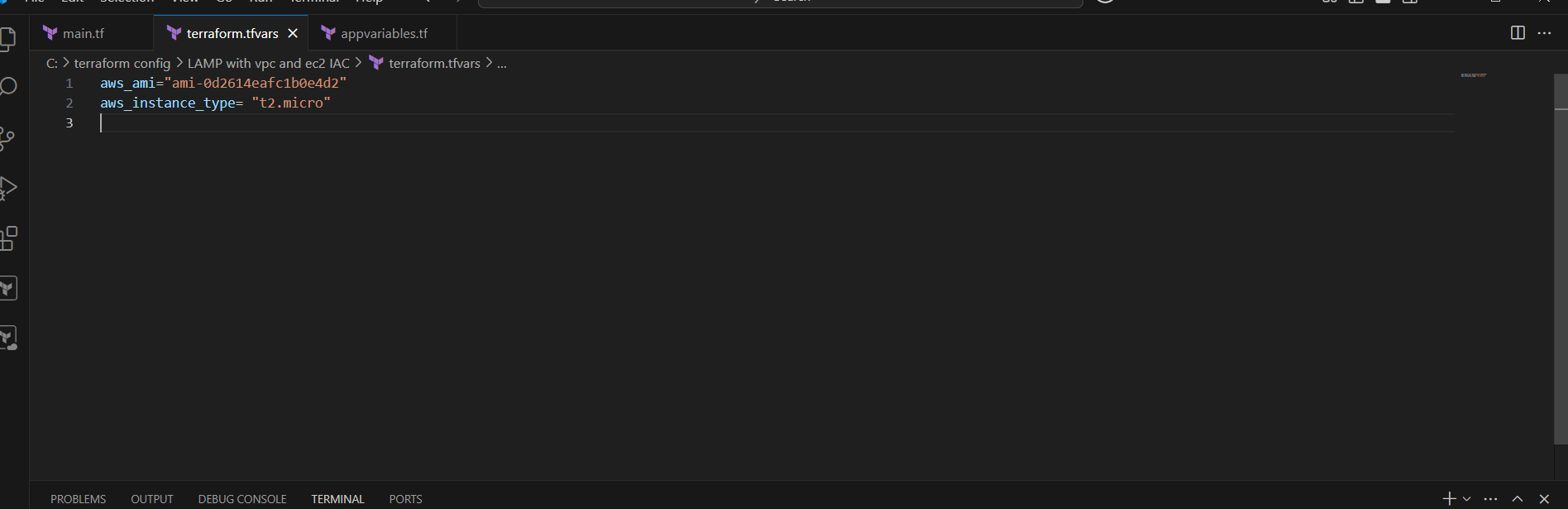
For the LAMP project we need to setup a infra of linux machine contains required package for HTTP, PHP and mysql(mariadb).

