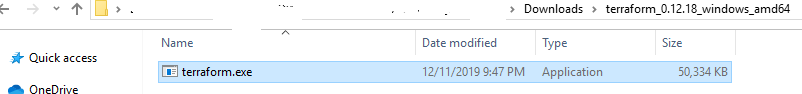
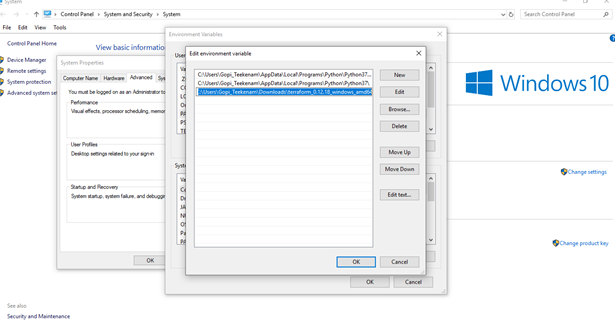
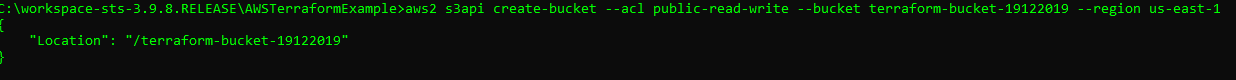
Download Terraform from **Hasicorp “**<https://www.terraform.io/downloads.html>” select the package based on the OS version

Once downloaded extract to a location and then set the path

****



Create a S3 bucket with right permissions to store terraform state



Terraform recommends enabling versioning to recover from any failure

**aws2 s3api put-bucket-versioning --bucket terraform-artifacts-bucket --versioning-configuration Status=Enabled**

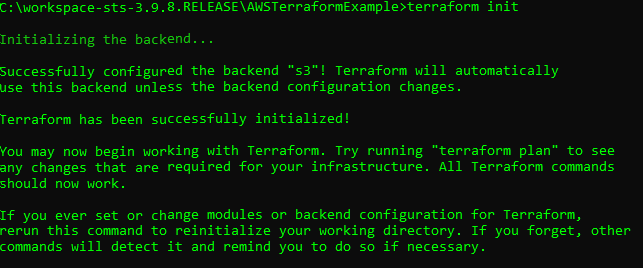
Once bucket is created, we can use the bucket for storing Terraform artifacts

Create a file with name terraform.tf in the root of the project, and then write the following script in it

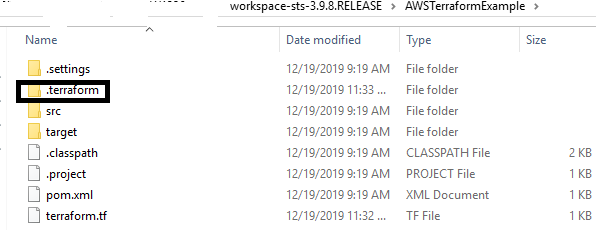
|  |
| --- |
| **terraform {**  **backend "s3" {**  **bucket = "terraform-bucket-19122019"**  **key = "cloudwatch-demo/terraform.tfstate"**  **region = "us-east-1"**  **}**  **}** |

Run the following command to initialize the backend storage

**terraform init**



As an outcome of above command “**.terraform”** folder is create in the project



Add the following script in terraform.tf file in the project

provider "aws" {

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

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

region = "${var.region}"

}

Define another file “**varilables.tf”** to provide access\_key and secret\_key

variable "access\_key" {}

variable "secret\_key" {}

variable "region" {

default = "us-east-1"

}

variable "alarms\_email" {}

create a file **“teffaform.tfvars”** with following contents

access\_key = "your-aws-access-key-here"

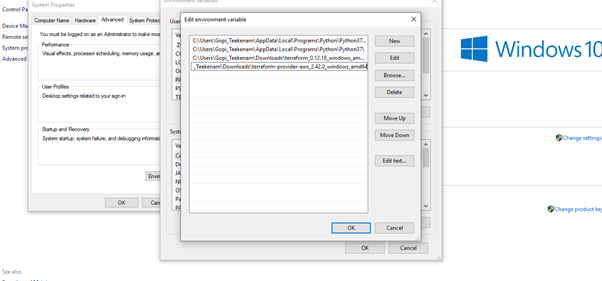
secret\_key = "your-aws-secret-key-here"

alarms\_email = "your@email.here"

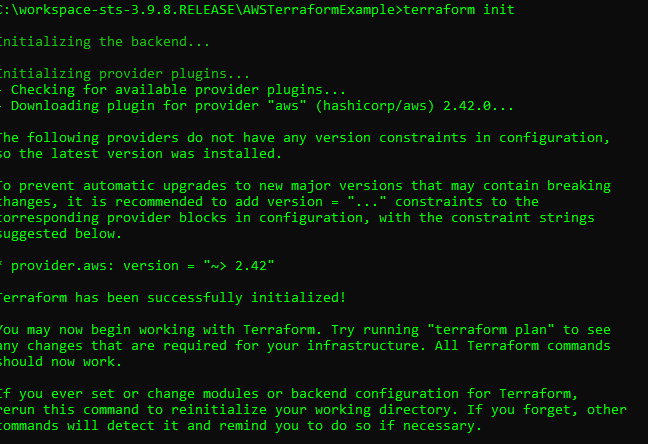
Provide you AWS access key and secret key for access\_key and secret\_key

Run terraform init

If you get an error message saying provider.aws no suitable version installed, download terraform aws provider and then configure in path variables



Run terraform.init



application.tf

|  |
| --- |
| resource "aws\_key\_pair" "ssh" {  key\_name = "default"  public\_key = "${file("~/.ssh/id\_rsa.pub")}"  }  resource "aws\_security\_group" "examplesg" {  ingress {  from\_port = 22  to\_port = 22  protocol = "tcp"  cidr\_blocks = ["0.0.0.0/0"]  }  }  resource "aws\_instance" "ec2\_instance" {  ami = "ami-00068cd7555f543d5"  instance\_type = "t2.micro"  vpc\_security\_group\_ids = ["${aws\_security\_group.examplesg.id}"]  key\_name = "${aws\_key\_pair.ssh.id}"    tags = {  Name = "my-test-server"  }  }  output "web\_public\_dns" {  value = "${aws\_instance.ec2\_instance.public\_dns}"  } |

Commands

terraform validate – To validate the file

terraform fmt – To format all the files

terraform plan – To set up the provision

terraform apply – To execute the plan

terraform destroy – To display the existing instances (need to be careful in applying this command)