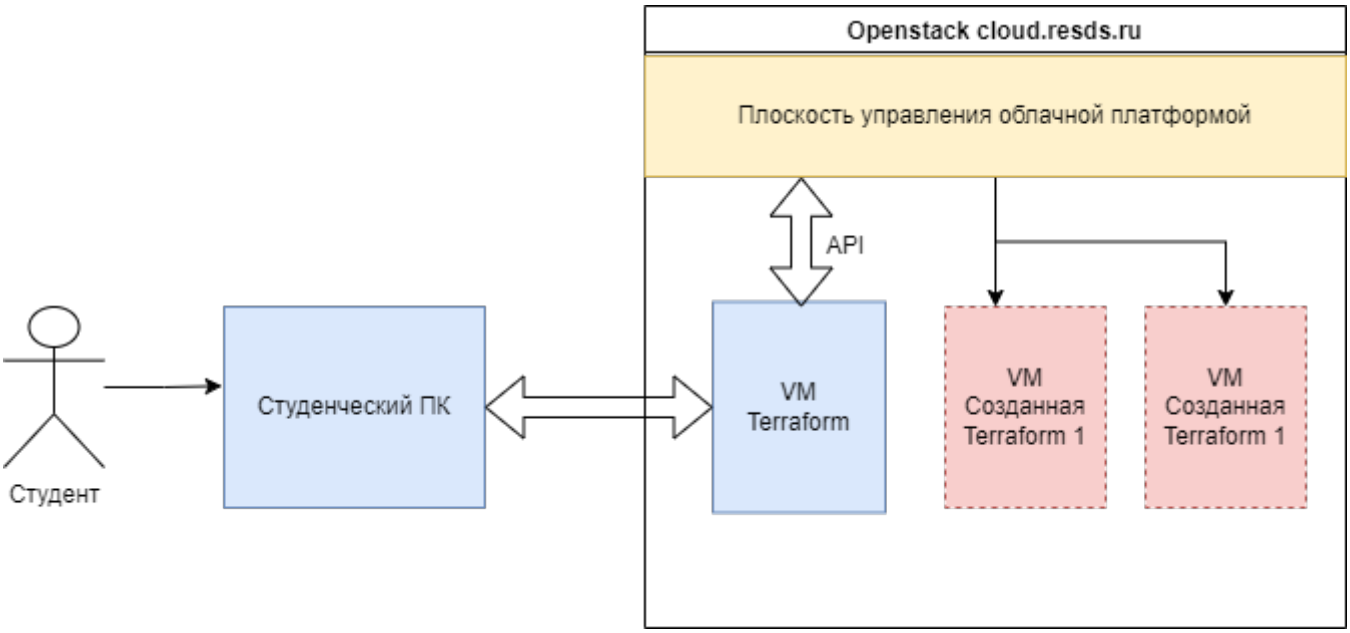
Управление инфраструктурой. Terraform.