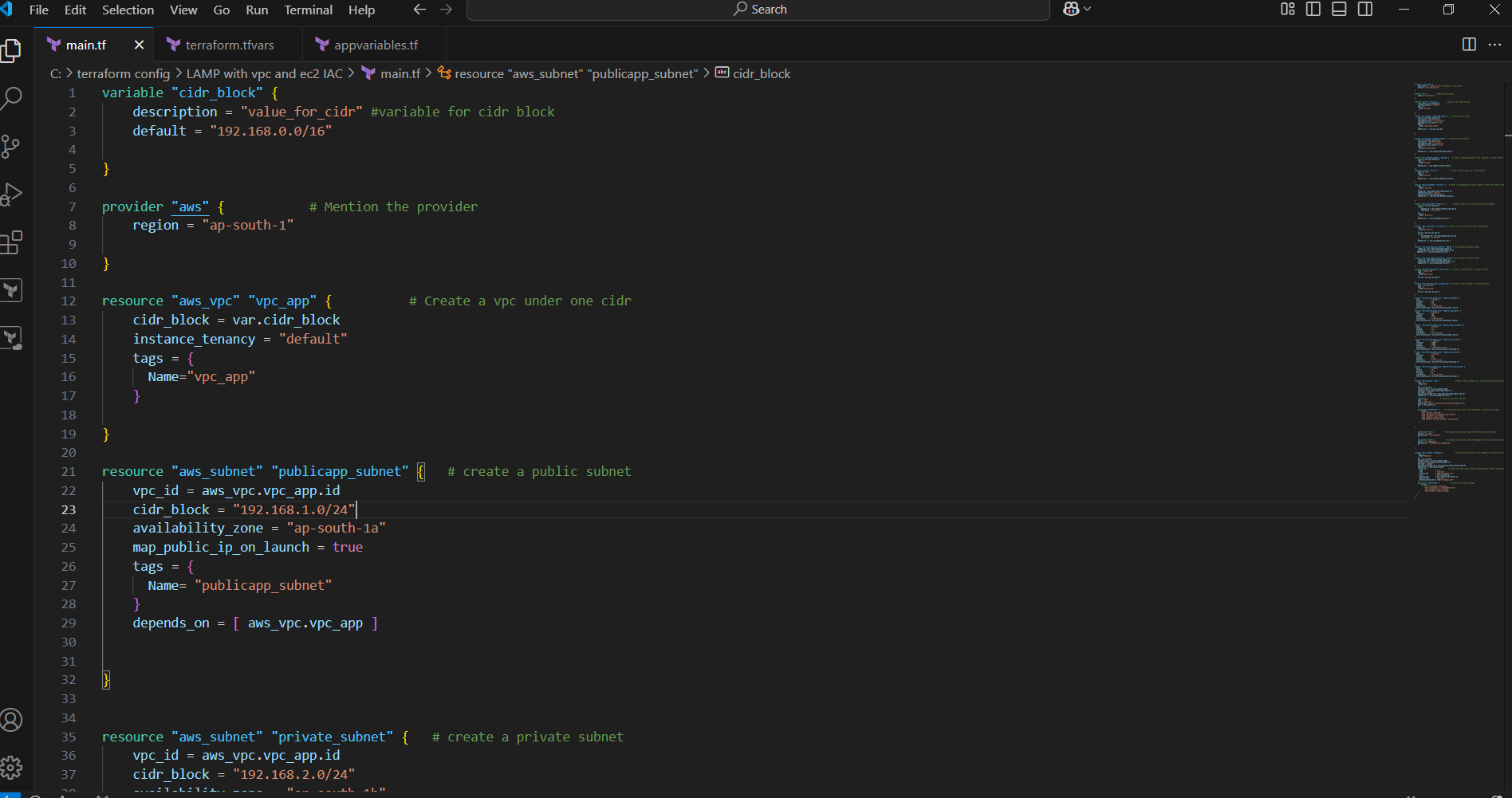
Through this infra we gonna create a database for student registration in private network and web page(php code) for registration and adding in public network.

We’re gonna setup this infra and req package into instance through INFRASTRUCTURE AS CODE by TERRAFORM.

TERRAFORM CONF:

I have set up the variable file and tfvars file individually, the variables for the resources are updated in the Variable.tf file and the Value for Variables are updated in the Terraform.tfvars file so can we implement the conf efficiently.





 availability\_zone = "ap-south-1b"

    map\_public\_ip\_on\_launch = false

    tags = {

      Name="private\_subnet"

    }

    depends\_on = [ aws\_subnet.publicapp\_subnet ]

}

resource "aws\_internet\_gateway" "igw\_app" {    # create a internet gateway for the resourece to access internet

    vpc\_id = aws\_vpc.vpc\_app.id

    tags = {

      Name="igw\_app"

    }

    depends\_on = [ aws\_subnet.private\_subnet ]

}

resource "aws\_eip" "nat\_ip" {               # create a elastic public ip for NAT gateway

    domain = "vpc"

    tags = {

      Name="nat\_eip"

    }

    depends\_on = [ aws\_internet\_gateway.igw\_app ]

}

resource "aws\_nat\_gateway" "nat\_priv" {    # Create a nat gateway for private subnet to access the internet under private sub

    tags = {

      Name= "nat\_priv"

    }

    subnet\_id = aws\_subnet.publicapp\_subnet.id

    connectivity\_type = "public"

    allocation\_id = aws\_eip.nat\_ip.id

    depends\_on = [ aws\_internet\_gateway.igw\_app ]

}

resource "aws\_route\_table" "public\_rt" {        #create a public route and route to internet gateway

    vpc\_id = aws\_vpc.vpc\_app.id

    route {

        gateway\_id = aws\_internet\_gateway.igw\_app.id

        cidr\_block = "0.0.0.0/0"

    }

    tags = {

      Name= "public\_rt"

    }

    depends\_on = [ aws\_nat\_gateway.nat\_priv ]

}

resource "aws\_route\_table" "private\_rt" {  # Create a private route and route to NAT gateway

    tags = {

      Name="private\_rt"

