IAM

1) Add one IAM user in aws without any policy

```
# cat base.tf
provider "aws" {
 region = "ap-south-1"
}

resource "aws_iam_user" "iam1" {
 name = "user1"
}
# terraform validate
# terraform plan
# terraform show
# terraform apply
```

2) Add one IAM user in aws with some predefined policy

```
# cat base.tf
provider "aws" {
 region = "ap-south-1"
}

resource "aws_iam_user" "iam1" {
 name = "user1"
}

resource "aws_iam_user_policy_attachment" "test-attach" {
  user="${aws_iam_user.iam1.name}"
  policy_arn="arn:aws:iam::aws:policy/AmazonEC2FullAccess"
}
# terraform validate
# terraform plan
# terraform apply
# terraform show
```

3) Add IAM group and add some user in it and add policy to it

```
# cat base.tf
provider "aws" {
 region = "ap-south-1"
}

resource "aws_iam_group_membership" "team" {
  name = "tf-testing-group-membership"
  users = [
    "${aws_iam_user.user_one.name}",
    "${aws_iam_user.user_two.name}",
  ]
  group = "${aws_iam_group.group.name}"
}
```

```
resource "aws_iam_group" "group" {
  name = "Developers"
}

resource "aws_iam_user" "user_one" {
  name = "user1"
}

resource "aws_iam_user" "user_two" {
  name = "user2"
}

resource "aws_iam_group_policy_attachment" "test-attach" {
  group      = "${aws_iam_group.group.name}"
  policy_arn = "arn:aws:iam::aws:policy/AmazonEC2FullAccess"
}
# terraform validate
# terraform plan
# terraform apply
# terraform show
```

4) Add a policy for ec2 “list instance access” action in mumbai region

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

resource "aws_iam_policy" "policy" {
  name        = "test_policy"
  description = "My test policy"

  policy = <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "ec2:DescribeInstances",
            "Resource": "*",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "aws:RequestedRegion": "ap-south-1"
                }
            }
        }
    ]
}
EOF
}
# terraform validate
# terraform plan
# terraform apply
# terraform show
```

5) Add policy from the template file

```
# cat policy.json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "ec2:DescribeInstances",
             "Resource": "*",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "aws:RequestedRegion": "ap-south-1"
                }
            }
        }
    ]
}
# cat base.tf
provider "aws" {
 region ="ap-south-1"
}

data "template_file" "temp1" {
  template = "${file("/home/vs/class-data/policy.json")}"
}

resource "aws_iam_policy" "policy" {
  name        = "test_policy"
  description = "My test policy"

  policy = "${data.template_file.temp1.rendered}"
}
# terraform validate
# terraform plan
# terraform apply
# terraform show
```

6) Some example of data sources

```
> to print the entire information abt the availability zones in a region
provider "aws" {
 region ="ap-south-1"
}

data "aws_availability_zones" "az" {
}

output "ov" {
  value = "${data.aws_availability_zones.az}"
}
 value = “${data.aws_availability_zones.az.keyname}” to find specific stuff
```

7) Create IAM user with terraform by reading a file and make them available

```
 ~/class-data$ --> cat user_list
user1
user2
user3
user4
user5
 ~/class-data$ --> cat base.tf
provider "aws" {
 region="ap-south-1"
}

/*
data "local_file" "file1" {
  filename = "/home/vs/class-data/user_list"
}
*/

locals {
  abc = compact(split("\n",file("/home/vs/class-data/user_list")))
}
output "new" {
 value = local.abc
}
resource "aws_iam_user" "user" {
 for_each = toset(local.abc)
 name = each.value
}
```

8) Same for ec2-instance, creating instances reading the name from the file

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

locals {
  abc = compact(split("\n",file("${var.filepath}")))
}

output "new" {
 value = local.abc
}

resource "aws_instance" "examples" {
 for_each = toset(local.abc)
 ami = "ami-0470e33cd681b2476"
 instance_type="t2.micro"
 tags = {
  Name = each.value
}
}
variable "filepath" {}
```

VPC

1) Create a vpc with 2 subnets and 1 internet gateway and route table

```
data "aws_availability_zones" "available" {}
resource "aws_vpc" "myVpc" {
  cidr_block           = "10.20.0.0/16"
  enable_dns_hostnames = true
  tags {
    Name = "myVpc"
  }
}resource "aws_subnet" "public_subnet" {
  count = "${length(data.aws_availability_zones.available.names)}"
  vpc_id = "${aws_vpc.myVpc.id}"
  cidr_block = "10.20.${10+count.index}.0/24"
  availability_zone = "$
{data.aws_availability_zones.available.names[count.index]}"
  map_public_ip_on_launch = true
  tags {
    Name = "PublicSubnet"
  }
}resource "aws_subnet" "private_subnet" {
  count = "${length(data.aws_availability_zones.available.names)}"
  vpc_id = "${aws_vpc.myVpc.id}"
  cidr_block = "10.20.${20+count.index}.0/24"
  availability_zone= "$
{data.aws_availability_zones.available.names[count.index]}"
  map_public_ip_on_launch = false
  tags {
    Name = "PrivateSubnet"
  }
```

2) Create a vpc with 2 subnets and 1 internet gateway and route table

Teamplate uploaded in Github