Terraform EC2 Login Using Locally Created Key Pair with Remote-Exec

This tutorial explains how to create an SSH key pair locally, use it in Terraform to provision an EC2 instance, and automatically log in using the **remote-exec** provisioner to run commands during *terraform apply*.

Step 1: Create SSH Key Locally

Run the following command in your terminal to create the key pair:

```
ssh-keygen -t rsa -b 4096 -f mykey
```

This generates two files:

1. **mykey** – Private key (keep it secure)
2. **mykey.pub** – Public key (shared with AWS)

Step 2: Terraform Configuration

Here's the Terraform code to upload your local public key, create the EC2 instance, and log in automatically to run setup commands:

```
provider "aws" {
  region = "ap-south-1"
}

resource "aws_key_pair" "my_key" {
  key_name   = "my-local-key"
  public_key = file("${path.module}/mykey.pub")
}

resource "aws_instance" "second-ec2" {
  ami           = "ami-084e7e1456028650e"
  instance_type = "t2.micro"
  key_name      = aws_key_pair.my_key.key_name

  provisioner "remote-exec" {
    inline = [
      "echo 'Hello from Terraform!' > /home/ec2-user/terraform_test.txt",
      "sudo yum install -y httpd",
      "sudo systemctl start httpd",
      "sudo systemctl enable httpd"
    ]

    connection {
      type        = "ssh"
      user        = "ec2-user"
      private_key = file("${path.module}/mykey")
      host        = self.public_ip
    }
  }

  tags = {
    Name = "EC2-Using-Local-Key"
  }
}
```

Step 3: How It Works

1. You manually create an SSH key pair on your machine.
2. Terraform uploads the public key to AWS as an EC2 key pair.

3. Terraform creates the EC2 instance using that key pair.
4. The **remote-exec** provisioner uses your local private key to SSH into the instance.
5. The specified commands are run on the EC2 instance during provisioning.

Step 4: Manual SSH Access

After Terraform finishes, you can still manually log in with:

```
ssh -i mykey ec2-user@<PUBLIC_IP>
```

Best Practices

1. Never commit your private key file to version control.
2. Use correct file permissions for the private key (chmod 400).
3. Store Terraform state remotely with locking (e.g., S3 + DynamoDB) in team environments.
4. Keep your key name consistent across environments to avoid confusion.

This method provides both security and automation, allowing Terraform to use your locally generated key pair for EC2 provisioning and initial setup without storing sensitive private keys in the Terraform state file.