Подготовка облачной инфраструктуры

Задание:

- a) Требования к виртуальным машинам:
  - 1. Основные характеристики:
    - i. Операционная система: Альт p10 StarterKit/~~Альт Сервер p10-cloud~~
    - ii. Количество vCPU: 1.
    - iii. Объём оперативной памяти: 1024 МБ.
    - iv. Объём диска: 10 ГБ/~~30 ГБ~~
    - ~~v. Тип диска: HDD.~~
- b) Подготовьте сценарий автоматизации развёртывания облачной инфраструктуры:
  - 1. Создание виртуальных машин и сетей:
    - i. Виртуальные машины и сети должны быть созданы строго в соответствии с предложенной топологией (см. Топология ниже).
    - ii. Имена виртуальных машин, сетей, подсетей и маршрутизаторов должны соответствовать именованиям, указанным в Топологии.
    - iii. Обеспечьте правильное подключение виртуальных машин к соответствующим сетям в рамках заданной топологии.
- 2. Безопасность и доступ:
  - i. Разрешите трафик по протоколу ICMP для всех виртуальных машин для диагностики сетевых подключений.
  - ii. Назначьте IP-адреса всем машинам. Сохраните внешние IP-адреса всех машин в файле /home/altlinux/white.ip на машине ControlVM.
  - iii. Настройте аутентификацию на основе открытых ключей для SSH.
  - iv. В случае предоставления внешнего доступа к виртуальным машинам, разрешите его только по протоколу SSH (публичный ключ, пароль отключён) и только с соответствующих IP-адресов.
- 3. Балансировка нагрузки:
  - i. Создайте балансировщик нагрузки и распределите трафик между серверами Web1 и Web2 (см. Топология).
  - ii. Ограничьте внешний доступ к балансировщику только протоколами HTTP и HTTPS. Все остальные порты должны быть закрыты.
  - iii. Балансировка нагрузки должна использовать алгоритм round robin.
  - iv. При обращении на внешний адрес балансировщика нагрузки должен выводиться ответ от приложения, работающего на внутренних серверах Web1 и Web2.
- 4. Настройка подключения:
  - i. Настройте машину WebAdm так, чтобы она могла подключаться по SSH с использованием пользователя altlinux и пароля «P@ssw0rd» к серверам Web1 и Web2 с помощью VPN туннеля.
  - ii. Убедитесь, что машина ControlVM может подключаться к машине WebAdm используя ключевую пару пользователя altlinux по SSH через её глобальный IP-адрес.

Вариант реализации:

ControlVM

Все файлы создаются в контексте каталога /home/altlinux/bin, если не сказано иное

- Создаём файл **network.tf** и описываем **последовательно** сетевую часть для развёртывания данной инфраструктуры с **поэтапным запуском** и **наблюдением** созданных ресурсов:

```
vim network.tf
```

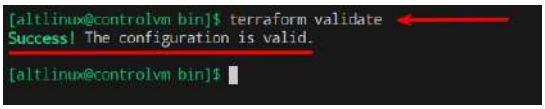
  - Помещаем следующее содержимое:
    - Создадим виртуальную сеть с именем **INTERNET** в соответствие с топологией;
    - В созданной виртуальной сети создадим одноимённую подсеть, т.к. все сетевые параметры задаются на подсеть в рамках сети (с произвольными параметрами);
      - В данном случае подсеть **INTERNET**имеет IP-адрес 192.168.200.0/24, DHCP-пул адресов для раздачи из данной подсети с 192.168.200.100 по 192.168.200.200, в качестве шлюза по умолчанию в данной подсети будет использоваться IP-адрес 192.168.200.1, в качестве DNS IP-адрес 77.88.8.8;
    - Также данную подсеть **INTERNET** необходимо добавить в ранее созданный виртуальный маршрутизатор с именем **cloud** (чтобы не создавать новый), который необходим для работы **ControlVM** и не должен быть удалён по завершению работы, для того чтобы для всех подключённых инстансов к данной подсети был доступ в сеть Интернет, т.к. на данном виртуальном маршрутизаторе ранее был выставлен чек-бокс для **SNAT**, и именно через него ControlVM и все остальные будут иметь доступ в сеть Интернет;

```
# Обращаемся к источнику данных, чтобы узнать ID существующего виртуального маршрутизатора
data "openstack_networking_router_v2" "router" {
  name = "cloud"
}

# Создаём сеть с именем "INTERNET" в соответствие с топологией
resource "openstack_networking_network_v2" "network" {
  name           = "INTERNET"
  admin_state_up = "true"
}

# Создаём подсеть с именем "INTERNET" в ранее созданной сети с именем "INTERNET"
resource "openstack_networking_subnet_v2" "subnet" {
  name            = "INTERNET"
  network_id      = openstack_networking_network_v2.network.id
  cidr            = "192.168.200.0/24"
  ip_version      = 4
  gateway_ip      = "192.168.200.1"
  dns_nameservers = [ "77.88.8.8" ]
  enable_dhcp     = true

  allocation_pool {
    start = "192.168.200.100"
    end   = "192.168.200.200"
  }
}

# Добавляем в существующий маршрутизатор созданную подсеть с именем "INTERNET", чтобы в дальнейшем ВМ
подключённые к данной подсети имели доступ в Интернет
resource "openstack_networking_router_interface_v2" "router_interface" {
    router_id = data.openstack_networking_router_v2.router.id
    subnet_id = openstack_networking_subnet_v2.subnet.id
}
```

- Выполняем проверку синтаксиса и структуры файлов конфигурации Terraform

```
terraform validate
```

  - Результат:



- Проверяем конфигурацию перед развёртыванием, смотрим план развёртывания ресурсов:

```
terraform plan
```

