Build AWS EC2 Instances, Security Groups using Terraform

Step-01: Introduction

Terraform Modules we will use

- terraform-aws-modules/vpc/aws
- terraform-aws-modules/security-group/aws
- terraform-aws-modules/ec2-instance/aws

Terraform New Concepts we will introduce

- aws_eip
- null_resource
- file provisioner
- remote-exec provisioner
- local-exec provisioner
- depends_on Meta-Argument

What are we going implement?

- Create VPC with 3-Tier Architecture (Web, App and DB) - Leverage code from previous section
- Create AWS Security Group Terraform Module and define HTTP port 80, 22 inbound rule for entire internet access 0.0.0.0/0
- Create Multiple EC2 Instances in VPC Private Subnets and install
- Create EC2 Instance in VPC Public Subnet Bastion Host
- Create Elastic IP for Bastion Host EC2 Instance
- Create null_resource with following 3 Terraform Provisioners
    - File Provisioner
    - Remote-exec Provisioner
    - Local-exec Provisioner

Pre-requisite

- Copy your AWS EC2 Key pair terraform-key.pem in private-key folder
- Folder name local-exec-output-files where local-exec provisioner creates a file (creation-time provisioner)

Step-02: Copy all the VPC TF Config files from 06-02

- Copy the following TF Config files from 06-02 section which will create a 3-Tier VPC
- c1-versions.tf

```
# Terraform Block
terraform {
  required_version = ">= 1.6" # which means any version equal & above 0.14 like 0.15, 0.16 etc and < 1.xx
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = ">= 5.0"
    }

      null = {
      source = "hashicorp/null"
      version = "~> 3.0"
    }

  }
}

# Provider Block
provider "aws" {
  region  = var.aws_region
  profile = "default"
}
/*
Note-1:  AWS Credentials Profile (profile = "default") configured on your local desktop terminal
$HOME/.aws/credentials
*/
```

- c2-generic-variables.tf

```
# Input Variables
# AWS Region
variable "aws_region" {
  description = "Region in which AWS Resources to be created"
  type = string
  default = "us-east-1"
}
# Environment Variable
variable "environment" {
  description = "Environment Variable used as a prefix"
  type = string
  default = "dev"
}
```

```
# Business Division
variable "business_divsion" {
  description = "Business Division in the large organization this Infrastructure belongs"
  type = string
  default = "SAP"
}
```

c3-local-values.tf

```
# Define Local Values in Terraform
locals {
  owners = var.business_divsion
  environment = var.environment
  name = "${var.business_divsion}-${var.environment}"
  #name = "${local.owners}-${local.environment}"
  common_tags = {
    owners = local.owners
    environment = local.environment
  }
}
```

c4-01-vpc-variables.tf

```
# VPC Input Variables


# VPC Name
variable "vpc_name" {
 description = "VPC Name"
 type = string
 default = "myvpc"
}


# VPC CIDR Block
variable "vpc_cidr_block" {
 description = "VPC CIDR Block"
 type = string
 default = "10.0.0.0/16"
}
```

```
# VPC Availability Zones
variable "vpc_availability_zones" {
  description = "VPC Availability Zones"
  type = list(string)
  default = ["us-east-1a", "us-east-1b"]
}

# VPC Public Subnets
variable "vpc_public_subnets" {
  description = "VPC Public Subnets"
  type = list(string)
  default = ["10.0.101.0/24", "10.0.102.0/24"]
}

# VPC Private Subnets
variable "vpc_private_subnets" {
  description = "VPC Private Subnets"
  type = list(string)
  default = ["10.0.1.0/24", "10.0.2.0/24"]
}

# VPC Database Subnets
variable "vpc_database_subnets" {
  description = "VPC Database Subnets"
  type = list(string)
  default = ["10.0.151.0/24", "10.0.152.0/24"]
}

# VPC Create Database Subnet Group (True / False)
variable "vpc_create_database_subnet_group" {
```

```
  description = "VPC Create Database Subnet Group"
  type = bool
  default = true
}

# VPC Create Database Subnet Route Table (True or False)
variable "vpc_create_database_subnet_route_table" {
  description = "VPC Create Database Subnet Route Table"
  type = bool
  default = true
}


# VPC Enable NAT Gateway (True or False)
variable "vpc_enable_nat_gateway" {
  description = "Enable NAT Gateways for Private Subnets Outbound Communication"
  type = bool
  default = true
}

# VPC Single NAT Gateway (True or False)
variable "vpc_single_nat_gateway" {
  description = "Enable only single NAT Gateway in one Availability Zone to save costs during our demos"
  type = bool
  default = true
}
```

