Quick and Easy Linux Instance Setup: Get Started in Minutes

Setting up Linux instances for development and testing can get tiresome. So, to eliminate the hassle, I've created a Terraform **template that deploys the instance, creates key pair for you on your local machine, and outputs the ssh script for you to sign in and get going.**

If you want an instance now, clone this repo to an environment that supports Terraform deployments on AWS, `cd basic-ec2-instance` and `terraform init && terraform apply -auto-approve`.

If you'd like to learn more about how it works, I'll walk you through the setup below.

Pre-requisites

- A basic understanding of Terraform
- An environment that can deploy Terraform to an AWS account (If you don't have one, AWS' Cloud9 IDE comes with Terraform and AWS installed by default)

Setup

1. Please create a new directory for your Terraform project and navigate to it.
2. Create the following director

```
project/
│
├── main.tf
├── providers.tf
│
└── modules/
    ├── security/
    │   └── main.tf
    │
    └── basic_ec2_instance/
        └── main.tf
```

3. Create a .gitignore file to prevent sensitive Terraform and .pem files from being committed

```
# Ignore all .pem files
*.pem*

# Local .terraform directories
**/.terraform/*

# .tfstate files
*.tfstate
*.tfstate.*

# Crash log files
crash.log

# Ignore override files as they are usually used to override resources locally and
# are not checked in
override.tf
override.tf.json
*_override.tf
*_override.tf.json
```

Setup the Root Level Files

This root-level code deploys an EC2 instance on AWS. It declares the necessary resources for the EC2 instance deployment, including the AWS provider, a data source

to fetch the current public IP address, a security module to create security groups and key pairs, and a module to create the basic EC2 instance. We'll get into the details of the modules in the following two sections.

providers.tf

```
#  -- root/providers.tf

# Declare the AWS provider as a required provider for the Terraform configuration
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"  # Specify the source of the AWS provider module
    }
  }
}

# Configure the AWS provider with the region to use
provider "aws" {
  region = "us-east-1"  # Specify the AWS region to use for resources in this con
}
```

main.tf

```
#  -- root/main.tf

# Define a data source to fetch the public IP address of the machine running Terra
data "http" "my_ip" {
  url = "https://ipv4.icanhazip.com"
}

# Define a local variable to store the current public IP address from the data sou
locals {
  current_ip = chomp(data.http.my_ip.body)  # Remove trailing whitespace from the
}

# Define a security module to create security groups and key pairs
module "security" {
  source = "./modules/security"  # Specify the location of the security module
  my_ip_address = local.current_ip  # Pass the current IP address to the security
```

```
}

# Define a module to create a basic EC2 instance
module "basic_ec2_instance" {
  source = "./modules/basic_ec2_instance"  # Specify the location of the EC2 insta
  security_group_id = module.security.security_group_id  # Pass the security group
  key_pair_name     = module.security.key_pair_name  # Pass the key pair name fro
}

# Define an output to display the SSH command to connect to the EC2 instance
output "ssh_command" {
  value = format("ssh -i %s ec2-user@%s", module.security.private_key_file, module
  description = "The SSH command to connect to the basic_ec2_instance"
}
```

Making the Security Module

security/variables.tf

This code creates a secure environment for an EC2 instance by creating a security group with strict ingress and egress rules and generating a key pair for secure SSH access to the instance. The output variables provide the necessary information to configure the EC2 instance in the main Terraform configuration.

The `variables.tf` file defines a variable called `my_ip_address` that will store the user's current IP address.

The `main.tf` file creates a TLS private key resource, an AWS key pair resource, a resource to store the key pair on your local machine, an AWS security group resource that allows ingress only from your machine, and a null resource that ensures the key pair has proper permissions.