  - Результат:
    - По плану будет создано (добавлено) **3** ресурса: Сеть **INTERNET**, Подсеть **INTERNET** и добавление в виртуальный маршрутизатор **cloud** подсети **INTERNET**:

```
[altlinux@controlvm bin]$ terraform plan
data.openstack_networking_router_v2.router: Reading...
data.openstack_networking_router_v2.router: Read complete after 2s [id=ba5d5a05-064a-4556-bab4-020266cdba07]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # openstack_networking_network_v2.network will be created
  + resource "openstack_networking_network_v2" "network" {
      + admin_state_up          = true
      + all_tags                = (known after apply)
      + availability_zone_hints = (known after apply)
      + dns_domain              = (known after apply)
      + external                = (known after apply)
      + id                      = (known after apply)
      + mtu                     = (known after apply)
      + name                    = "INTERNET"
      + port_security_enabled   = (known after apply)
      + qos_policy_id           = (known after apply)
      + region                  = (known after apply)
      + shared                  = (known after apply)
      + tenant_id               = (known after apply)
      + transparent_vlan        = (known after apply)

      + segments (known after apply)
    }

  # openstack_networking_router_interface_v2.router_interface will be created
  + resource "openstack_networking_router_interface_v2" "router_interface" {
      + force_destroy = false
      + id            = (known after apply)
      + port_id       = (known after apply)
      + region        = (known after apply)
      + router_id     = "ba5d5a05-064a-4556-bab4-020266cdba07"
      + subnet_id     = (known after apply)
    }

  # openstack_networking_subnet_v2.subnet will be created
  + resource "openstack_networking_subnet_v2" "subnet" {
      + all_tags          = (known after apply)
      + cidr              = "192.168.200.0/24"
      + dns_nameservers   = [
          + "77.88.8.8",
        ]
      + enable_dhcp       = true
      + gateway_ip        = "192.168.200.1"
      + id                = (known after apply)
      + ip_version        = 4
      + ipv6_address_mode = (known after apply)
      + ipv6_ra_mode      = (known after apply)
      + name              = "INTERNET"
      + network_id        = (known after apply)
      + no_gateway        = false
      + region            = (known after apply)
      + service_types     = (known after apply)
      + tenant_id         = (known after apply)

      + allocation_pool {
          + end   = "192.168.200.200"
          + start = "192.168.200.100"
        }
    }

Plan: 3 to add, 0 to change, 0 to destroy.

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.
[altlinux@controlvm bin]$
```

- Запускаем развёртывание данных ресурсов:

```
terraform apply
```

  - Подтверждаем развёртывание введя **yes**:

```
Plan: 3 to add, 0 to change, 0 to destroy.

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
```

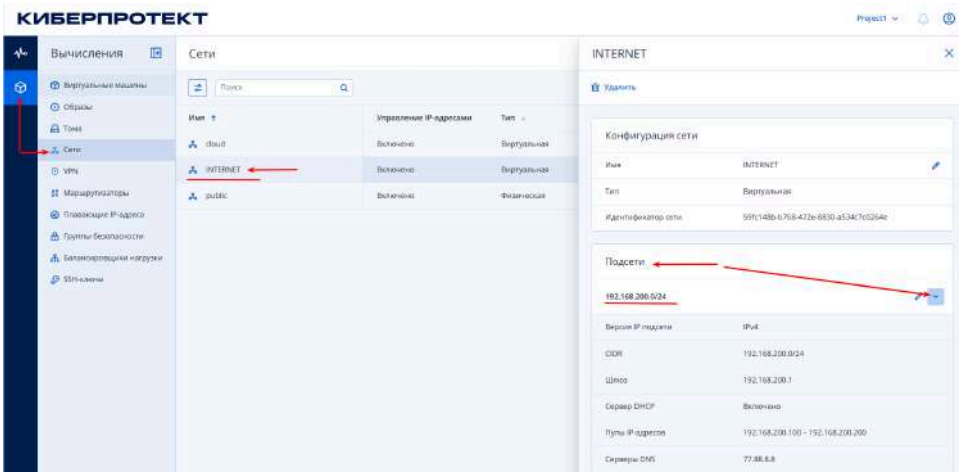
  - Результат:

```
Plan: 3 to add, 0 to change, 0 to destroy.

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

openstack_networking_network_v2.network: Creating...
openstack_networking_network_v2.network: Creation complete after 7s [id=59fc148b-b768-472e-8830-a534c7c0264e]
openstack_networking_subnet_v2.subnet: Creating...
openstack_networking_subnet_v2.subnet: Creation complete after 6s [id=1d7e8d18-b0ec-497d-9a9c-b726064efc50]
openstack_networking_router_interface_v2.router_interface: Creating...
openstack_networking_router_interface_v2.router_interface: Still creating... [10s elapsed]
openstack_networking_router_interface_v2.router_interface: Creation complete after 13s [id=6576962b-6c57-46e1-b843-e40ad371d71a]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
[altlinux@controlvm bin]$
```

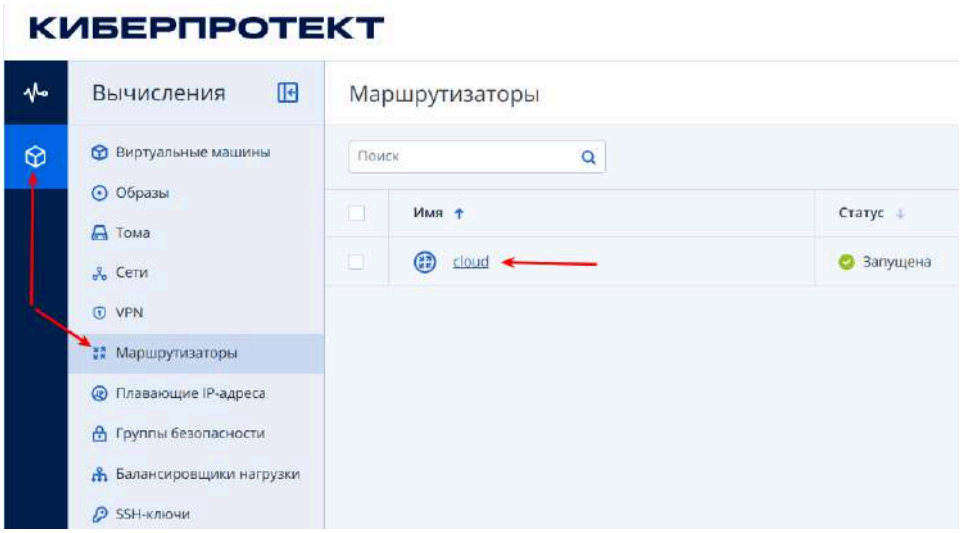
- Проверяем наличие созданных ресурсов средствами **openstack-cli**:
  - Сеть с подсетью:

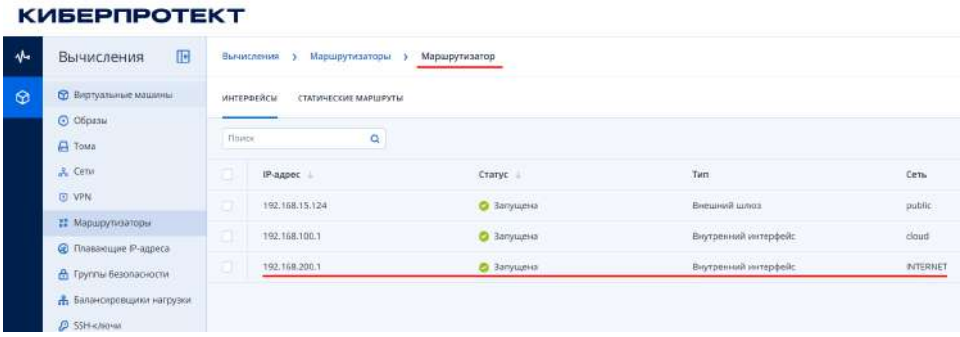
```
[altlinux@controlvm bin]$ openstack --insecure network list
+--------------------------------------+----------+--------------------------------------------------------------------------+
| ID                                   | Name     | Subnets                                                                  |
+--------------------------------------+----------+--------------------------------------------------------------------------+
| 59fc148b-b768-472e-8830-a534c7c0264e | INTERNET | 1d7e8d18-b0ec-497d-9a9c-b726064efc50                                     |
| 88d8e8c2-3b17-425e-b9d8-f7f7b3b360aa | public   | 41a04224-4412-4617-9785-58bb9e3afd9a, ffa6a7f8-3fc5-4d50-afda-0ff17591463d |
| c87b7204-0dd5-4679-a97a-8b96b34bb2d6 | cloud    | 2e112e29-7c2c-413d-b964-145bb0338305                                     |
+--------------------------------------+----------+--------------------------------------------------------------------------+
[altlinux@controlvm bin]$ openstack --insecure subnet list
+--------------------------------------+--------------------+--------------------------------------+-----------------+
| ID                                   | Name               | Network                              | Subnet          |
+--------------------------------------+--------------------+--------------------------------------+-----------------+
| 1d7e8d18-b0ec-497d-9a9c-b726064efc50 | INTERNET           | 59fc148b-b768-472e-8830-a534c7c0264e | 192.168.200.0/24 |
| 2e112e29-7c2c-413d-b964-145bb0338305 |                    | c87b7204-0dd5-4679-a97a-8b96b34bb2d6 | 192.168.100.0/24 |
| 41a04224-4412-4617-9785-58bb9e3afd9a |                    | 88d8e8c2-3b17-425e-b9d8-f7f7b3b360aa | 192.168.15.0/24  |
| ffa6a7f8-3fc5-4d50-afda-0ff17591463d | fip-service-subnet | 88d8e8c2-3b17-425e-b9d8-f7f7b3b360aa | 169.254.32.0/23  |
+--------------------------------------+--------------------+--------------------------------------+-----------------+
[altlinux@controlvm bin]$
```

- Проверяем наличие созданных ресурсов средствами **веб-интерфейса**:
  - Сеть с подсетью:



  - Подсеть добавленная в существующий маршрутизатор (для доступа в сеть Интернет из данной подсети):





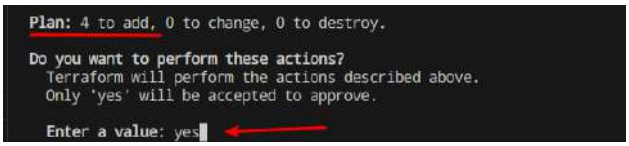
- В файл **network.tf** добавляем следующий код:
  - Создаём порты в сети **INTERNET** в одноимённой подсети для каждого инстанса и балансировщика нагрузки
  - Обращаясь уже к ранее созданным ресурсам, а именно к идентификаторам **id** ресурсов с именемами **network** и **subnet**

```
# Создадим порт для инстанса WebADM
resource "openstack_networking_port_v2" "port_webadm" {
  name       = "webadm"
  network_id = openstack_networking_network_v2.network.id

  fixed_ip {
    subnet_id = openstack_networking_subnet_v2.subnet.id
    ip_address = "192.168.200.20"
  }
}

# Создадим порт для инстанса WEB1
resource "openstack_networking_port_v2" "port_web1" {
  name       = "web1"
  network_id = openstack_networking_network_v2.network.id

  fixed_ip {
    subnet_id = openstack_networking_subnet_v2.subnet.id
    ip_address = "192.168.200.21"
  }
}

# Создадим порт для инстанса WEB2
resource "openstack_networking_port_v2" "port_web2" {
  name       = "web2"
  network_id = openstack_networking_network_v2.network.id

  fixed_ip {
    subnet_id = openstack_networking_subnet_v2.subnet.id
    ip_address = "192.168.200.22"
  }
}

# Создадим порт для балансировщика нагрузки Load Balancer
resource "openstack_networking_port_v2" "port_loadbalancer" {
  name       = "Load Balancer"
  network_id = openstack_networking_network_v2.network.id

  fixed_ip {
    subnet_id = openstack_networking_subnet_v2.subnet.id
    ip_address = "192.168.200.23"
  }
}
```

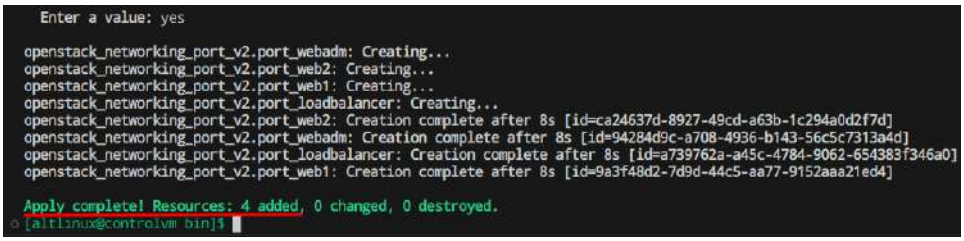
- Проверяем конфигурацию и план **terraform validate** и **terraform plan** и запускаем развёртывание ресурсов:

```
terraform apply
```