- c4-02-vpc-module.tf

```
# Create VPC Terraform Module
module "vpc" {
  source  = "terraform-aws-modules/vpc/aws"
  #version = "2.78.0"
  #version = "~> 2.78"
  version = "5.4.0"

  # VPC Basic Details
  name = "${local.name}-${var.vpc_name}"
  cidr = var.vpc_cidr_block
  azs             = var.vpc_availability_zones
  public_subnets  = var.vpc_public_subnets
  private_subnets = var.vpc_private_subnets

  # Database Subnets
  database_subnets = var.vpc_database_subnets
  create_database_subnet_group = var.vpc_create_database_subnet_group
  create_database_subnet_route_table = var.vpc_create_database_subnet_route_table
  # create_database_internet_gateway_route = true
  # create_database_nat_gateway_route = true

  # NAT Gateways - Outbound Communication
  enable_nat_gateway = var.vpc_enable_nat_gateway
  single_nat_gateway = var.vpc_single_nat_gateway

  # VPC DNS Parameters
  enable_dns_hostnames = true
  enable_dns_support   = true

  tags = local.common_tags
  vpc_tags = local.common_tags

  # Additional Tags to Subnets
  public_subnet_tags = {
    Type = "Public Subnets"
  }
  private_subnet_tags = {
    Type = "Private Subnets"
  }
  database_subnet_tags = {
    Type = "Private Database Subnets"
  }
}
```

c4-03-vpc-outputs.tf

```
# VPC Output Values

# VPC ID
output "vpc_id" {
  description = "The ID of the VPC"
  value       = module.vpc.vpc_id
}

# VPC CIDR blocks
output "vpc_cidr_block" {
  description = "The CIDR block of the VPC"
  value       = module.vpc.vpc_cidr_block
}

# VPC Private Subnets
output "private_subnets" {
  description = "List of IDs of private subnets"
  value       = module.vpc.private_subnets
}

# VPC Public Subnets
output "public_subnets" {
  description = "List of IDs of public subnets"
  value       = module.vpc.public_subnets
}

# VPC NAT gateway Public IP
output "nat_public_ips" {
  description = "List of public Elastic IPs created for AWS NAT Gateway"
  value       = module.vpc.nat_public_ips
```

```
}

# VPC AZs
output "azs" {
  description = "A list of availability zones spefified as argument to this module"
  value       = module.vpc.azs
}


# VPC Output Values

# VPC ID
output "vpc_id" {
  description = "The ID of the VPC"
  value       = module.vpc.vpc_id
}

# VPC CIDR blocks
output "vpc_cidr_block" {
  description = "The CIDR block of the VPC"
  value       = module.vpc.vpc_cidr_block
}

# VPC Private Subnets
output "private_subnets" {
  description = "List of IDs of private subnets"
  value       = module.vpc.private_subnets
}

# VPC Public Subnets
output "public_subnets" {
```

```
 description = "List of IDs of public subnets"
 value       = module.vpc.public_subnets
}

# VPC NAT gateway Public IP
output "nat_public_ips" {
 description = "List of public Elastic IPs created for AWS NAT Gateway"
 value       = module.vpc.nat_public_ips
}

# VPC AZs
output "azs" {
 description = "A list of availability zones spefified as argument to this module"
 value       = module.vpc.azs
}
```

terraform.tfvars

```
# Generic Variables
aws_region = "us-east-1"
environment = "stag"
business_divsion = "HR"
```

- vpc.auto.tfvars

```
# VPC Variables
vpc_name = "myvpc"
vpc_cidr_block = "10.0.0.0/16"
vpc_availability_zones = ["us-east-1a", "us-east-1b"]
vpc_public_subnets = ["10.0.101.0/24", "10.0.102.0/24"]
vpc_private_subnets = ["10.0.1.0/24", "10.0.2.0/24"]
vpc_database_subnets= ["10.0.151.0/24", "10.0.152.0/24"]
vpc_create_database_subnet_group = true
vpc_create_database_subnet_route_table = true
```