Описание стенда



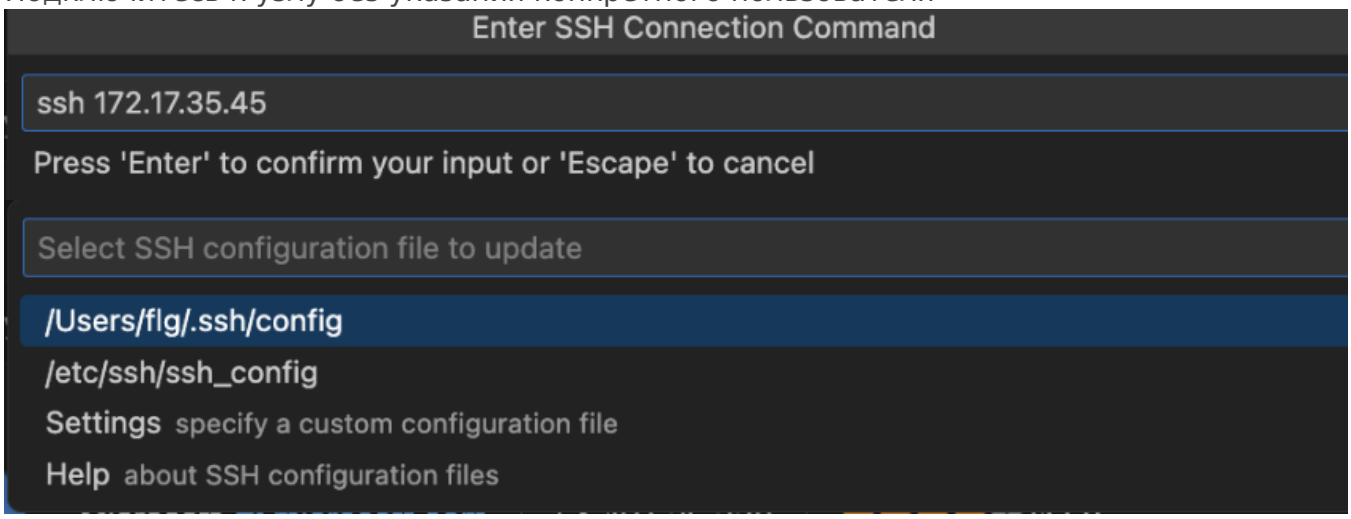
Подготовка окружения

1. Установите vscode
2. Установите плагин Remote SSH
3. Создайте VM для роли управления terraform
4. Добавить конфигурацию узла в VScode с ранее созданным узлом
   Если при подключении запрашивается пароль, то можно модифицировать конфигурацию ssh(`~/.ssh/config`)

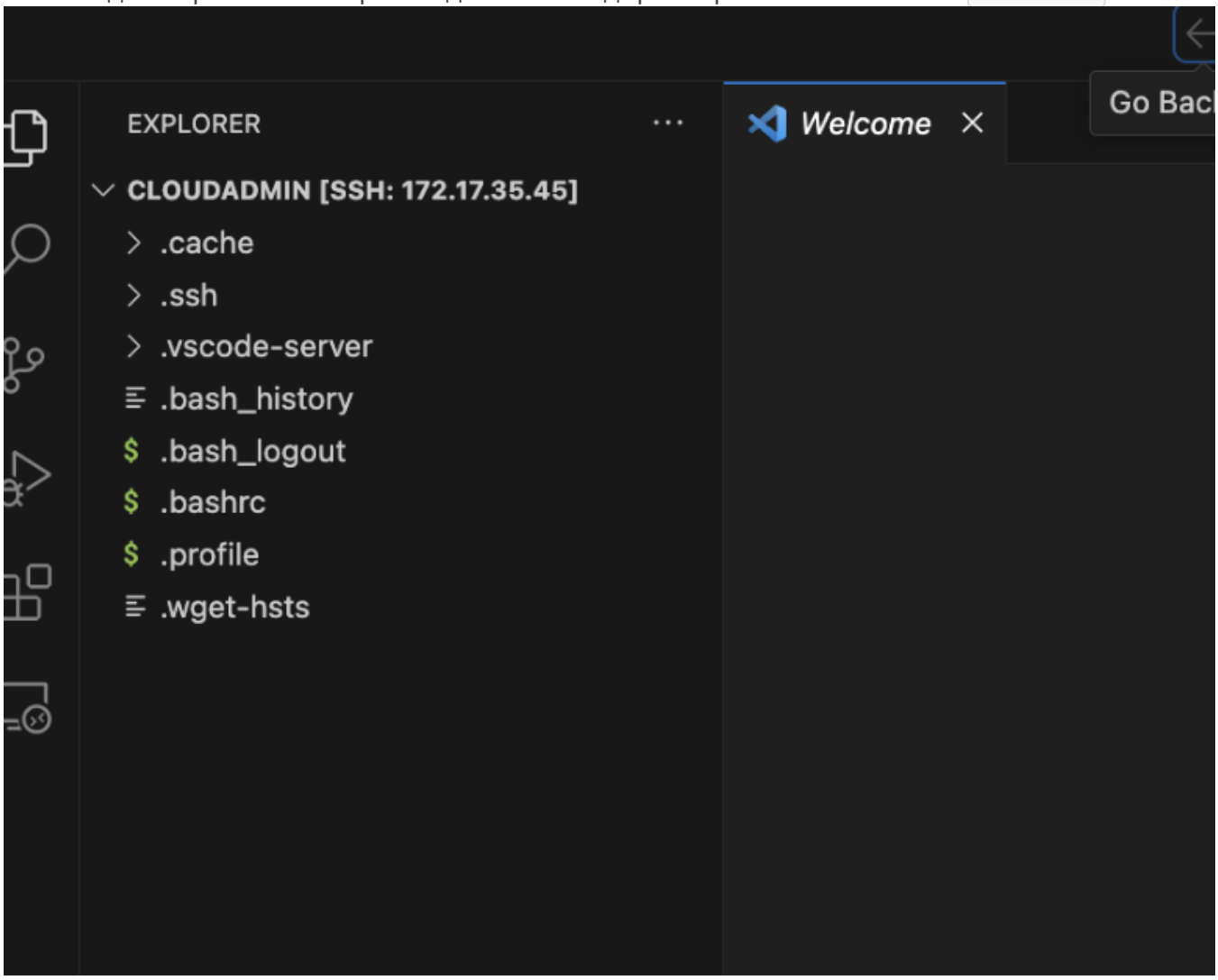
```
Host 172.17.*
  User cloudadmin
  IdentityFile ~/.ssh/id_ed25519
```