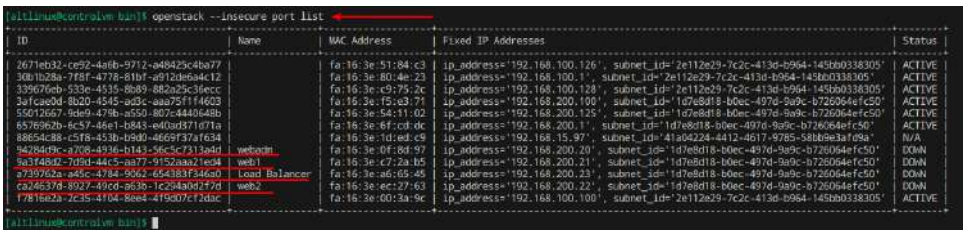
  - Подтверждаем развёртывание введя **yes**:



  - Результат:



- Проверяем наличие созданных ресурсов средствами **openstack-cli**:
  - в веб-интерфейсе нет возможности посмотреть созданные порты;



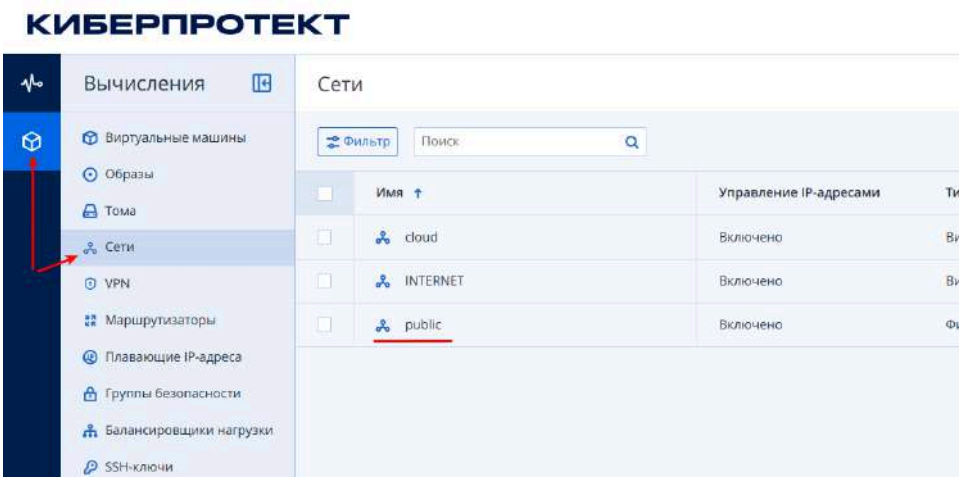
- В файл **network.tf** добавляем следующий код:
  - Создаём плавающие IP-адреса ("публичные") для каждого инстанса и балансировщика нагрузки;
  - Плавающие IP-адреса должны браться из "Публичной" сети указав её имя, в данном случае **public**;

```
# Создадим плавающий IP для инстанса WebADM
resource "openstack_networking_floatingip_v2" "floatingip_webadm" {
  pool = "public"
}

# Создадим плавающий IP для инстанса WEB1
resource "openstack_networking_floatingip_v2" "floatingip_web1" {
  pool = "public"
}

# Создадим плавающий IP для инстанса WEB2
resource "openstack_networking_floatingip_v2" "floatingip_web2" {
  pool = "public"
}

# Создадим плавающий IP для балансировщика нагрузки Load Balancer
resource "openstack_networking_floatingip_v2" "floatingip_loadbalancer" {
  pool = "public"
}
```

- Имя "публичной" сети можно посмотреть через **openstack-cli**:
  - Та сеть, которую ранее не создавали в ручную для работы ControlVM **(cloud)** и средствами Terraform в соответствие с топологией (**INTERNET**);

```
[altlinux@controlvm bin]$ openstack --insecure network list
+--------------------------------------+----------+--------------------------------------+
| ID                                   | Name     | Subnets                              |
+--------------------------------------+----------+--------------------------------------+
| 59fc148b-b768-472e-8830-a534c7c0264e | INTERNET | 1d7e8d18-b0ec-497d-9a9c-b726064efc50 |
| 88d8e8c2-3b17-425e-b9d8-f7f7b3b360aa | public   | 41a04224-4412-4617-9785-58bb9e3afd9a, ffa6a7f8-3fc5-4d50-afda-0ff17591463d |
| c87b7204-0dd5-4679-a97a-8b96b34bb2d6 | cloud    | 2e112e29-7c2c-413d-b964-145bb0338305 |
+--------------------------------------+----------+--------------------------------------+
[altlinux@controlvm bin]$
```

- Имя "публичной" сети можно посмотреть через **веб-интерфейс**:



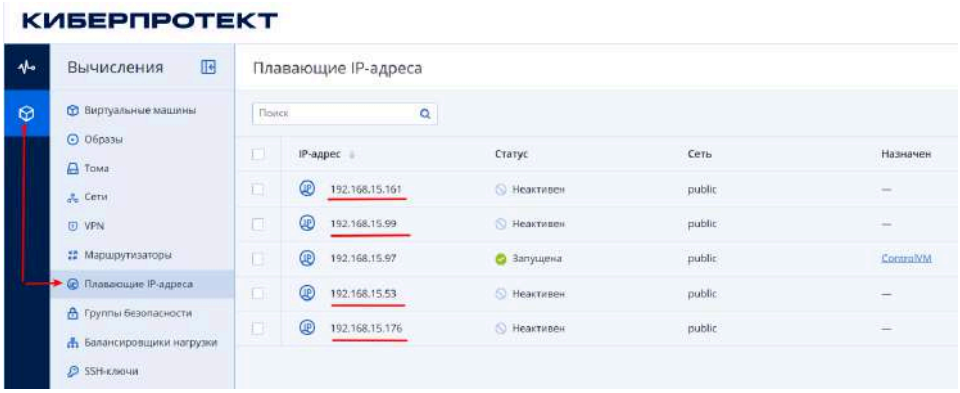
- Проверяем конфигурацию и план **terraform validate** и **terraform plan** и запускаем развёртывание ресурсов **terraform apply**:
  - Результат:

```
  Enter a value: yes

openstack_networking_floatingip_v2.floatingip_webadm: Creating...
openstack_networking_floatingip_v2.floatingip_web2: Creating...
openstack_networking_floatingip_v2.floatingip_web1: Creating...
openstack_networking_floatingip_v2.floatingip_loadbalancer: Creating...
openstack_networking_floatingip_v2.floatingip_webadm: Still creating... [10s elapsed]
openstack_networking_floatingip_v2.floatingip_web2: Still creating... [10s elapsed]
openstack_networking_floatingip_v2.floatingip_web1: Still creating... [10s elapsed]
openstack_networking_floatingip_v2.floatingip_loadbalancer: Still creating... [10s elapsed]
openstack_networking_floatingip_v2.floatingip_web2: Creation complete after 11s [id=13336582-a17a-4de5-8e33-a0151662cb77]
openstack_networking_floatingip_v2.floatingip_web1: Creation complete after 11s [id=dc648177-ac4c-44d7-8391-15130960f2d7]
openstack_networking_floatingip_v2.floatingip_webadm: Creation complete after 11s [id=750a2a96-6d20-4f15-81ac-fcb44db85192]
openstack_networking_floatingip_v2.floatingip_loadbalancer: Creation complete after 11s [id=cc559f43-dd65-4cba-aed6-cdab66a66027]

Apply complete! Resources: 4 added, 0 changed, 0 destroyed.
[altlinux@controlvm bin]$
```