- `vpc_enable_nat_gateway = true`
- `vpc_single_nat_gateway = true`

- private-key/terraform-key.pem

Step-03: Add app1-install.sh

- Add app1-install.sh in working directory

#! /bin/bash

Instance Identity Metadata Reference - https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-identity-documents.html

sudo yum update -y

sudo yum install -y httpd

sudo systemctl enable httpd

sudo service httpd start

sudo echo '<h1>Welcome to StackSimplify - APP-1</h1>' | sudo tee /var/www/html/index.html

sudo mkdir /var/www/html/app1

sudo echo '<!DOCTYPE html> <html> <body style="background-color:rgb(250, 210, 210);"> <h1>Welcome to Stack Simplify - APP-1</h1> Terraform Demo Application Version: V1 </body></html>' | sudo tee /var/www/html/app1/index.html

sudo curl http://169.254.169.254/latest/dynamic/instance-identity/document -o /var/www/html/app1/metadata.html

Step-04: Create Security Groups for Bastion Host and Private Subnet Hosts

Step-04-01: c5-01-securitygroup-variables.tf

- Place holder file for defining any Input Variables for EC2 Security Groups

Step-04-02: c5-03-securitygroup-bastionsg.tf

- SG Module Examples for Reference

AWS EC2 Security Group Terraform Module

Security Group for Public Bastion Host

module "public_bastion_sg" {

source = "terraform-aws-modules/security-group/aws"

version = "3.18.0"

name = "public-bastion-sg"

```
  description = "Security group with SSH port open for everybody (IPv4 CIDR), egress ports are all world open"
  vpc_id      = module.vpc.vpc_id
  # Ingress Rules & CIDR Block
  ingress_rules = ["ssh-tcp"]
  ingress_cidr_blocks = ["0.0.0.0/0"]
  # Egress Rule - all-all open
  egress_rules = ["all-all"]
  tags = local.common_tags
}
```

Step-04-03: c5-04-securitygroup-privatesg.tf

```
# AWS EC2 Security Group Terraform Module
# Security Group for Private EC2 Instances
module "private_sg" {
  source  = "terraform-aws-modules/security-group/aws"
  version = "3.18.0"

  name        = "private-sg"
  description = "Security group with HTTP & SSH ports open for everybody (IPv4 CIDR), egress ports are all world open"
  vpc_id      = module.vpc.vpc_id
  ingress_rules = ["ssh-tcp", "http-80-tcp"]
  ingress_cidr_blocks = ["0.0.0.0/0"]
  egress_rules = ["all-all"]
  tags = local.common_tags
}
```

Step-04-04: c5-02-securitygroup-outputs.tf

- SG Module Examples for Reference

```
# Public Bastion Host Security Group Outputs
output "public_bastion_sg_group_id" {
  description = "The ID of the security group"
  value       = module.public_bastion_sg.this_security_group_id
```

```
}
output "public_bastion_sg_group_vpc_id" {
  description = "The VPC ID"
  value       = module.public_bastion_sg.this_security_group_vpc_id
}
output "public_bastion_sg_group_name" {
  description = "The name of the security group"
  value       = module.public_bastion_sg.this_security_group_name
}


# Private EC2 Instances Security Group Outputs
output "private_sg_group_id" {
  description = "The ID of the security group"
  value       = module.private_sg.this_security_group_id
}
output "private_sg_group_vpc_id" {
  description = "The VPC ID"
  value       = module.private_sg.this_security_group_vpc_id
}
output "private_sg_group_name" {
  description = "The name of the security group"
  value       = module.private_sg.this_security_group_name
}
```

Step-05: c6-01-datasource-ami.tf

```
# Get latest AMI ID for Amazon Linux2 OS
data "aws_ami" "amzlinux2" {
  most_recent = true
  owners = [ "amazon" ]
  filter {
    name = "name"
```

```
    values = [ "amzn2-ami-hvm-*-gp2" ]
  }
  filter {
    name = "root-device-type"
    values = [ "ebs" ]
  }
  filter {
    name = "virtualization-type"
    values = [ "hvm" ]
  }
  filter {
    name = "architecture"
    values = [ "x86_64" ]
  }
}
```

