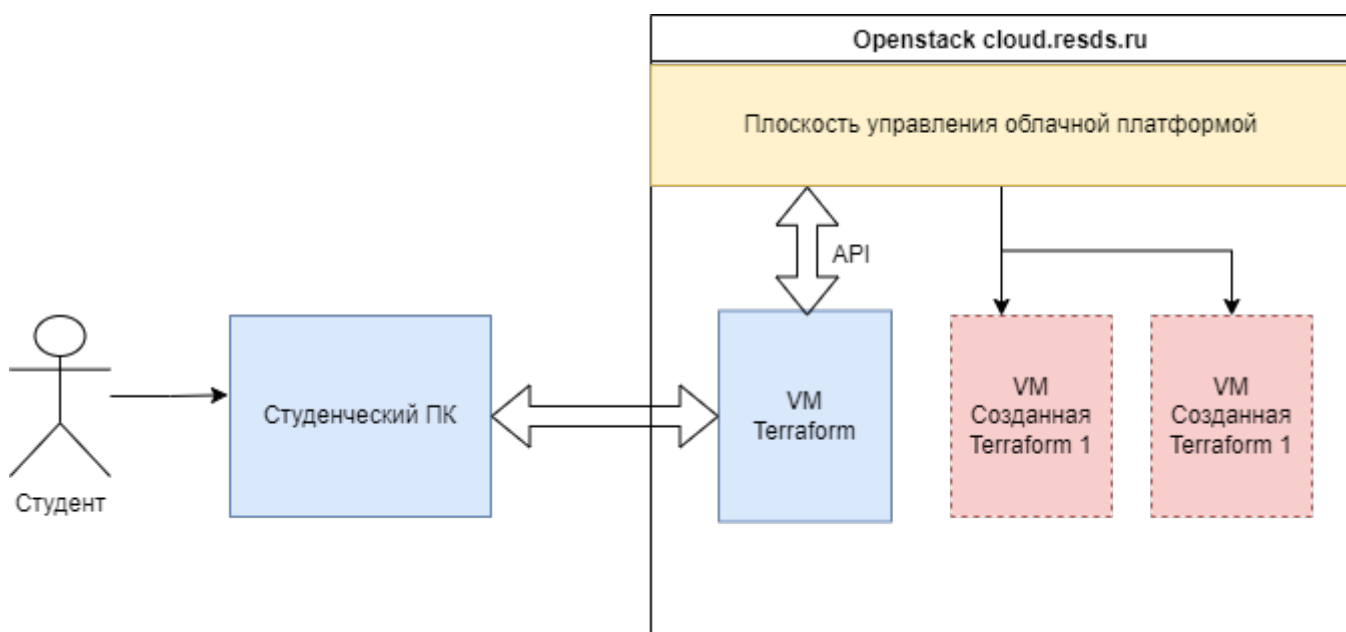


Управление инфраструктурой. Terraform.