> В `IdentityFile` нужно вписать путь до вашего приватного ключа

5. Подключитесь к узлу без указания конкретного пользователя



6. Во вкладке с файлами откройте домашнюю директорию пользователя `cloudadmin`



7. Установите на узле плагин terraform от hashicorp

Установка terraform

1. Перейти на сайт: https://developer.hashicorp.com/terraform/install
2. Убеждаемся, что HashiCorp не очень любит Россию и либо пользуемся зеркалом https://cloud.vk.com/docs/manage/tools-for-using-services/terraform/quick-start (первый пункт)

   ```
   curl -l -O https://hashicorp-releases.mcs.mail.ru/terraform/1.7.4/terraform_1.7.4_linux_amd64.zip
   ```

   либо скачиваем с google диска

   ```
   pip install gdown
   gdown https://drive.google.com/uc?id=1W6z0_-DDcEobFHoNz4lErO1wMHF2wnyc
   ```

3. Разархивируем

```
sudo unzip terraform_1.7.4_linux_amd64.zip  -d /usr/local/bin
```

4. Проверим, что все установилось

Подготовка terraform

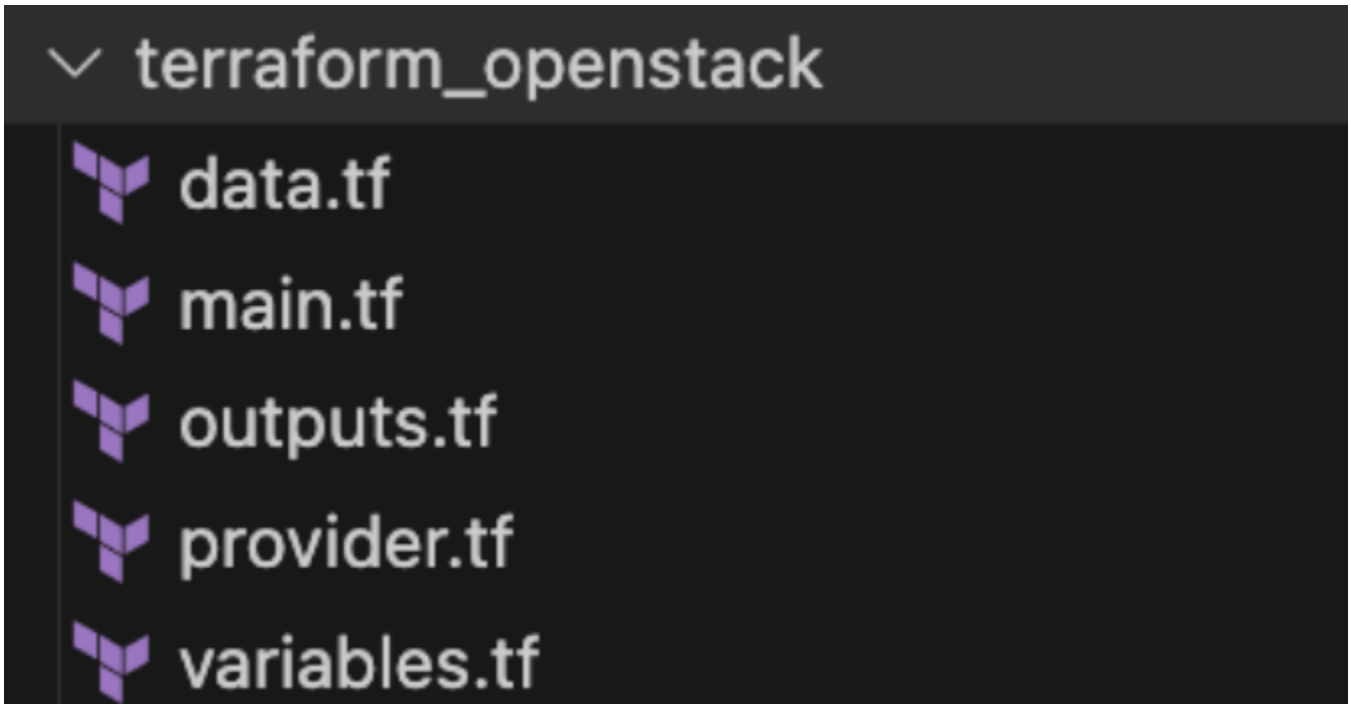
1. Добавим зеркало для работы с terraform без vpn
   Создаем файл `.terraformrc`

