# Automating Jenkins Server Setup on AWS with Terraform

In this article, I'll demonstrate how to automate the setup of a Jenkins server on an AWS EC2 instance using Terraform. Jenkins is a popular open-source automation server that helps automate parts of the software development process. Using Terraform, we can provision the necessary AWS resources and install Jenkins seamlessly.

## Prerequisites

Before you begin, ensure you have the following:

- An AWS account
- Terraform installed on your local machine
- SSH keys generated for accessing the EC2 instance

**For complete code and shell script please visit to my GitHub repository and download the code https://github.com/ajayumredkar78/aws-jenkins-server.git**

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

# Creating the security group required for jenkins instance
resource "aws_security_group" "allow_ssh_http" {
  name        = "allow_ssh_http"
  description = "Allow SSH and HTTP access"

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    from_port   = 8080
    to_port     = 8080
    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"]
  }
}
```

```
# Creating the resource Jenkins server with Ubuntu OS
resource "aws_instance" "jenkins_server" {
  ami           = data.aws_ami.ubuntu.id
  instance_type = "t2.micro"
  key_name      = "terrakey"
  security_groups = [aws_security_group.allow_ssh_http.name]

  # user data has been saved in the jenkins-script.sh, Should be in the
same folder to execute the installation files.
  user_data = file("jenkins-script.sh")

  tags = {
    Name = "JenkinsServer"
  }
}

data "aws_ami" "ubuntu" {
  most_recent = true
  owners      = ["099720109477"]

  filter {
    name   = "name"
    values = ["ubuntu/images/hvm-ssd/ubuntu-*-amd64-server-*"]
  }

  filter {
    name   = "virtualization-type"
    values = ["hvm"]
  }

  filter {
    name   = "root-device-type"
    values = ["ebs"]
  }
}
```

## Terraform Code Breakdown

The following Terraform script sets up an AWS EC2 instance with Jenkins installed. Let's break down the code step by step.

### Provider Configuration

First, we configure the AWS provider to specify the AWS region where we want to create our resources:

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

### Security Group for Jenkins

Next, we create a security group to allow SSH and HTTP access to our Jenkins server:

```
resource "aws_security_group" "allow_ssh_http" {
  name        = "allow_ssh_http"
  description = "Allow SSH and HTTP access"

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    from_port   = 8080
    to_port     = 8080
    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"]
  }
}
```

- **Ingress rules** allow SSH (port 22) and HTTP (port 8080) access from any IP address (`0.0.0.0/0`).
- **Egress rule** allows all outbound traffic.

**Data Source for Ubuntu AMI**

We use a data source to find the latest Ubuntu AMI (Amazon Machine Image):

```
data "aws_ami" "ubuntu" {
  most_recent = true
  owners      = ["099720109477"]

  filter {
    name   = "name"
    values = ["ubuntu/images/hvm-ssd/ubuntu-*-amd64-server-*"]
  }

  filter {
    name   = "virtualization-type"
    values = ["hvm"]
  }

  filter {
    name   = "root-device-type"
    values = ["ebs"]
  }
}
```

This configuration fetches the most recent Ubuntu AMI owned by Canonical (owner ID `099720109477`).

**EC2 Instance for Jenkins**

We define the EC2 instance resource and specify the AMI, instance type, key pair for SSH access, and security group. We also provide a user data script to install Jenkins:

```
resource "aws_instance" "jenkins_server" {
  ami           = data.aws_ami.ubuntu.id
  instance_type = "t2.micro"
  key_name      = "terrakey"
  security_groups = [aws_security_group.allow_ssh_http.name]

  user_data = file("jenkins-script.sh")

  tags = {
    Name = "JenkinsServer"
  }
}
```

- **AMI**: Uses the latest Ubuntu AMI fetched earlier.
- **Instance type**: `t2.micro` for a cost-effective option.
- **Key name**: `terrakey` (Ensure you have the corresponding private key file).
- **Security group**: The one we created earlier to allow SSH and HTTP access.
- **User data**: A script (`jenkins-script.sh`) to install Jenkins on the instance startup.
- **Tags**: Adds a tag to name the instance "JenkinsServer".

## Jenkins Installation Script

The `jenkins-script.sh` file should be in the same directory as your Terraform configuration file and contains the commands to install Jenkins:

```
#!/bin/bash
# Update package list
sudo apt-get update -y
# Install Java (Jenkins requires Java to run)
sudo apt-get install -y openjdk-11-jdk
# Add the Jenkins repository and key
wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -
sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d
# Update package list again
sudo apt-get update -y
# Install Jenkins
sudo apt-get install -y jenkins
# Start Jenkins service
sudo systemctl start jenkins
# Enable Jenkins service to start on boot
sudo systemctl enable jenkins
# Print the initial admin password to a known file location
sudo cat /var/lib/jenkins/secrets/initialAdminPassword > /tmp/jenkins_initial_admin_passwo
```

## Running the Terraform Script

To execute this script:

1. **Initialize Terraform**: Run `terraform init` to initialize the working directory.
2. **Apply the Configuration**: Run `terraform apply` to create the resources. Review the plan and confirm the apply step.

After the resources are created, you can SSH into the instance using your private key and check the initial Jenkins admin password stored at `/tmp/jenkins_initial_admin_password`.

## Conclusion

With this Terraform script, you can quickly set up a Jenkins server on AWS, automating the provisioning of infrastructure and software installation. This setup helps streamline the deployment process, allowing you to focus on your development and CI/CD pipelines.

Feel free to customize the script further based on your specific requirements. Happy automating!

This article walks through the Terraform code step by step, explaining each part and how it contributes to setting up a Jenkins server on AWS.