Details Code: This Terraform script automatically provisions an AWS EC2 instance, secures it with an SSH key and security group, and returns the instance's public I

```
resource "aws_key_pair" "my_ssh_key" {
   key_name = "terra-key-auto"
   public_key = file("/home/ubuntu/terra-key-auto.pub")
}



resource "aws_default_vpc" "default" {
}
resource "aws_security_group" "my_sg" {
   name = "my_sg"
   description = "This is terra auto description"
   vpc_id = aws_default_vpc.default.id
   ingress {
      from_port = 22
   to_port = 22
   protocol = "tcp"
   cidr_blocks = ["0.0.0.0/0"]
   description = "for incoming world"
   }


   egress {
   from_port = 0
   to_port = 0
   protocol = "-1"
   cidr_blocks = ["0.0.0.0/0"]
   description = "For utside world"
```

```
}


}
resource "aws_instance" "my_instance" {
 tags = {
   Name = "My-Auto-Server"
 }

 ami           = "ami-075686beab831bb7f"
 instance_type = "t2.micro"
 key_name      = aws_key_pair.my_ssh_key.key_name
 security_groups = [aws_security_group.my_sg.name]
}

#resource "aws_ec2_instance_state" "my_state" {
#   instance_id = aws_instance.my_instance.id
#   state = "stopped"
#}
output "my_ec2_ip" {
   value = aws_instance.my_instance.public_i
}
```

Explanation:

1. **Creating an SSH Key Pair**

resource "aws_key_pair" "my_ssh_key" { key_name = "terra-key-auto" public_key = file("/home/ubuntu/terra-key-auto.pub") }

```hcl
resource "aws_key_pair" "my_ssh_key" {
    key_name = "terra-key-auto"
    public_key = file("/home/ubuntu/terra-key-auto.pub")
}
```

- This defines an AWS key pair named **"terra-key-auto"**.
- It uses an existing public key from the specified file (/home/ubuntu/terra-key-auto.pub).
- This key pair is used for secure SSH access to the instance.

2. **Using the Default VPC**

```hcl
resource "aws_default_vpc" "default" {}
```

- This ensures Terraform will use the default AWS VPC.
- No additional configuration is provided, meaning it adopts AWS default settings.

3. **Creating a Security Group**

```hcl
resource "aws_security_group" "my_sg" {
    name = "my_sg"
    description = "This is terra auto description"
    vpc_id = aws_default_vpc.default.id
```

- This defines a security group named **"my_sg"** inside the default VPC.
- Security groups control inbound and outbound traffic.

Ingress (Inbound Rule)

```hcl
    ingress {
        from_port = 22
        to_port = 22
        protocol = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
        description = "for incoming world"
    }
```

- Allows **SSH (port 22)** traffic from anywhere (0.0.0.0/0).
- This rule makes the instance accessible from any IP.

Egress (Outbound Rule)

```hcl
    egress {
        from_port = 0
        to_port = 0
        protocol = "-1"
        cidr_blocks = ["0.0.0.0/0"]
        description = "For utside world"
    }
```

- Allows **all outbound traffic** (protocol -1 means all types).
- Ensures the instance can communicate externally.

4. **Launching an EC2 Instance**

```hcl
resource "aws_instance" "my_instance" {
  tags = {
    Name = "My-Auto-Server"
  }
  ami           = "ami-075686beab831bb7f"
  instance_type = "t2.micro"
  key_name      = aws_key_pair.my_ssh_key.key_name
  security_groups = [aws_security_group.my_sg.name]
}
```

- Creates an EC2 instance named **"My-Auto-Server"**.
- Uses the specified **Amazon Machine Image (AMI)** (ami-075686beab831bb7f).
- Uses the **"t2.micro"** instance type.
- Assigns the previously defined **security group** and **SSH key pair**.

5. **(Optional) Stopping the Instance**

```hcl
# resource "aws_ec2_instance_state" "my_state" {
#    instance_id = aws_instance.my_instance.id
#    state = "stopped"
# }
```

resource "aws_ec2_instance_state" "my_state" { # instance_id = aws_instance.my_instance.id # state = "stopped" # }

- This block is **commented out** and doesn't execute.
- If uncommented, it ensures the instance is **stopped** instead of running.

6. **Outputting the Instance's Public IP**

```hcl
output "my_ec2_ip" {
    value = aws_instance.my_instance.public_ip
}
```

- Displays the **public IP** of the EC2 instance after Terraform execution.
- Useful for accessing the instance remotely.