Задача 1. Создадим бэкэнд в S3 (необязательно, но крайне желательно).

Если в рамках предыдущего задания у вас уже есть аккаунт AWS, то давайте продолжим знакомство со взаимодействием терраформа и aws.

1.Создайте s3 бакет, iam роль и пользователя от которого будет работать терраформ. Можно создать отдельного пользователя, а можно использовать созданного в рамках предыдущего задания, просто добавьте ему необходимы права, как описано здесь.

2.Зарегистрируйте бэкэнд в терраформ проекте как описано по ссылке выше.

Инициализация backend прошла успешно

```
user@ubuntu:~/devops-netology/terraform$ terraform13 init

Initializing the backend...

Initializing provider plugins...

- Using previously-installed hashicorp/aws v3.15.0



Terraform has been successfully initialized!



You may now begin working with Terraform. Try running "terraform plan" to see

any changes that are required for your infrastructure. All Terraform commands

should now work.
```

Задача 2. Инициализируем проект и создаем воркспейсы.

1.Выполните terraform init:
-если был создан бэкэнд в S3, то терраформ создат файл стейтов в S3 и запись в таблице dynamodb.
-иначе будет создан локальный файл со стейтами.
2.Создайте два воркспейса stage и prod.
3.В уже созданный aws_instance добавьте зависимость типа инстанса от вокспейса, что бы в разных ворскспейсах использовались разные instance_type.
4.Добавим count. Для stage должен создаться один экземпляр ec2, а для prod два.
5.Создайте рядом еще один aws_instance, но теперь определите их количество при помощи for_each, а не count.
6.Что бы при изменении типа инстанса не возникло ситуации, когда не будет ни одного инстанса добавьте параметр жизненного цикла create_before_destroy = true в один из рессурсов aws_instance.
7.При желании поэкспериментируйте с другими параметрами и рессурсами.

В виде результата работы пришлите:

-Вывод команды terraform workspace list.

```
11:28 user@ubuntu:~/devops-netology/terraform$ terraform13 workspace list
  default
* prod
  stage
```

-Вывод команды terraform plan для воркспейса prod.

```
11:47 user@ubuntu:~/devops-netology/terraform$ terraform13 plan
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.

data.aws_region.current: Refreshing state...
data.aws_caller_identity.current: Refreshing state...
data.aws_ami.ubuntu: Refreshing state...

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

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.panorama["t3.large"] will be created
  + resource "aws_instance" "panorama" {
      + ami                          = "ami-082e4f383a98efbe9"
      + arn                          = (known after apply)
      + associate_public_ip_address  = (known after apply)
      + availability_zone            = (known after apply)
      + cpu_core_count               = (known after apply)
      + cpu_threads_per_core         = (known after apply)
      + get_password_data            = false
      + host_id                      = (known after apply)
      + id                           = (known after apply)
      + instance_state               = (known after apply)
      + instance_type                = "t3.large"
      + ipv6_address_count           = (known after apply)
      + ipv6_addresses               = (known after apply)
      + key_name                     = (known after apply)
      + outpost_arn                  = (known after apply)
      + password_data                = (known after apply)
      + placement_group              = (known after apply)
      + primary_network_interface_id = (known after apply)
      + private_dns                  = (known after apply)
      + private_ip                   = (known after apply)
      + public_dns                   = (known after apply)
      + public_ip                    = (known after apply)
      + secondary_private_ips        = (known after apply)
      + security_groups              = (known after apply)
      + source_dest_check            = true
      + subnet_id                    = (known after apply)
      + tags                         = {
          + "Name" = "machine"
        }
      + tenancy                      = (known after apply)
      + volume_tags                  = (known after apply)
      + vpc_security_group_ids       = (known after apply)

      + ebs_block_device {
          + delete_on_termination = (known after apply)
          + device_name           = (known after apply)
          + encrypted             = (known after apply)
          + iops                  = (known after apply)
          + kms_key_id            = (known after apply)
          + snapshot_id           = (known after apply)
          + volume_id             = (known after apply)
          + volume_size           = (known after apply)
          + volume_type           = (known after apply)
        }

      + ephemeral_block_device {
          + device_name  = (known after apply)
          + no_device    = (known after apply)
          + virtual_name = (known after apply)
        }

      + metadata_options {
          + http_endpoint               = (known after apply)
          + http_put_response_hop_limit = (known after apply)
          + http_tokens                 = (known after apply)
        }

      + network_interface {
          + delete_on_termination = (known after apply)
          + device_index          = (known after apply)
          + network_interface_id  = (known after apply)
```

```
      }

    + root_block_device {
        + delete_on_termination = (known after apply)
        + device_name           = (known after apply)
        + encrypted             = (known after apply)
        + iops                  = (known after apply)
        + kms_key_id            = (known after apply)
        + volume_id             = (known after apply)
        + volume_size           = (known after apply)
        + volume_type           = (known after apply)
      }
  }

# aws_instance.panorama["t3.micro"] will be created
+ resource "aws_instance" "panorama" {
    + ami                          = "ami-082e4f383a98efbe9"
    + arn                          = (known after apply)
    + associate_public_ip_address  = (known after apply)
    + availability_zone            = (known after apply)
    + cpu_core_count               = (known after apply)
    + cpu_threads_per_core         = (known after apply)
    + get_password_data            = false
    + host_id                      = (known after apply)
    + id                           = (known after apply)
    + instance_state               = (known after apply)
    + instance_type                = "t3.micro"
    + ipv6_address_count           = (known after apply)
    + ipv6_addresses               = (known after apply)
    + key_name                     = (known after apply)
    + outpost_arn                  = (known after apply)
    + password_data                = (known after apply)
    + placement_group              = (known after apply)
    + primary_network_interface_id = (known after apply)
    + private_dns                  = (known after apply)
    + private_ip                   = (known after apply)
    + public_dns                   = (known after apply)
    + public_ip                    = (known after apply)
    + secondary_private_ips        = (known after apply)
    + security_groups              = (known after apply)
    + source_dest_check            = true
    + subnet_id                    = (known after apply)
    + tags                         = {
        + "Name" = "machine"
      }
    + tenancy                      = (known after apply)
    + volume_tags                  = (known after apply)
    + vpc_security_group_ids       = (known after apply)

    + ebs_block_device {
        + delete_on_termination = (known after apply)
        + device_name           = (known after apply)
        + encrypted             = (known after apply)
        + iops                  = (known after apply)
        + kms_key_id            = (known after apply)
        + snapshot_id           = (known after apply)
        + volume_id             = (known after apply)
        + volume_size           = (known after apply)
        + volume_type           = (known after apply)
      }
```