   ```
   touch ~/. terraformrc
   ```

   Заполним содержимое:

   ```
   provider_installation {
     network_mirror {
       url = "https://terraform-mirror.mcs.mail.ru"
       include = ["registry.terraform.io/*/*"]
     }
     direct {
       exclude = ["registry.terraform.io/*/*"]
     }
   }
   ```

2. Создать директорию `terraform_openstack` и в ней файлы: `data.tf`, `main.tf`, `outputs.tf`, `provider.tf`, `variables.tf`



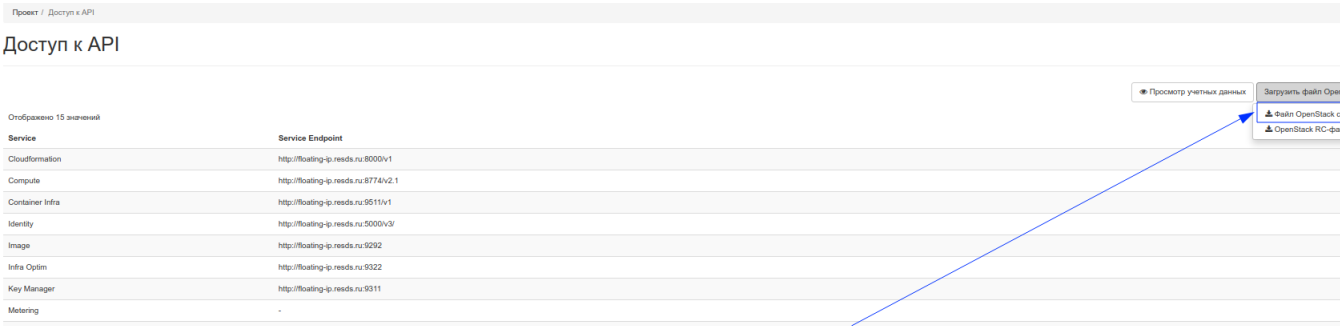
3. Заполнить `provider.tf`

```
terraform {
 required_version = ">= 0.14.0"
   required_providers {
     openstack = {
       source  = "terraform-provider-openstack/openstack"
       version = "~> 1.53.0"
     }
   }
}
```