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

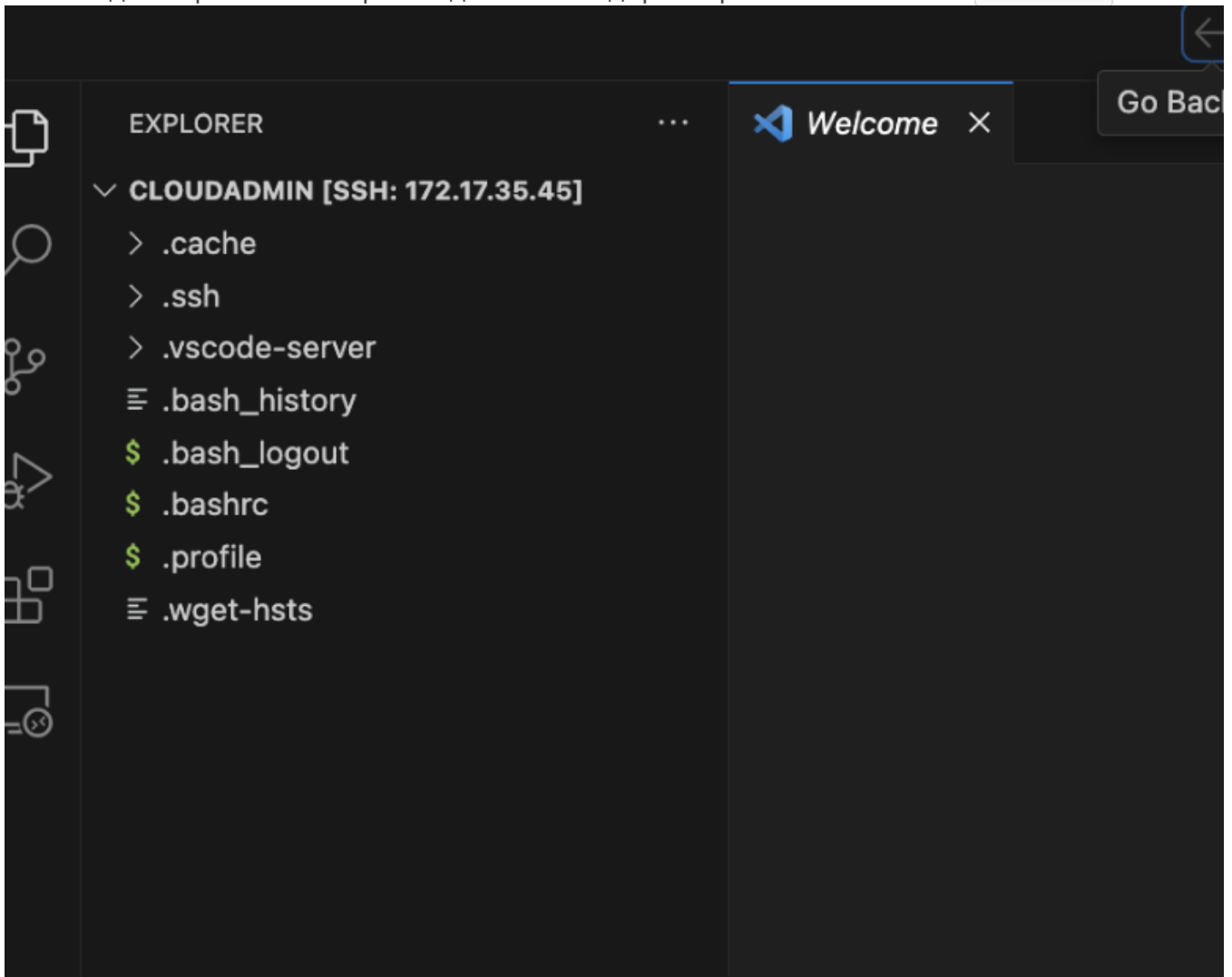
IdentityFile ~/.ssh/id_ed25519

В IdentityFile нужно вписать путь до вашего приватного ключа

5. Подключитесь к узлу без указания конкретного пользователя

```
Enter SSH Connection Command
ssh 172.17.35.45
Press 'Enter' to confirm your input or 'Escape' to cancel
Select SSH configuration file to update
/Users/flg/.ssh/config
/etc/ssh/ssh_config
Settings specify a custom configuration file
Help about SSH configuration files
```