Step-06: EC2 Instances

Step-06-01: c7-01-ec2instance-variables.tf

```
# AWS EC2 Instance Type
variable "instance_type" {
  description = "EC2 Instance Type"
  type = string
  default = "t3.micro"
}
# AWS EC2 Instance Key Pair
variable "instance_keypair" {
  description = "AWS EC2 Key pair that need to be associated with EC2 Instance"
  type = string
  default = "terraform-key"
}
```

Step-06-02: c7-03-ec2instance-bastion.tf

- Example EC2 Instance Module for Reference

```
# AWS EC2 Instance Terraform Module
# Bastion Host - EC2 Instance that will be created in VPC Public Subnet
module "ec2_public" {
  source  = "terraform-aws-modules/ec2-instance/aws"
  version = "2.17.0"
  # insert the 10 required variables here
  name = "${var.environment}-BastionHost"
  ami = data.aws_ami.amzlinux2.id
  instance_type = var.instance_type
  key_name = var.instance_keypair
  subnet_id = module.vpc.public_subnets[0]
  vpc_security_group_ids = [module.public_bastion_sg.this_security_group_id]
  tags = local.common_tags
}
```

Step-06-03: c7-04-ec2instance-private.tf

- Example EC2 Instance Module for Reference

```
# EC2 Instances that will be created in VPC Private Subnets
module "ec2_private" {
  source  = "terraform-aws-modules/ec2-instance/aws"
  version = "2.17.0"
  name = "${var.environment}-vm"
  ami = data.aws_ami.amzlinux2.id
  instance_type = var.instance_type
  user_data = file("${path.module}/apache-install.sh")
  key_name = var.instance_keypair
  #subnet_id = module.vpc.private_subnets[0] # Single Instance
  vpc_security_group_ids = [module.private_sg.this_security_group_id]
  instance_count = 3
  subnet_ids = [
    module.vpc.private_subnets[0],
    module.vpc.private_subnets[1],
```

```
  ]
 tags = local.common_tags
}
```

Step-06-04: c7-02-ec2instance-outputs.tf

```
# AWS EC2 Instance Terraform Outputs
# Public EC2 Instances - Bastion Host
output "ec2_bastion_public_instance_ids" {
 description = "List of IDs of instances"
 value       = module.ec2_public.id
}
output "ec2_bastion_public_ip" {
 description = "List of Public ip address assigned to the instances"
 value       = module.ec2_public.public_ip
}
# Private EC2 Instances
output "ec2_private_instance_ids" {
 description = "List of IDs of instances"
 value       = module.ec2_private.id
}
output "ec2_private_ip" {
 description = "List of private ip address assigned to the instances"
 value       = module.ec2_private.private_ip
}
```

Step-07: EC2 Elastic IP for Bastion Host - c8-elasticip.tf

- learn about Terraform Resource Meta-Argument depends_on

```
# Create Elastic IP for Bastion Host
# Resource - depends_on Meta-Argument
resource "aws_eip" "bastion_eip" {
 depends_on = [module.ec2_public]
 instance =  module.ec2_public.id[0]
 vpc = true
```

```
  tags = local.common_tags
}
```

Step-08: c9-nullresource-provisioners.tf

Step-08-01: Define null resource in c1-versions.tf

- Learn about Terraform Null Resource
- Define null resource in c1-versions.tf in terraform block

```
    null = {
      source = "hashicorp/null"
      version = "~> 3.0.0"
    }
```