4. Выполнить команду `terraform init` и убедиться что все прошло успешно

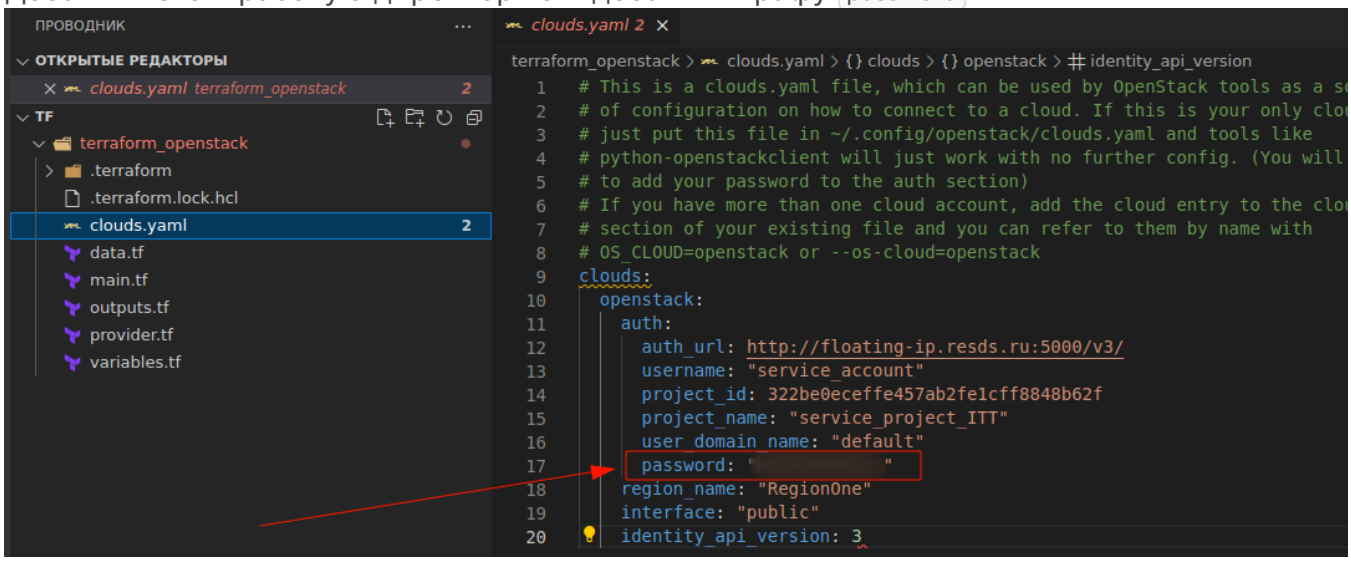
```
cloudadmin@test:~/terraform_openstack$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding terraform-provider-openstack/openstack versions matching "~> 1.53.0"...
- Installing terraform-provider-openstack/openstack v1.53.0...
- Installed terraform-provider-openstack/openstack v1.53.0 (unauthenticated)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

  Warning: Incomplete lock file information for providers

  Due to your customized provider installation methods, Terraform was forced to calculate lock file checksums locally for the following providers:
    - terraform-provider-openstack/openstack

  The current .terraform.lock.hcl file only includes checksums for linux_amd64, so Terraform running on another platform will fail to install these providers.

  To calculate additional checksums for another platform, run:
    terraform providers lock -platform=linux_amd64
  (where linux_amd64 is the platform to generate)

Terraform has been successfully initialized!
```