- Проверяем наличие созданных ресурсов средствами **openstack-cli**:
  - Плавающие IP-адреса (4 шт.):

```
[altlinux@controlvm bin]$ openstack --insecure floating ip list
+--------------------------------------+---------------------+------------------+------+--------------------------------------+----------------------------------+
| ID                                   | Floating IP Address | Fixed IP Address | Port | Floating Network                     | Project                          |
+--------------------------------------+---------------------+------------------+------+--------------------------------------+----------------------------------+
| 13336582-a17a-4de5-8e33-a0151662cb77 | 192.168.15.161      | None             | None | 88d8e8c2-3b17-425e-b9d8-f7f7b3b360aa | 4deb49c7b35546948e7e1a1a39941814 |
| 750a2a96-6d20-4f15-81ac-fcb44db85192 | 192.168.15.99       | None             | None | 88d8e8c2-3b17-425e-b9d8-f7f7b3b360aa | 4deb49c7b35546948e7e1a1a39941814 |
| 92c0452c-ad30-47e3-ad1a-c00fd29a739e | 192.168.15.97       | 192.168.100.128  | 339676eb-533e-4535-8b89-882a25c36ecc | 88d8e8c2-3b17-425e-b9d8-f7f7b3b360aa | 4deb49c7b35546948e7e1a1a39941814 |
| cc559f43-dd65-4cba-aed6-cdab66a66027 | 192.168.15.53       | None             | None | 88d8e8c2-3b17-425e-b9d8-f7f7b3b360aa | 4deb49c7b35546948e7e1a1a39941814 |
| dc648177-ac4c-44d7-8391-15130960f2d7 | 192.168.15.176      | None             | None | 88d8e8c2-3b17-425e-b9d8-f7f7b3b360aa | 4deb49c7b35546948e7e1a1a39941814 |
+--------------------------------------+---------------------+------------------+------+--------------------------------------+----------------------------------+
[altlinux@controlvm bin]$
```

- Проверяем наличие созданных ресурсов средствами **веб-интерфейса**:
  - Плавающие IP-адреса (4 шт.):

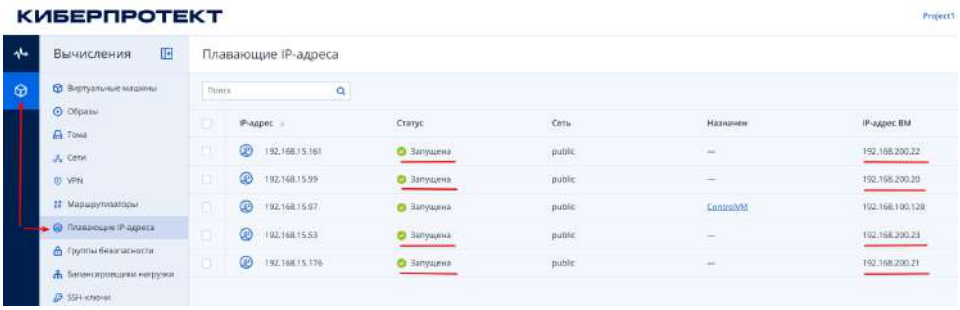


- В файл **network.tf** добавляем следующий код:
  - Создавая ассоциацию ранее созданного плавающего IP-адреса с ещё ранее созданным портов;
  - Для каждого инстанса и балансировщика нагрузки;

```
# Создадим для WebADM ассоциацию плавающего IP и порт (публичного и приватного IP адресов)
resource "openstack_networking_floatingip_associate_v2" "association_webadm" {
  port_id     = openstack_networking_port_v2.port_webadm.id
  floating_ip = openstack_networking_floatingip_v2.floatingip_webadm.address
}

# Создадим для WEB1 ассоциацию плавающего IP и порт (публичного и приватного IP адресов)
resource "openstack_networking_floatingip_associate_v2" "association_web1" {
  port_id     = openstack_networking_port_v2.port_web1.id
  floating_ip = openstack_networking_floatingip_v2.floatingip_web1.address
}

# Создадим для WEB2 ассоциацию плавающего IP и порт (публичного и приватного IP адресов)
resource "openstack_networking_floatingip_associate_v2" "association_web2" {
  port_id     = openstack_networking_port_v2.port_web2.id
  floating_ip = openstack_networking_floatingip_v2.floatingip_web2.address
}

# Создадим для Load Balancer ассоциацию плавающего IP и порт (публичного и приватного IP адресов)
resource "openstack_networking_floatingip_associate_v2" "association_loadbalancer" {
  port_id     = openstack_networking_port_v2.port_loadbalancer.id
  floating_ip = openstack_networking_floatingip_v2.floatingip_loadbalancer.address
}
```

- Проверяем конфигурацию и план **terraform validate** и **terraform plan** и запускаем развёртывание ресурсов **terraform apply**:
  - Результат:

- Проверяем в **веб-интерфейсе** статус плавающих IP-адресов сменился с **Неактивен** на **Запущена**:
  - Также появились **IP-адреса ВМ** (но сами ВМ (инстансы) ещё не созданы) из подсети **INTERNET** (192.168.200.0/24 созданной ранее);



- Создаём файл **security.tf** и описываем **последовательно** часть касающуюся сетевой безопасности (с точки зрения требования задания) для развёртывания данной инфраструктуры с **поэтапным запуском** и **наблюдением** созданных ресурсов:

```
vim network.tf
```