    }

    vpc\_id = aws\_vpc.vpc\_app.id

    route {

        nat\_gateway\_id = aws\_nat\_gateway.nat\_priv.id

        cidr\_block = "0.0.0.0/0"

    }

    depends\_on = [ aws\_nat\_gateway.nat\_priv ]

}

resource "aws\_route\_table\_association" "public" {  # Assosciate the public subnet

    subnet\_id = aws\_subnet.publicapp\_subnet.id

    route\_table\_id = aws\_route\_table.public\_rt.id

   depends\_on = [ aws\_nat\_gateway.nat\_priv ]

}

resource "aws\_route\_table\_association" "private" { # Associate the private subnet

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

    route\_table\_id = aws\_route\_table.private\_rt.id

    depends\_on = [ aws\_nat\_gateway.nat\_priv ]

}

resource "aws\_security\_group" "public\_sgw" {   # Create a security group for public instance

    name = "public\_sgw"

    tags = {

      Name="public\_sgw"

    }

    vpc\_id = aws\_vpc.vpc\_app.id

}

resource "aws\_security\_group" "private\_sgw" {  # Create a security group for private instance

    name = "private\_sgw"

    tags = {

      Name="private\_sgw"

    }

    vpc\_id = aws\_vpc.vpc\_app.id

}

resource "aws\_security\_group\_rule" "ingress\_ssh\_public" {

  type              = "ingress"

  from\_port         = 22

  to\_port           = 22

  protocol          = "tcp"

  cidr\_blocks       = ["0.0.0.0/0"]

  security\_group\_id = aws\_security\_group.public\_sgw.id

}

resource "aws\_security\_group\_rule" "ingress\_http\_public" {

  type              = "ingress"

  from\_port         = 80

  to\_port           = 80

  protocol          = "tcp"

  cidr\_blocks       = ["0.0.0.0/0"]

  security\_group\_id = aws\_security\_group.public\_sgw.id

}

resource "aws\_security\_group\_rule" "egress\_allow\_all\_public" {

  type              = "egress"

  to\_port           = 0

  protocol          = "-1"

  from\_port         = 0

  cidr\_blocks       = ["0.0.0.0/0"]

  security\_group\_id = aws\_security\_group.public\_sgw.id

}

resource "aws\_security\_group\_rule" "ingress\_sql\_private" {

  type              = "ingress"

  from\_port         = 3306

  to\_port           = 3306

  protocol          = "tcp"

  cidr\_blocks       = ["192.168.1.0/24"]

  security\_group\_id = aws\_security\_group.private\_sgw.id

}

resource "aws\_security\_group\_rule" "ingress\_ssh\_private" {

  type              = "ingress"

  from\_port         = 22

  to\_port           = 22

  protocol          = "tcp"

  cidr\_blocks       = ["0.0.0.0/0"]

  security\_group\_id = aws\_security\_group.private\_sgw.id

}

resource "aws\_security\_group\_rule" "egress\_allow\_all\_private" {

  type              = "egress"

  to\_port           = 0

  protocol          = "-1"

  from\_port         = 0

  cidr\_blocks       = ["0.0.0.0/0"]

  security\_group\_id = aws\_security\_group.private\_sgw.id

}

resource "aws\_instance" "web" {                   # Create a Public instance (for web) and associate with public subnet and sgw

    tags = {

      Name="Web"

    }

    ami = var.aws\_ami

    instance\_type = var.aws\_instance\_type

    subnet\_id = aws\_subnet.publicapp\_subnet.id

    key\_name = "appkey"

    vpc\_security\_group\_ids = [aws\_security\_group.public\_sgw.id]

    depends\_on = [ aws\_nat\_gateway.nat\_priv ]

    connection {                # connect to the public instance

    type = "ssh"

    user = "ec2-user"

    private\_key = file("C:/Users/Pranesh/Downloads/appkey.pem")

    host = self.public\_ip

    }

    provisioner "remote-exec" {     # by using provisioner install the req package for web server and php

        inline = [

        "echo 'machine connected'",

        "sudo yum install -y httpd php php-mysqlnd",

        "sudo systemctl start httpd",

        "sudo systemctl enable httpd",

        "sudo chown -R ec2-user:ec2-user /var/www/html"

     ]

}

    provisioner "file" {              # by using file provisioner copy the code from local to instance

    source = "E:/new2"

    destination = "/var/www/html"