5. Перейти в openstack в меню API ACCESS



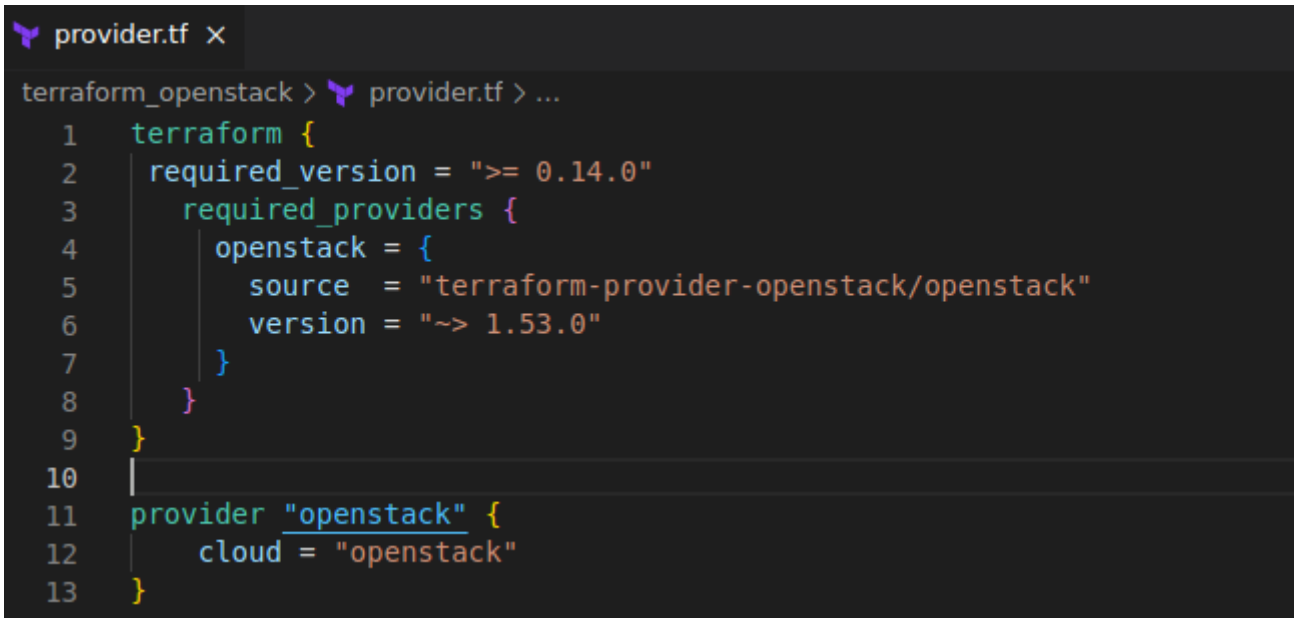
Скачайте `clouds.yml`

Добавить его в рабочую директорию и добавить графу `password`



6. в `provider.tf` добавьте директиву

```
provider "openstack" {
    cloud = "openstack"
}
```



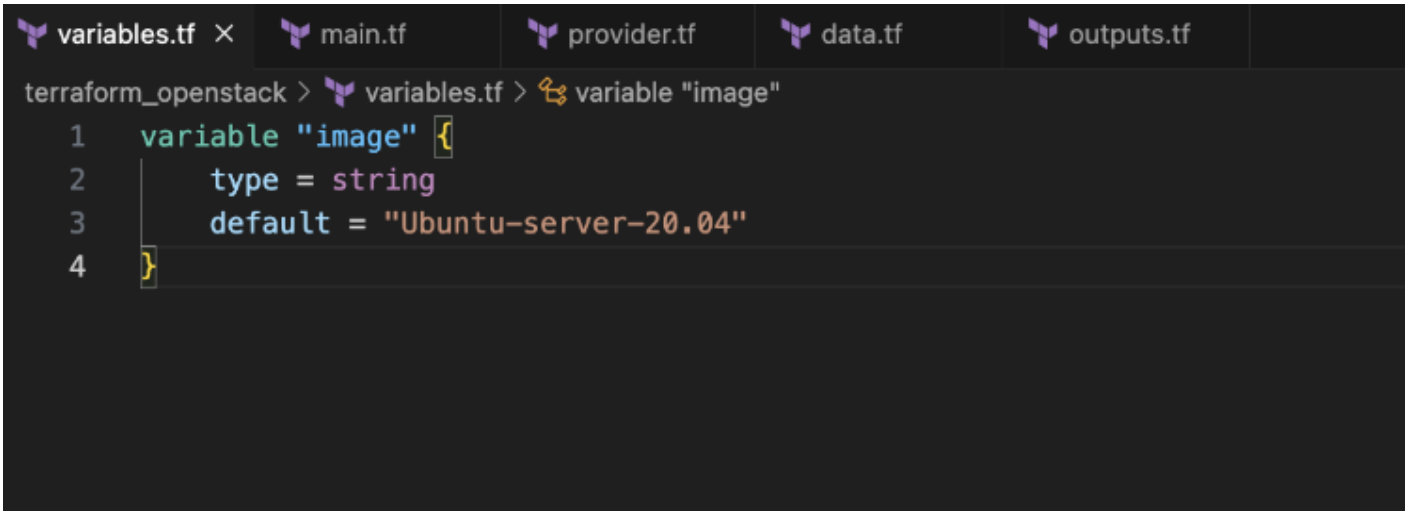
Задание 1.

