Terraform exercise

GENERAL

in this exercise you will create AWS ngnix EC2 instance using terraform.

Scripts content can be found in this URL:
https://github.com/eli2983/lectures/blob/master/Terraform_excersize

PREREQUISITED

- ✓ OS: Ubuntu
- ✓ AWS Account

1. Download Terraform executable

Terraform must first be installed on your machine. Terraform is distributed as a binary package for all supported platforms and architectures.

a. **Get Terraform executable**

```
ver=0.11.7
wget https://releases.hashicorp.com/terraform/${ver}/terraform_${ver}_linux_amd64.zip
```

If a newer version exists, please replace 0.11.7 with the new one.

b. **Unzip Terraform executable**

```
ver=0.11.7
unzip terraform_${ver}_linux_amd64.zip
```

c. **Move Terraform executable to "/usr/local/bin"**

```
mv terraform /usr/local/bin
```

2. Create Terraform files (inside a dedicated folder)

a. create terraform variable file (terraform.tfvars)

replace access/secret keys and key path with your own AWS credentials:

```
cat <<EOT >> terraform.tfvars
aws_access_key = "<insert access key here>"
aws_secret_key = "<insert secret key here>"
private_key_path = "<path to private key>"
EOT
```

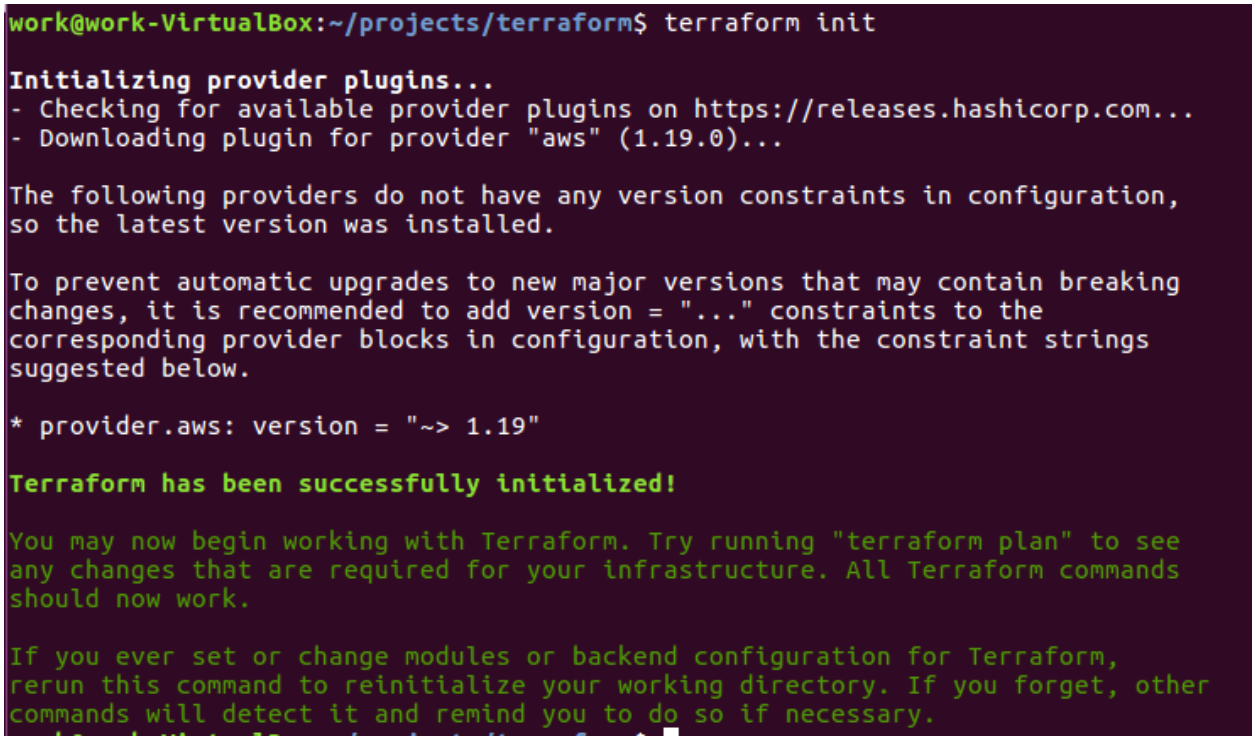
b. create terraform file (terraform.tf)

