Terraform

Variables: vars.tf file  
**vars.tf**

Variable “AWS\_ACCESS\_KEY”{}

Variable “AWS\_SECRET\_KEY{}

Variable “AWS\_REGION”{

Default=”eu-west-1”

}

Variable “AMIS”{

type = ‘map’

default = {

us-east-1 = “ami-13be557e”

uswest-2 = “ami-06b94666”

eu-west-1 = “ami-od729a60”

}

}

**Provider.tf**provider “aws”{

Access\_key=”${var.AWS\_ACCESS\_KEY}”

Secret\_key=”$var.AWS\_SECRET\_KEY}”

Region=”${var.AWS\_REGION}”

}

Now we will create a third file name terraform.tfvars:- we are going to put the value of our variables

**terraform.tfvars**

AWS\_ACCESS\_KEY = “bsghayei”

AWS\_SECRET\_KEY= “26yshter7ainao”

AWS\_REGION = “ap-south-1”

**Instance.tf**

Resource “aws\_instance” “example”{

ami = “${lookup(var.AMIS,var.AWS\_REGION)}”

instance\_type =”t2.micro”

}

***Software Provisioning***

There are two ways to provision software on your instances

You can build your own custom AMI and bundle your software with the image

: Packer is a great tool to do this

Another way is to boot standardized AMIs, and then install the software on it you need

* Using file uploads
* Using remote exec
* Using automation tools like chef, puppet, ansible

Use “output” to display the public IP address of an AWS resource:

Resource “aws\_instance” “example”{

ami = “${lookup(var.AMIS, var.AWS\_REGION)}”

instance\_type = “t2.micro”

}

Output “ip”{

value = “${aws\_instance.example.public\_ip}”

}

You can refer to any attribute by specifying the following elements in variable:

The resource type: aws\_instance

The resource name : example

The attribute name: public\_ip

Terraform State

Terraform keeps the remote state of the infrastructure

It store it in a file called terraform.tfstate

You can keep the terraform.tfstate in version control

To configure a consul remote store, you can add a file backend.tf with the following contents:

terraform{

backend “consul”{

address = “demo.consul.io” #hostname of the consul cluster

path = “terraform/myproject”

}

}