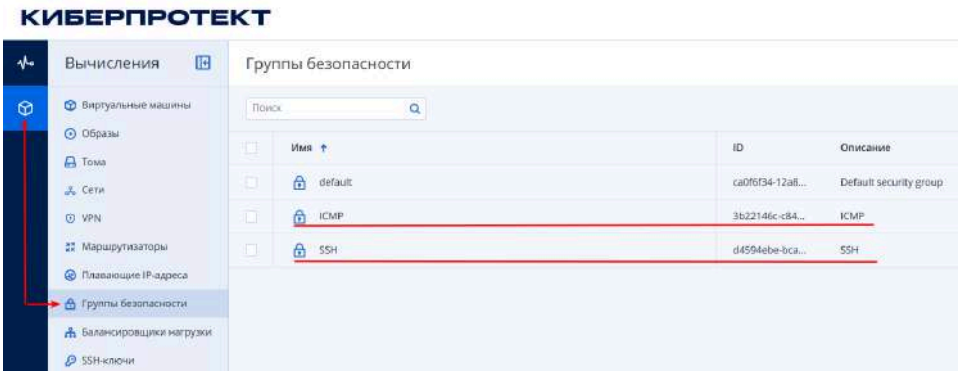
  - Помещаем следующее содержимое:
    - Создаём **две** группы безопасности для будущего контроля **ICMP** и **SSH**

```
# Создаём группу безопасности для ICMP
resource "openstack_networking_secgroup_v2" "secgroup_1" {
  name = "ICMP"
  description = "ICMP"
}

# Создаём группу безопасности для SSH
resource "openstack_networking_secgroup_v2" "secgroup_2" {
  name = "SSH"
  description = "SSH"
}
```

- Проверяем конфигурацию и план **terraform validate** и **terraform plan** и запускаем развёртывание ресурсов **terraform apply**:
  - Результат:

- Проверяем наличие созданных ресурсов средствами **openstack-cli**:
  - Группы безапасности:

- Проверяем наличие созданных ресурсов средствами **веб-интерфейса**:
  - Группы безапасности:



- В файл **security.tf** добавляем следующий код:
  - Создавая правило, в ранее созданную группу безопасности обращаясь по идентификатору **id** созданного ресурса **secgroup_1** для фильтрации по протоколу **icmp** с любых IP-адресов для входящего трафика;
  - Создавая правило, в ранее созданную группу безопасности обращаясь по идентификатору **id** созданного ресурса **secgroup_2** для фильтрации по протоколу **tcp** на порт **22** (SSH) с любых IP-адресов для входящего трафика;
  - Ограничивая тем самом доступ к будущим инстансам по ICMP и SSH из внешних сетей;

```
# Добавляем правило в группу безопасности для ICMP
resource "openstack_networking_secgroup_rule_v2" "secgroup_rule_icmp" {
  direction         = "ingress"
  ethertype         = "IPv4"
  protocol          = "icmp"
  remote_ip_prefix  = "0.0.0.0/0"
  security_group_id = openstack_networking_secgroup_v2.secgroup_1.id
}

# Добавляем правило в группу безопасности для SSH
resource "openstack_networking_secgroup_rule_v2" "secgroup_rule_ssh" {
  direction         = "ingress"
  ethertype         = "IPv4"
  protocol          = "tcp"
  port_range_min    = 22
  port_range_max    = 22
  remote_ip_prefix  = "0.0.0.0/0"
  security_group_id = openstack_networking_secgroup_v2.secgroup_2.id
}
```

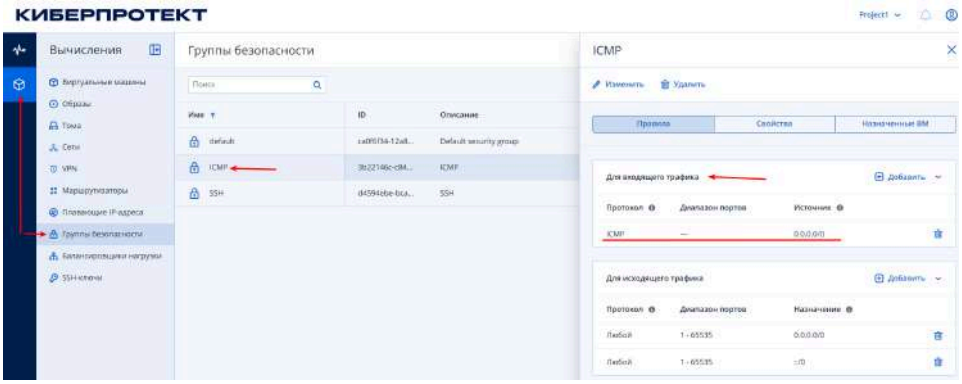
- Проверяем конфигурацию и план **terraform validate** и **terraform plan** и запускаем развёртывание ресурсов **terraform apply**:
  - Результат:

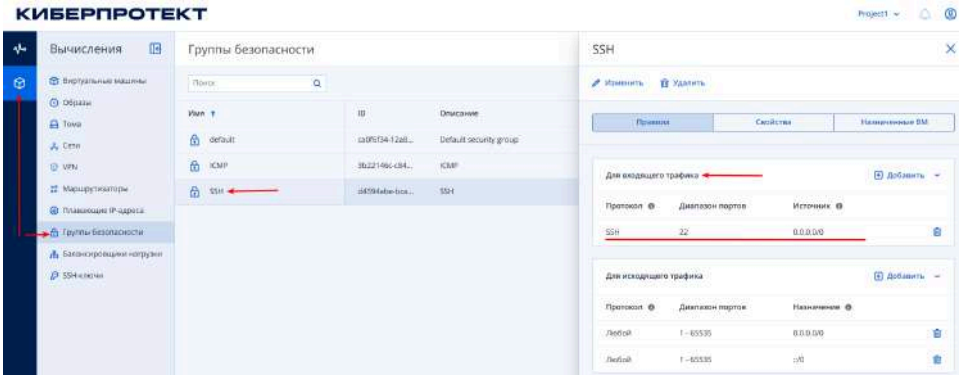
```
  Enter a value: yes

openstack_networking_secgroup_rule_v2.secgroup_rule_ssh: Creating...
openstack_networking_secgroup_rule_v2.secgroup_rule_icmp: Creating...
openstack_networking_secgroup_rule_v2.secgroup_rule_icmp: Creation complete after 1s [id=ad88027a-77d7-4a24-90dc-03b86b081360]
openstack_networking_secgroup_rule_v2.secgroup_rule_ssh: Creation complete after 1s [id=4ab8a53a-63fe-41d2-961e-3406e0a4aa21]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
[altlinux@controlvm bin]$
```