The `outputs.tf` file defines output variables for the security group ID, key pair name, and private key file path. These outputs will be used in the main Terraform configuration to set up the EC2 instance.

```
# -- security/variables.tf
```

```
# Declare a variable to store the user's current IP address
variable "my_ip_address" {
  description = "Your current IP address"
  type        = string
}security/main.tf
```

```
# -- security/main.tf

# Create a TLS private key resource with the specified algorithm
resource "tls_private_key" "my_key_pair" {
  algorithm = "RSA"  # Specify the algorithm for the private key
}

# Create an AWS key pair resource with the specified name and public key
resource "aws_key_pair" "my_key_pair" {
  key_name   = "my_key_pair"
  public_key = tls_private_key.my_key_pair.public_key_openssh  # Use the public ke
}

# Create a local file resource to store the private key in a file
  content  = tls_private_key.my_key_pair.private_key_pem  # Specify the content o
  filename = "my_key_pair.pem"
}

# Create an AWS security group resource with the specified name and ingress/egress
resource "aws_security_group" "my_security_group" {
  name = "my_security_group"

  ingress {  # Define an ingress rule to allow SSH traffic from the user's IP addr
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["${var.my_ip_address}/32"]
  }

  ingress {  # Define an ingress rule to allow HTTP traffic from the user's IP add
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["${var.my_ip_address}/32"]
  }

  egress {  # Define an egress rule to allow all traffic to all destinations
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
```

```
  }
}

# Create a null resource to chmod the private key file after it's created
resource "null_resource" "chmod_private_key" {
  depends_on = [local_file.private_key]  # Make sure the local file is created be

  provisioner "local-exec" {  # Use a local-exec provisioner to change the permis
    command = "chmod 400 ${local_file.private_key.filename}"
  }
}
```

security/output.tf

```
# -- security/outputs.tf

output "security_group_id" {
  value = aws_security_group.my_security_group.id
}

output "key_pair_name" {
  value = aws_key_pair.my_key_pair.key_name
}

output "private_key_file" {
  value       = local_file.private_key.filename
  description = "The path to the private key file"
}
```

Preparing the EC2 Module

This Terraform configuration creates an EC2 instance on AWS with the required security group and key pair to ensure secure access. The output variable provides the public IP address of the instance, making it straightforward to connect to the instance via SSH.

The `variables.tf` file defines two variables: `security_group_id` and `key_pair_name`. These variables will be used to reference the security group and key pair created in the `security` module.

The `main.tf` file creates a data source for the most recent Amazon Linux 2 AMI and uses it to launch a new EC2 instance. The instance is launched with the specified `instance_type`, `key_name`, and `vpc_security_group_ids`. The `tags` block provides a name for the instance.

The `outputs.tf` file defines a single output variable for the public IP address of the instance. This output will be used in the main Terraform configuration to display the SSH command to connect to the instance.

basic_ec2_instance/variables.tf

```
# -- basic_ec2_instance/variables.tf

variable "security_group_id" {
  description = "The ID of the security group"
  type        = string
}

variable "key_pair_name" {
  description = "The name of the key pair"
  type        = string
}
```

basic_ec2_instance/main.tf

```
# -- basic_ec2_instance/main.tf

data "aws_ami" "amazon_linux_2" {
  most_recent = true

  filter {
    name   = "name"
    values = ["amzn2-ami-hvm-2.0*"]
  }

  owners = ["amazon"]
}

resource "aws_instance" "basic_ec2_instance" {
```

```
  ami           = data.aws_ami.amazon_linux_2.id
  instance_type = "t2.micro"
  key_name      = var.key_pair_name

  vpc_security_group_ids = [var.security_group_id]

  tags = {
    Name = "basic_ec2_instance"
  }
}
```

basic_ec2_instance/outputs.tf

```
# -- basic_ec2_instance/outputs.tf

output "public_ip" {
  value       = aws_instance.basic_ec2_instance.public_ip
  description = "The public IP of the basic_ec2_instance"
}
```

Wrap Up

Once you're ready, make sure you run this template to experience the ease and speed with which you can set up a remote server you can access from your laptop.

I find this incredibly useful when developing new scripts, taking a new library for a spin, or brushing up on CLI skills. I hope you do as well.

Automation Terraform Ec2 Instance Linux Tutorial