i. Replace “key_name” variable default value with your AWS account default keypair name

ii. Make sure AWS VPC default security group allows inbound SSH & HTTP/s connections

```
################################################################################
# VARIABLES
################################################################################

variable "aws_access_key" {}
variable "aws_secret_key" {}
variable "private_key_path" {}
variable "key_name" {
 default = "defaultKeys"
}

################################################################################
# PROVIDERS
################################################################################

provider "aws" {
 access_key = "${var.aws_access_key}"
 secret_key = "${var.aws_secret_key}"
 region     = "us-east-1"
}

################################################################################
# RESOURCES
################################################################################
resource "aws_instance" "nginx" {
 ami          = "ami-c58c1dd3"
 instance_type = "t2.micro"
 key_name       = "${var.key_name}"

 connection {
   user       = "ec2-user"
   private_key = "${file(var.private_key_path)}"
 }
 provisioner "remote-exec" {
   inline = [
     "sudo yum install nginx -y",
     "sudo service nginx start"
   ]
 }
}

################################################################################
# OUTPUT
################################################################################

output "aws_instance_public_dns" {
   value = "${aws_instance.nginx.public_dns}"
}
```

3. Run “terraform init”



4. Run “terraform plan” command and check the output

```
terraform plan -var-file='terraform.tfvars'
```

```
work@work-VirtualBox:~/projects/terraform$ terraform plan -var-file='terraform.tfvars'
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.


------------------------------------------------------------------------

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  + aws_instance.nginx
      id:                               <computed>
      ami:                              "ami-c58c1dd3"
      associate_public_ip_address:      <computed>
      availability_zone:                <computed>
      ebs_block_device.#:               <computed>
      ephemeral_block_device.#:         <computed>
      get_password_data:                "false"
      instance_state:                   <computed>
      instance_type:                    "t2.micro"
      ipv6_address_count:               <computed>
      ipv6_addresses.#:                 <computed>
      key_name:                         "defaultKeys"
      network_interface.#:              <computed>
      network_interface_id:             <computed>
      password_data:                    <computed>
      placement_group:                  <computed>
      primary_network_interface_id:     <computed>
      private_dns:                      <computed>
      private_ip:                       <computed>
      public_dns:                       <computed>
      public_ip:                        <computed>
      root_block_device.#:              <computed>
      security_groups.#:                <computed>
      source_dest_check:                "true"
      subnet_id:                        <computed>
      tenancy:                          <computed>
      volume_tags.%:                    <computed>
      vpc_security_group_ids.#:         <computed>


Plan: 1 to add, 0 to change, 0 to destroy.
```

5. Run “terraform apply” command and wait for “Outputs”

```
terraform apply -var-file='terraform.tfvars'
```

```
aws_instance.nginx (remote-exec): Installed:
aws_instance.nginx (remote-exec):   nginx.x86_64 1:1.12.1-1.33.amzn1

aws_instance.nginx (remote-exec): Dependency Installed:
aws_instance.nginx (remote-exec):   gperftools-libs.x86_64 0:2.0-11.5.amzn1
aws_instance.nginx (remote-exec):   libunwind.x86_64 0:1.1-10.8.amzn1

aws_instance.nginx (remote-exec): Dependency Updated:
aws_instance.nginx (remote-exec):   openssl.x86_64 1:1.0.2k-12.109.amzn1

aws_instance.nginx (remote-exec): Complete!
aws_instance.nginx (remote-exec): Starting nginx:          [  OK  ]
aws_instance.nginx: Creation complete after 53s (ID: i-00c592d327a14b698)

Apply complete! Resources: 1 added, 0 changed, 1 destroyed.

Outputs:

aws_instance_public_dns = ec2-18-207-220-207.compute-1.amazonaws.com
```

6. Browse to the new “ngnix” instance