- Проверяем наличие созданных ресурсов средствами **openstack-cli**:
  - Правила в группах безопасности:

```
[altlinux@controlvm bin]$ openstack --insecure security group rule list
+--------------------------------------+-------------+-----------+-----------+-------------+--------------------------------------+----------------------+--------------------------------------+
| ID                                   | IP Protocol | Ethertype | IP Range  | Port Range  | Direction | Remote Security Group    | Remote Address Group | Security Group                       |
+--------------------------------------+-------------+-----------+-----------+-------------+-----------+--------------------------+----------------------+--------------------------------------+
| 0986faa6-3719-497a-aa05-68e422ae9b64 | None        | IPv6      | ::/0      |             | egress    | None                     | None                 | ca0f6f34-12a8-4c36-a466-4b30a6ea80   |
| 2173a6dc-c1ad-44c9-834a-349f67c640bc | None        | IPv4      | 0.0.0.0/0 |             | ingress   | ca0f6f34-12a8-4c36-a466-4b30a6ea807d | None                 | ca0f6f34-12a8-4c36-a466-4b30a6ea80   |
| 34da14db-167f-4fb6-8630-41553e0a35f0 | None        | IPv4      | 0.0.0.0/0 |             | egress    | None                     | None                 | d4594ebe-bcae-4396-a991-c9cd6c2b16   |
| 3f8f3dde-fe56-4f6c-babf-cedcba2b595c | None        | IPv6      | ::/0      |             | ingress   | ca0f6f34-12a8-4c36-a466-4b30a6ea807d | None                 | ca0f6f34-12a8-4c36-a466-4b30a6ea80   |
| 4ab8a53a-63fe-41d2-961e-3406e0a4aa21 | tcp         | IPv4      | 0.0.0.0/0 | 22:22       | ingress   | None                     | None                 | d4594ebe-bcae-4396-a991-c9cd6c2b16   |
| 4b32153c-9b33-455e-8741-136466d71434 | None        | IPv4      | 0.0.0.0/0 |             | egress    | None                     | None                 | 3b221a6c-c84e-4b50-9b2b-e232a770a6   |
| 8448a4e4-467b-473d-ae7f-934cb0c88593 | None        | IPv6      | ::/0      |             | ingress   | None                     | None                 | ca0f6f34-12a8-4c36-a466-4b30a6ea80   |
| 9421176f-3853-49dc-8e54-951194e21a6f | None        | IPv4      | 0.0.0.0/0 |             | ingress   | None                     | None                 | ca0f6f34-12a8-4c36-a466-4b30a6ea80   |
| a21b9d83-88d5-4cbe-ae73-ced406e28d06 | None        | IPv6      | ::/0      |             | egress    | None                     | None                 | d4594ebe-bcae-4396-a991-c9cd6c2b16   |
| a7217f79-4a8c-4069-bac0-bb7306e41c9f | None        | IPv6      | ::/0      |             | egress    | None                     | None                 | 3b221a6c-c84e-4b50-9b2b-e232a770a6   |
| ad88027a-77d7-4a24-90dc-03b86b081360 | icmp        | IPv4      | 0.0.0.0/0 |             | ingress   | None                     | None                 | 3b221a6c-c84e-4b50-9b2b-e232a770a6   |
| ff9ad581-757f-4a3a-abff-8962b1b9cbd6 | None        | IPv4      | 0.0.0.0/0 |             | egress    | None                     | None                 | ca0f6f34-12a8-4c36-a466-4b30a6ea80   |
+--------------------------------------+-------------+-----------+-----------+-------------+-----------+--------------------------+----------------------+--------------------------------------+
[altlinux@controlvm bin]$
```

- Проверяем наличие созданных ресурсов средствами **веб-интерфейса**:
  - Правила в группах безопасности:

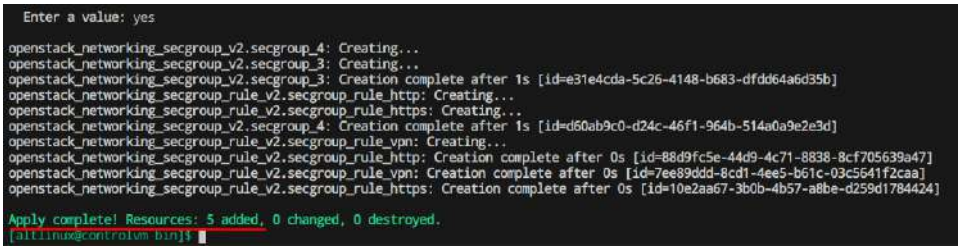




- В файл **security.tf** добавляем следующий код:
  - Поскольку в дальнейшем по заданию на инстансах **WEB1** и **WEB2** должно быть развёрнуто веб-приложение, а значит они должны принимать входящий трафик на порты **80/tcp** и **443/tcp** (HTTP и HTTPS)
  - Поскольку в дальнейшем по заданию необходимо реализовать туннельное соединение между инстансами **WebADM** и **WEB1**, **WEB2**, а значит WebADM должен принимать входящий трафик на порт реализуемого VPN-соединения, например WireGuard (**51820/udp**);
  - Таким образом необходимо создать ещё 1 или 2 группы безопасности с соответствующими правилами;
  - В данном примере создаются **две** группы безопасности с именами **WEB** и **VPN** и **три** правила: **два** для группы **WEB** и **одно** для группы **VPN**;

```
# Создаём группу безопасность для WEB-трафика (HTTP/HTTPS)
resource "openstack_networking_secgroup_v2" "secgroup_3" {
  name = "WEB"
  description = "WEB for HTTP/HTTPS"
}

# Добавляем первое правило в группу безопасности для WEB (HTTP)
resource "openstack_networking_secgroup_rule_v2" "secgroup_rule_http" {
  direction         = "ingress"
  ethertype         = "IPv4"
  protocol          = "tcp"
  port_range_min    = 80
  port_range_max    = 80
  remote_ip_prefix  = "0.0.0.0/0"
  security_group_id = openstack_networking_secgroup_v2.secgroup_3.id
}

# Добавляем второе правило в группу безопасности для WEB (HTTPS)
resource "openstack_networking_secgroup_rule_v2" "secgroup_rule_https" {
  direction         = "ingress"
  ethertype         = "IPv4"
  protocol          = "tcp"
  port_range_min    = 443
  port_range_max    = 443
  remote_ip_prefix  = "0.0.0.0/0"
  security_group_id = openstack_networking_secgroup_v2.secgroup_3.id
}

# Создаём группу безопасность для VPN (WireGuard)
resource "openstack_networking_secgroup_v2" "secgroup_4" {
  name = "VPN"
  description = "VPN (Wireguard)"
}

# Добавляем правило в группу безопасности для VPN (WireGuard)
resource "openstack_networking_secgroup_rule_v2" "secgroup_rule_vpn" {
  direction         = "ingress"
  ethertype         = "IPv4"
  protocol          = "udp"
  port_range_min    = 51820
  port_range_max    = 51820
  remote_ip_prefix  = "0.0.0.0/0"
  security_group_id = openstack_networking_secgroup_v2.secgroup_4.id
}
```