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

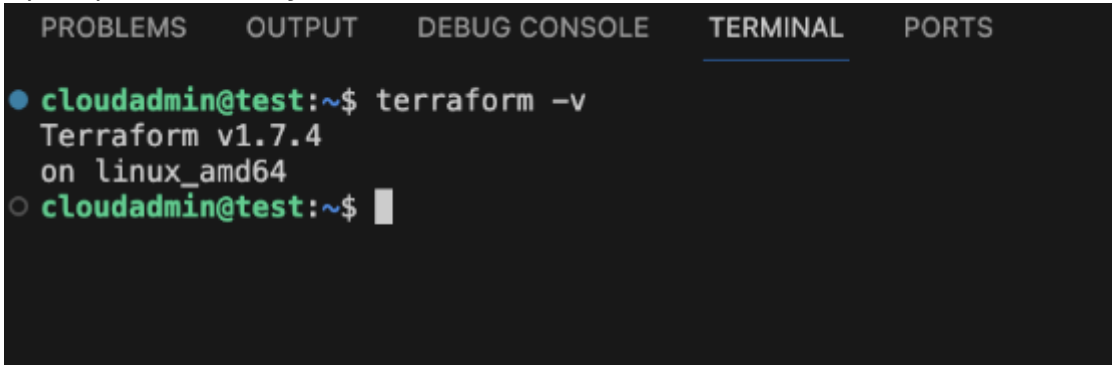
либо скачиваем с google диска

```
pip install gdown  
gdown https://drive.google.com/uc?id=1W6z0_-DDcEobFH0Nz4IErO1wMHF2wnyc
```

