* **What is terraform in AWS?**
* **Terraform** is an Infrastructure as a Code tool that allows you to create and improve infrastructure. Learn how to spin up **Terraform** instances with **AWS**. ... **Amazon Web Services** (**AWS**) is a secure cloud services platform provided by Amazon

1)CF is in json and little Complex

2)CF is Slow

3)Existing device or new devices outside CF Cannot be managed with CF

4)Detailed plan is not visible

**1)aws ,ec2,keypair,vpc,subnet, internet\_gateway, route\_table**

provider "aws"{

region = "us-east-1"

access\_key = "AKIASGURCE3BYWVT557L"

secret\_key ="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"

}

resource "aws\_instance" "role-test" {

ami = "ami-024a64a6685d05041"

instance\_type = "t2.micro"

key\_name = "deployer-key"

}

resource "aws\_key\_pair" "deployer" {

key\_name = "deployer-key"

public\_key = "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDAoJqhCcZ0XNMo2UGMFP6brkR4yDy/5iNkKwLf1rqAuAgdlHEN5uSbFXbHgKsk+ZuBRi6ncDcVsR9jIyMQiANhzYXNDosRHamneNAWfrIrNSQi1hhSKwB96eMbhBsoa6kt7C7jDfBgbvKfXa0iFjGnYVI2TogGNoN/THK7bw4xUyexSgPGmkw5Sg93/A4LNgMHLq2/MGvsXeuzM0dBmzU5UKMIhptctYkRyMp8hoGM9QvpYLKwOzORSeouyNFRWbz+wWVlFtfhV2k3VNhYFjjZD+hxXDt+YfVeLmi4IXYG+vcntX9f201+FW1Gk22XNI/LZaQqXF1R2y3QBamMYDx3 root@ip-172-31-30-152"

}

resource "aws\_vpc" "main" {

cidr\_block = "120.51.0.0/16"

instance\_tenancy = "default"

tags = {

Name = "main"

Locaction = "Hyd"

}}

resource "aws\_subnet" "main" {

vpc\_id = "${aws\_vpc.main.id}"

cidr\_block = "120.51.1.0/24"

tags = {

Name = "subnet"

}

}

resource "aws\_internet\_gateway" "main" {

vpc\_id = "${aws\_vpc.main.id}"

tags = {

Name = "main"

}

}

resource "aws\_route\_table" "r" {

vpc\_id = "${aws\_vpc.main.id}"

route {

cidr\_block = "10.0.1.0/24"

gateway\_id = "${aws\_internet\_gateway.main.id}"

subnet\_id = "${aws\_subnet.main\_id}"

}

tags = {

Name = "main"

}

}

**2)Elb and SecurityGroup ,ec2**

provider "aws"{

region = "us-east-1"

access\_key = "AKIASGURCE3BYWVT557L"

secret\_key ="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"

}

resource "aws\_security\_group" "rvg\_test" {

name = "rvg\_test"

description = "default VPC security group"

# TCP access

ingress {

from\_port = 0

to\_port = 65535

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

# HTTP access from anywhere

ingress {

from\_port = 80

to\_port = 80

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

}

resource "aws\_instance" "inbox-test" {

ami = "ami-024a64a6685d05041"

instance\_type = "t2.micro"

key\_name = "pepptytest"

}

resource "aws\_elb" "inbox-test" {

name = "rvg-inbox-test"

# The same availability zone as our instance

availability\_zones = ["us-east-1c", "us-east-1b", "us-east-1a", "us-east-1e"]

security\_groups = ["${aws\_security\_group.rvg\_test.id}"]

listener {

instance\_port = 4000

instance\_protocol = "http"

lb\_port = 80

lb\_protocol = "http"

}

listener {

instance\_port = 4000

instance\_protocol = "http"

lb\_port = 443

lb\_protocol = "https"

ssl\_certificate\_id = "arn:aws:iam::231961042942:server-certificate/cloudfront/wildcard-bounceexchange-com/wildcard-bounceexchange-com"

}

health\_check {

healthy\_threshold = 2

unhealthy\_threshold = 2

target = "HTTP:4000/ping"

interval = 10

timeout = 2

}

# The instance is registered automatically

#instances = ["${aws\_instance.inbox-test.\*.id}"]

#provisioner "local-exec" {

# command = "cd .. && ansible-playbook --inventory-file=/etc/ansible/ec2.py -l key\_rvg-inbox-test -e 'aws\_access\_key=${var.access\_key} aws\_secret\_key=${var.secret\_key}' provisioning/playbook.yml -vvvv"

#}

}

resource "aws\_eip" "ip" {

instance = "${aws\_instance.personal.id}"

}

**main.tf**

provider "aws"{

region = "us-east-1"

access\_key = "${var.aws\_access\_key}"

secret\_key ="${var.aws\_secret\_key}"