    }

    provisioner "file" {               # by using file provisioner copy the keypair from local to public instance

    source = "E:/appkey.pem"

    destination = "/home/ec2-user/appkey.pem"

    }

}

resource "aws\_instance" "database" {              # Create a Private instance (for database) and associate with private subnet and sgw

    tags = {

      Name="Database"

    }

    ami = var.aws\_ami

    instance\_type = var.aws\_instance\_type

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

    key\_name = "appkey"

    vpc\_security\_group\_ids = [aws\_security\_group.private\_sgw.id]

    depends\_on = [ aws\_instance.web ]

    connection {                          # connect with the private instance through public instance using BASTION host, user, key

      type                = "ssh"

      user                = "ec2-user"

      private\_key         = file("E:/appkey.pem")

      host                = self.private\_ip

      bastion\_host        = aws\_instance.web.public\_ip

      bastion\_user        = "ec2-user"

      bastion\_private\_key = file("E:/appkey.pem")

    }

    provisioner "remote-exec" {             # install the req mysql package

        inline = [

            "echo 'connected to machine'",

            "sudo yum install -y mariadb105-server",

            "sudo systemctl start mariadb",

            "sudo systemctl enable mariadb"

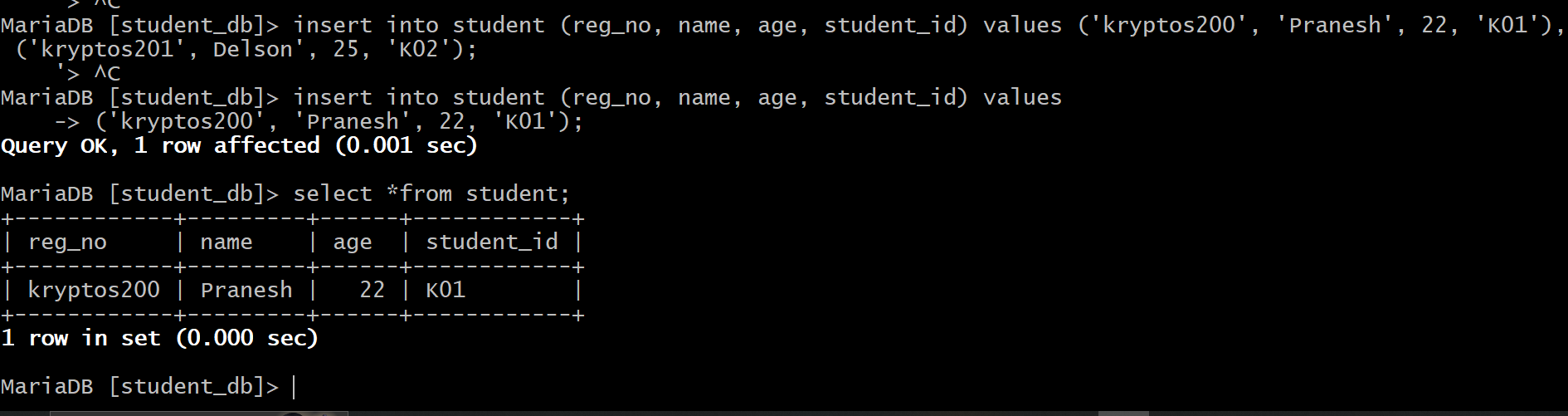
    ]

  }

}

So by the code we are creating a Public and private instance under the VPC, NAT, SG,and executing the operation of installing package, coping files, connecting the private instance through public instance all by executing the program itself.

After this connect to the private instance and create a database for student registration and create a user with password in sql and grant all priviledge to the user for the database with private ip of public instance so that public instance can connect with sql, and save the changes by giving flush priviledge;



Now to tighten the security vi /etc/my.cnf.d/mariadb-server.cnf and give private ip of private instance in BIND-ADDRESS so that sql can access only by private instance.

Now database is created go to public machine and copy the php code in path /var/www/html/<file-name>

**Php code:**

**In the php code give server name: private ip of private instance give username and password that created in sql.**

<?php

$servername = "localhost";

$username = "root";

$password = "";  // MySQL root password (empty if not set)

$dbname = "student\_db";

// Create connection

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect\_error) {

    die("Connection failed: " . $conn->connect\_error);

}

$reg\_no = "";

$name = "";

$age = "";

$student\_id = "";

$message = "";

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

Now give the public ip of the public instance with the path to get the web page.