3. Разархивируем

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

4. Проверим, что все установилось



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
● cloudadmin@test:~$ terraform -v
Terraform v1.7.4
on linux_amd64
○ cloudadmin@test:~$
```

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`

✓ terraform_openstack

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 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**

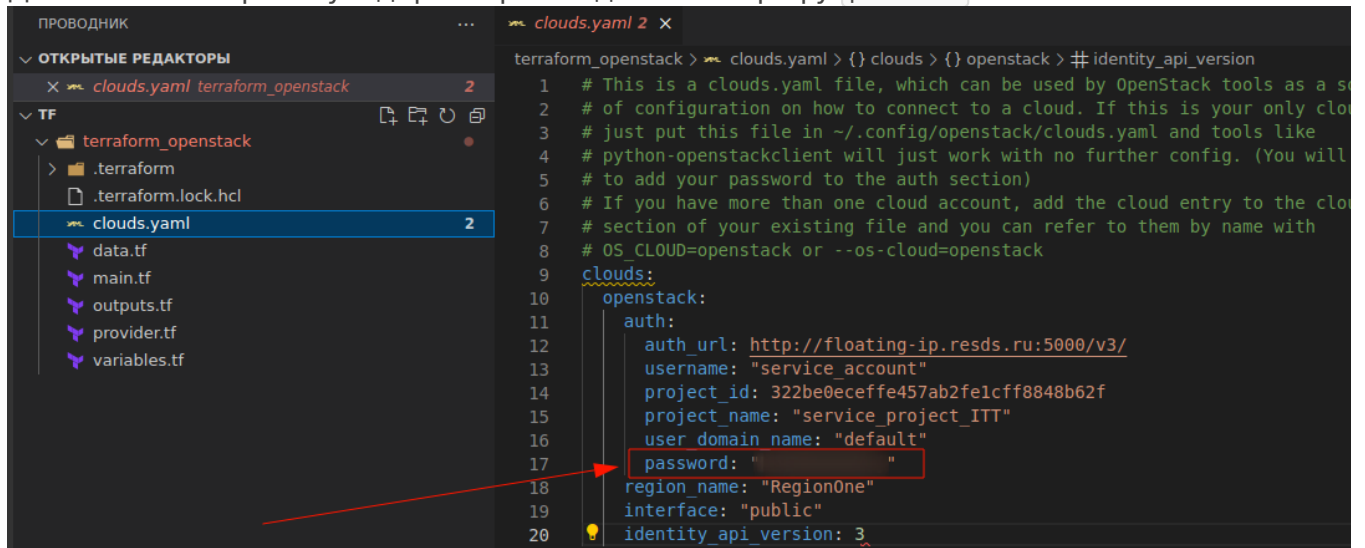
Проект / Доступ к API

Доступ к API

Отображено 15 значений		Просмотр учетных данных	Загрузить файл OpenStack с...
Service	Service Endpoint		
CloudFormation	http://floating-ip.needs.ru:8000/v1		
Compute	http://floating-ip.needs.ru:8774/v2.1		
Container Infra	http://floating-ip.needs.ru:9511/v1		
Identity	http://floating-ip.needs.ru:5000/v3		
Image	http://floating-ip.needs.ru:9292		
Infra Optim	http://floating-ip.needs.ru:9322		
Key Manager	http://floating-ip.needs.ru:9311		
Metering	-		

Скачайте `clouds.yml`

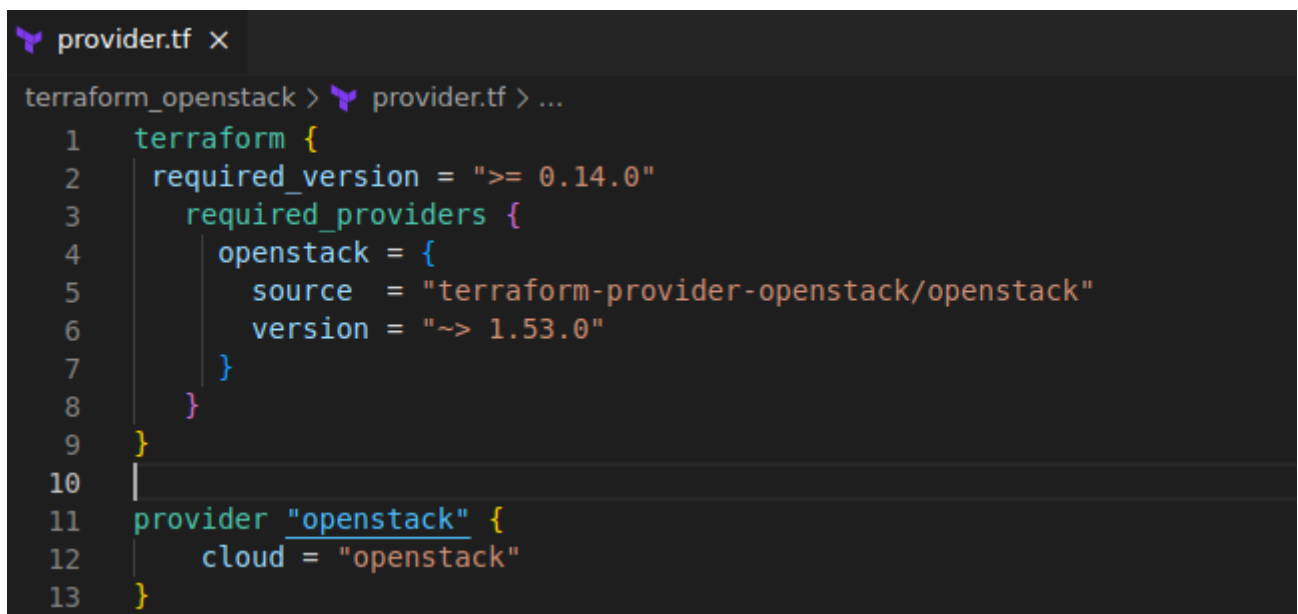
Добавить его в рабочую директорию и добавить графу `password`



```
terraform_openstack > clouds.yml > {} clouds > {} openstack > # identity_api_version
1 # This is a clouds.yml file, which can be used by OpenStack tools as a source
2 # of configuration on how to connect to a cloud. If this is your only cloud
3 # just put this file in ~/.config/openstack/clouds.yml and tools like
4 # python-openstackclient will just work with no further config. (You will
5 # to add your password to the auth section)
6 # If you have more than one cloud account, add the cloud entry to the clouds
7 # section of your existing file and you can refer to them by name with
8 # OS_CLOUD=openstack or --os-cloud=openstack
9 clouds:
10   openstack:
11     auth:
12       auth_url: http://floating-ip.resds.ru:5000/v3/
13       username: "service_account"
14       project_id: 322be0ecef457ab2fe1cff8848b62f
15       project_name: "service_project_ITT"
16       user_domain_name: "default"
17       password: ""
18     region_name: "RegionOne"
19     interface: "public"
20     identity_api_version: 3
```

6. в `provider.tf` добавьте директиву

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