}

resource "aws\_vpc" "default" {

cidr\_block = "${var.vpc\_cider}"

enable\_dns\_hostnames = true

tags = {

Name = "${var.vpc\_name}"

}}

resource "aws\_internet\_gateway" "default" {

vpc\_id = "${aws\_vpc.default.id}"

tags = {

Name = "main"

}

}

resource "aws\_subnet" "subnet1-public" {

vpc\_id = "${aws\_vpc.default.id}"

cidr\_block = "${var.public\_subnet1\_cidr"

availability\_zone = "us-east-1a"

tags = {

Name = "${var.public\_subnet1\_name}"

}

}

resource "aws\_subnet" "subnet2-public" {

vpc\_id = "${aws\_vpc.default.id}"

cidr\_block = "${var.public\_subnet2\_cidr"

availability\_zone = "us-east-1b"

tags = {

Name = "${var.public\_subnet2\_name}"

}

}

resource "aws\_subnet" "subnet3-public" {

vpc\_id = "${aws\_vpc.default.id}"

cidr\_block = "${var.public\_subnet3\_cidr"

availability\_zone = "us-east-1c"

tags = {

Name = "${var.public\_subnet3\_name}"

}

}

resource "aws\_route\_table" "terraform-public" {

vpc\_id = "${aws\_vpc.dafault.id}"

route {

cidr\_block = "0.0.0.0/0"

gateway\_id = "${aws\_internet\_gateway.default.id}"

}

tags = {

Name = "${var.main\_Routing\_Table}"

}

}

resource "aws\_route\_table\_association" "terraform-public" {

subnet\_id = "${aws\_subnet.subnet1\_public.id}"

route\_table\_id = "${aws\_route\_table.terraform-public.id}"

}

resource "aws\_security\_group" "allow\_all" {

name = "allow\_all"

description = "Allow all inbound traffic"

vpc\_id = "${aws\_vpc.default.id}"

ingress {

from\_port = 0

to\_port = 0

protocol = "-1"

cidr blocks = {"0.0.0.0/0"}

egress {

from\_port = 0

to\_port = 0

protocol = "-1"

cidr blocks = {"0.0.0.0/0"}

}

}

resource "aws\_instance" "web-2" {

ami = "${lookup(var.amis, var.aws\_region)}"

availability\_zone = "us-east-1a"

instance\_type = "t2.micro"

key\_name = "${var.key\_name}"

subnet\_id = "{"${aws\_subnet.subnet1\_public.id}"}

vpc\_security\_group\_ids = ["${aws\_security\_group.allow\_all.id}"}

associate\_public\_ip\_address = true

tag {

name = "server\_1"

Env = "prod"

owner = "sree"

}

}

resource "aws\_instance" "web\_2"

ami = "${lookup(var.amis, var.aws\_region)}"

availability\_zone = "us-east-1a"

instance\_type = "t2.micro"

}

**Variable.tf**

variable "aws\_access\_key" {}

variable "aws\_secret\_key" {}

variable "aws\_region"{}

variable "amis"{

description = "AMIs by region"

default = {

us-east-1 = "ami-97785bed"

us-east-2 = "ami-f63b1193"

}

}

variable "private\_subnet\_cidr" {}

variable "public\_subnet1\_cidr" {}

variable "public\_subnet2\_cidr" {}

variable "public\_subnet3\_cidr" {}

variable "private\_subnet\_cidr" {}

variable main\_Routing\_Table {}

variable "azs" {

description = "run the EC2 Instance in these availability Zone"

type = "list"

default = {"is-east-1a", "us-east-lb", us-east-1c" }

}

variable"enviroment" { default = "dev" }

variable "instance\_type" {

type = "map"

default = {

dev = "t2.micro"

test = "t2.micro"

prod = "t2.medium"

}

}

**test.tf**

aws\_access\_key =" "

aws\_secret\_key = ""

aws\_region = "us-east-1"

vpc\_cidr = "10.10.0.0/16"

public\_subnet1\_cidr = "10.10.1.0/24"

public\_subnet2\_cidr = "10.10.2.0/24"

public\_subnet3\_cidr = "10.10.3.0/24"

private\_subnet\_cidr = "10.10.20.0/24"

vpc\_name = "terraform-aws-testing"

IGW\_name = "terraform-aws-igw"

public\_subnet1\_name = "terraform\_public\_subnet1\_testing"

public\_subnet2\_name = "terraform\_public\_subnet2-testing"

public-subnet3-name = "terraform-public-subnet3-testing"

private\_subnet\_name = "terraform\_private\_subnet\_testing"

main\_Routing\_Table = "terraform\_main\_table\_testing"

key\_name = "SP-Key"

environment = "dev'