Чтобы создать виртуальную машину, нам нужен какой-то образ, в openstack образы можно называть одинаково, но вот id образов будет разный, необходимо получить id необходимого нам образа Для этого пользуемся документацией(она работает тоже только с vpn или прокси) https://registry.terraform.io/providers/terraform-provider-openstack/openstack/latest/docs/data-sources/images_image_v2

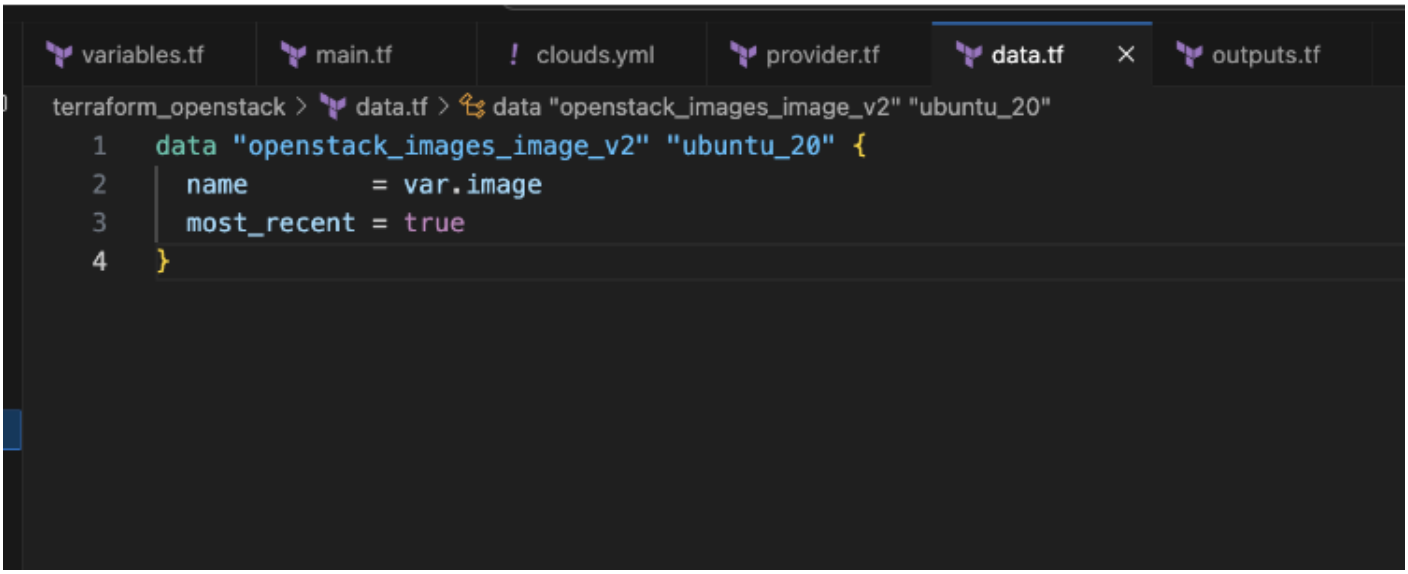
В моем случае я буду пользоваться Ubuntu-server-20.04

Суть задания, добавить переменную с названием образа, создать data source образа и вывести его id в outputs