```
terraform_openstack > provider.tf > ...
1 terraform {
2   required_version = ">= 0.14.0"
3   required_providers {
4     openstack = {
5       source = "terraform-provider-openstack/openstack"
6       version = "~> 1.53.0"
7     }
8   }
9 }
10
11 provider "openstack" {
12   cloud = "openstack"
13 }
```

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

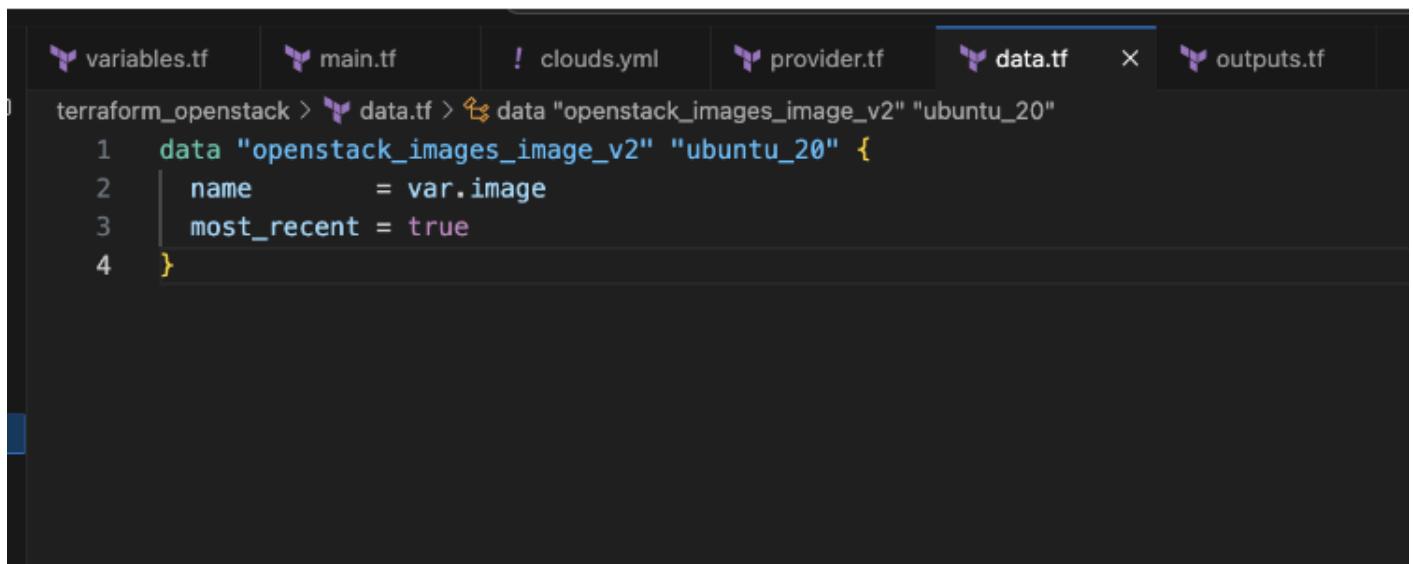
Суть задания, добавить переменную с названием образа, создать data source образа и вывести его id в outputs

Пример:



```
terraform_openstack > variables.tf > variable "image"
1  variable "image" {
2      type = string
3      default = "Ubuntu-server-20.04"
4  }
```

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



```
terraform_openstack > data.tf > data "openstack_images_image_v2" "ubuntu_20"
1  data "openstack_images_image_v2" "ubuntu_20" {
2      name          = var.image
3      most_recent    = true
4  }
```

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

```
variables.tf  main.tf  ! clouds.yml  provider.tf  data.tf  outputs.tf X
terraform_openstack > outputs.tf > output "image_id" > value
1  output "image_id" {
2    value = data.openstack_images_image_v2.ubuntu_20.id
3  }
```

```
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