Step-08-02: Understand about Null Resource and Provisioners

- Learn about Terraform Null Resource
- Learn about Terraform File Provisioner
- Learn about Terraform Remote-Exec Provisioner
- Learn about Terraform Local-Exec Provisioner

```
# Create a Null Resource and Provisioners
resource "null_resource" "name" {
  depends_on = [module.ec2_public ]
  # Connection Block for Provisioners to connect to EC2 Instance
  connection {
    type = "ssh"
    host = aws_eip.bastion_eip.public_ip
    user = "ec2-user"
    password = ""
    private_key = file("private-key/terraform-key.pem")
  }

 # Copies the terraform-key.pem file to /tmp/terraform-key.pem
  provisioner "file" {
    source      = "private-key/terraform-key.pem"
    destination = "/tmp/terraform-key.pem"
```

```
  }


# Using remote-exec provisioner fix the private key permissions on Bastion Host
  provisioner "remote-exec" {
    inline = [
      "sudo chmod 400 /tmp/terraform-key.pem"
    ]
  }
  # local-exec provisioner (Creation-Time Provisioner - Triggered during Create Resource)
  provisioner "local-exec" {
    command = "echo VPC created on `date` and VPC ID: ${module.vpc.vpc_id} >> creation-time-vpc-id.txt"
    working_dir = "local-exec-output-files/"
    #on_failure = continue
  }
## Local Exec Provisioner:  local-exec provisioner (Destroy-Time Provisioner - Triggered during deletion of Resource)
  provisioner "local-exec" {
    command = "echo Destroy time prov `date` >> destroy-time-prov.txt"
    working_dir = "local-exec-output-files/"
    when = destroy
    #on_failure = continue
  }
}
```

Step-09: ec2instance.auto.tfvars

```
# EC2 Instance Variables
instance_type = "t3.micro"
instance_keypair = "terraform-key"
```

Step-10: Usage of depends_on Meta-Argument

Step-10-01: c7-04-ec2instance-private.tf

- We have put depends_on so that EC2 Private Instances will not get created until all the resources of VPC module are created

- why?
- VPC NAT Gateway should be created before EC2 Instances in private subnets because these private instances has a userdata which will try to go outbound to download the HTTPD package using YUM to install the webserver
- If Private EC2 Instances gets created first before VPC NAT Gateway provisioning of webserver in these EC2 Instances will fail.

depends_on = [module.vpc]

Step-10-02: c8-elasticip.tf

- We have put depends_on in Elastic IP resource.
- This elastic ip resource will explicitly wait for till the bastion EC2 instance module.ec2_public is created.
- This elastic ip resource will wait till all the VPC resources are created primarily the Internet Gateway IGW.

depends_on = [module.ec2_public, module.vpc]

Step-10-03: c9-nullresource-provisioners.tf

- We have put depends_on in Null Resource
- This Null resource contains a file provisioner which will copy the private-key/terraform-key.pem to Bastion Host ec2_public module created ec2 instance.
- So we added explicit dependency in terraform to have this null_resource wait till respective EC2 instance is ready so file provisioner can copy the private-key/terraform-key.pem file

depends_on = [module.ec2_public]

Step-11: Execute Terraform Commands

Terraform Initialize

terraform init

Terraform Validate

terraform validate

Terraform Plan

terraform plan

Observation:

1) Review Security Group resources

2) Review EC2 Instance resources

3) Review all other resources (vpc, elasticip)

Terraform Apply

terraform apply -auto-approve

Observation:

1) VERY IMPORTANT: Primarily observe that first VPC NAT Gateway will be created and after that only module.ec2_private related EC2 Instance will be created

Step-12: Connect to Bastion EC2 Instance and Test

Connect to Bastion EC2 Instance from local desktop

ssh -i private-key/terraform-key.pem ec2-user@<PUBLIC_IP_FOR_BASTION_HOST>

Curl Test for Bastion EC2 Instance to Private EC2 Instances

curl http://<Private-Instance-1-Private-IP>

curl http://<Private-Instance-2-Private-IP>

Connect to Private EC2 Instances from Bastion EC2 Instance

ssh -i /tmp/terraform-key.pem ec2-user@<Private-Instance-1-Private-IP>

cd /var/www/html

ls -lrta

Observation:

1) Should find index.html

2) Should find app1 folder

3) Should find app1/index.html file

4) Should find app1/metadata.html file

5) If required verify same for second instance too.

6) # Additionalyy To verify userdata passed to Instance

curl http://169.254.169.254/latest/user-data

Additional Troubleshooting if any issues

Connect to Private EC2 Instances from Bastion EC2 Instance

ssh -i /tmp/terraform-key.pem ec2-user@<Private-Instance-1-Private-IP>

cd /var/log

more cloud-init-output.log

Observation:

1) Verify the file cloud-init-output.log to see if any issues

2) This file (cloud-init-output.log) will show you if your httpd package got installed and all your userdata commands executed successfully or not

Step-13: Clean-Up

Terraform Destroy

terraform destroy -auto-approve

Clean-Up

rm -rf .terraform*

rm -rf terraform.tfstate*