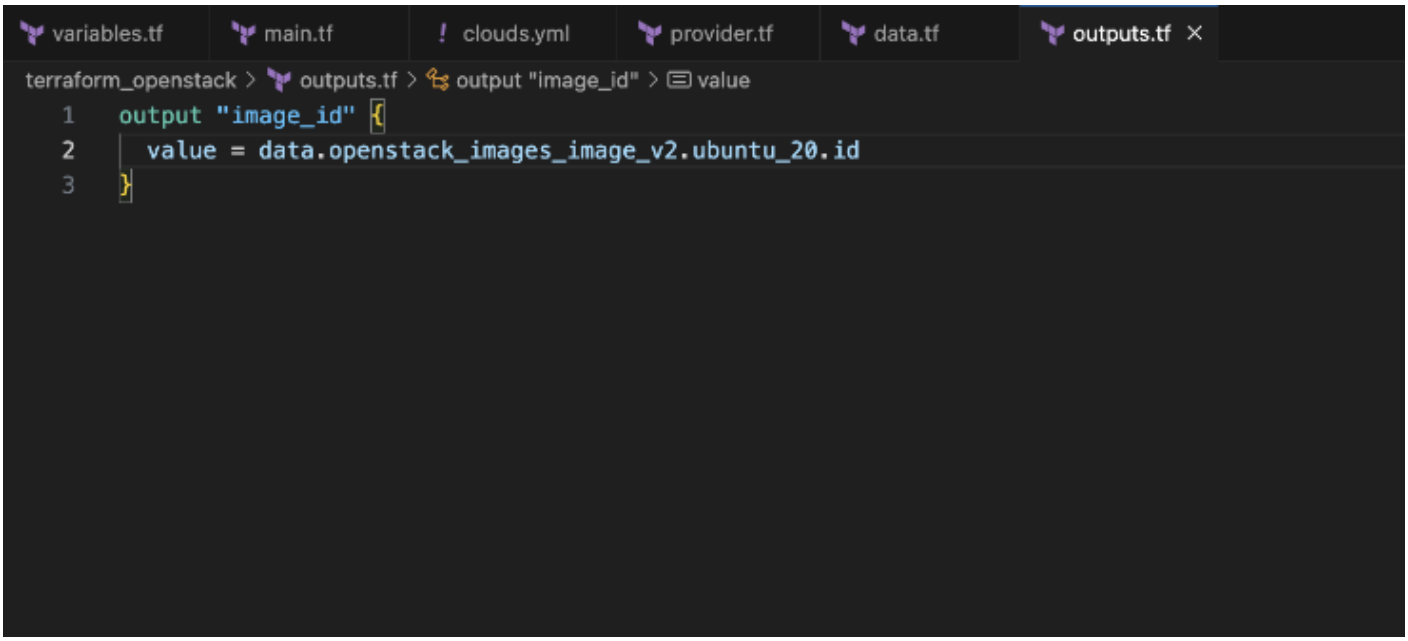
Пример:




```
variable "image" {
  type = string
  default = "Ubuntu-server-20.04"
}
```




```
data "openstack_images_image_v2" "ubuntu20" {
  name = var.image
  most_recent = true
}
```




```
output "image_id" {
   value = data.openstack_images_image_v2.ubuntu20.id
}
```

Выполнить `terraform plan` и убедиться что именно это мы и хотели создать

```
~/tf/terraform_openstack terraform plan
data.openstack_images_image_v2.ubuntu20: Reading...
data.openstack_images_image_v2.ubuntu20: Read complete after 1s [id=612b280c-3667-448f-9cd1-2dc55306d8a9]

Changes to Outputs:
  + image_id = "612b280c-3667-448f-9cd1-2dc55306d8a9"

You can apply this plan to save these new output values to the Terraform state, without changing any real infrastructure.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
```

Ну и выполняем(`terraform apply`):

```
~/tf/terraform_openstack terraform apply
data.openstack_images_image_v2.ubuntu20: Reading...
data.openstack_images_image_v2.ubuntu20: Read complete after 1s [id=612b280c-3667-448f-9cd1-2dc55306d8a9]

Changes to Outputs:
  + image_id = "612b280c-3667-448f-9cd1-2dc55306d8a9"

You can apply this plan to save these new output values to the Terraform state, without changing any real infrastructure.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes


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

Outputs:

image_id = "612b280c-3667-448f-9cd1-2dc55306d8a9"
```

Задание 2

Создать 2 виртуальные машины. Ссылка: https://registry.terraform.io/providers/terraform-provider-openstack/openstack/latest/docs/resources/compute_instance_v2

Требования:

1. Ресурсы создать в main.tf
2. Flavor должен быть small
3. Id образа не вписывать в ручную
4. У машин должен быть разный образ, но обе должны быть ubuntu
5. У машин должен быть разный объем диска
6. Outputs должен быть вида: имя машины: IP адрес машины, дополнительные сведения можно добавить по желанию
7. В переменные необходимо занести: объем диска, имя машин
8. Убедиться, что к машинам можно подключиться по ssh

Задание 3:

Удалить все что создали с помощью terraform

Версия #5
Тарабанов Илья Федорович создал 12 марта 2024 14:41:44
Тарабанов Илья Федорович обновил 16 мая 2024 20:22:24