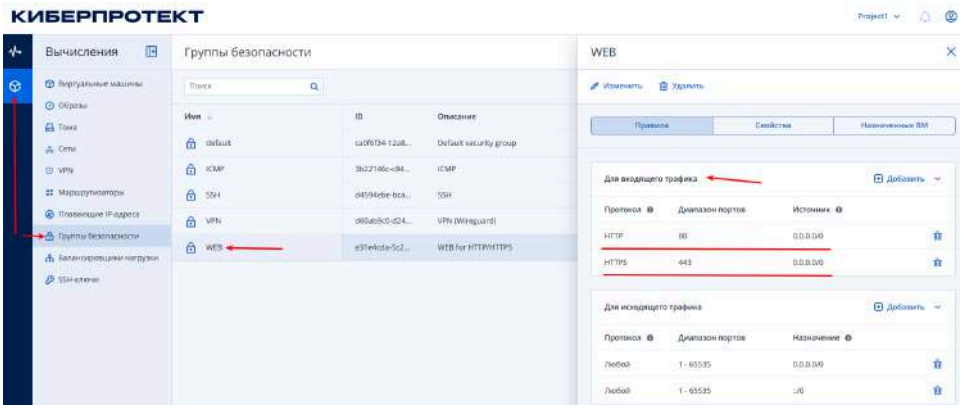
- Проверяем конфигурацию и план **terraform validate** и **terraform plan** и запускаем развёртывание ресурсов **terraform apply**:
  - Результат:

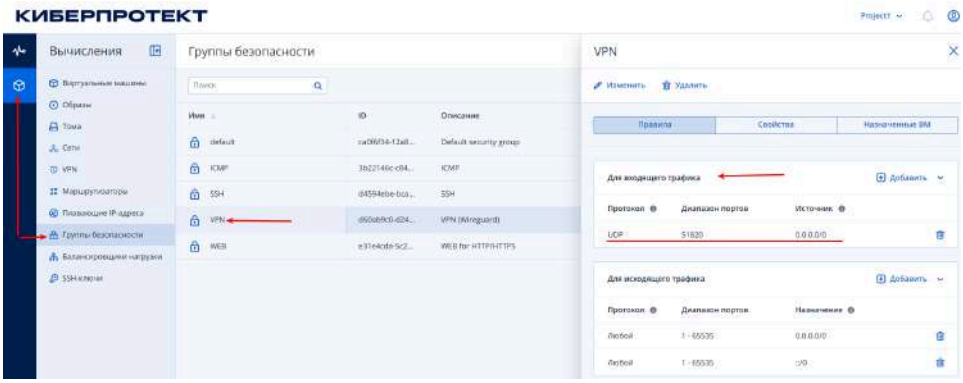


- Проверяем наличие созданных ресурсов средствами **openstack-cli**:
  - Правила в группах безопасности:

```
[altlinux@controlvm bin]$ openstack --insecure security group rule list
+--------------------------------------+-------------+-----------+-----------+-------------+-----------+--------------------------------------+
| ID                                   | IP Protocol | Ethertype | IP Range  | Port Range  | Direction | Remote Security Group                |
+--------------------------------------+-------------+-----------+-----------+-------------+-----------+--------------------------------------+
| 0986faa0-3719-497a-aa05-68e422ae9b64 | None        | IPv6      | ::/0      |             | egress    | None                                 |
| 10e2aa67-3b0b-4b57-a8be-d259d1784424 | tcp         | IPv4      | 0.0.0.0/0 | 443:443     | ingress   | None                                 |
| 2173a6dc-c1ad-44c9-834a-349f67c844bc | None        | IPv4      | 0.0.0.0/0 |             | ingress   | ca0f6f34-12a8-4c36-a466-4b30a6ea807d |
| 2bfa8a46-f6c3-4a5d-a39c-662d73c3f896 | None        | IPv6      | ::/0      |             | egress    | None                                 |
| 34da24db-167f-4fb6-8630-41553e0e35f0 | None        | IPv4      | 0.0.0.0/0 |             | egress    | None                                 |
| 3f9f3dde-fe56-4f6c-babf-cedcba2b595c | None        | IPv6      | ::/0      |             | ingress   | ca0f6f34-12a8-4c36-a466-4b30a6ea807d |
| 4ab8a53a-63fe-41d2-961e-3406e0a4aa21 | tcp         | IPv4      | 0.0.0.0/0 | 22:22       | ingress   | None                                 |
| 4b32153c-9b33-455e-8741-136466d71434 | None        | IPv4      | 0.0.0.0/0 |             | egress    | None                                 |
| 51cff047-cef4-446e-8690-1a301121e94e | None        | IPv4      | 0.0.0.0/0 |             | egress    | None                                 |
| 66796b6f-4828-4cf6-8ad9-15b4c055cad6 | None        | IPv4      | 0.0.0.0/0 |             | egress    | None                                 |
| 7ee89ddd-8cd1-4ee5-b61c-03c5641f2caa | udp         | IPv4      | 0.0.0.0/0 | 51820:51820 | ingress   | None                                 |
| 8448a4e4-467b-473d-ae7f-934cb0cb8593 | None        | IPv6      | ::/0      |             | ingress   | None                                 |
| 88d9fc5e-44d9-4c71-8838-8cf705639a47 | tcp         | IPv4      | 0.0.0.0/0 | 80:80       | ingress   | None                                 |
| 9421176f-3853-49dd-8e54-951194e21a6f | None        | IPv4      | 0.0.0.0/0 |             | ingress   | None                                 |
| a21b9d83-88d5-4cba-ae73-cad40be28d66 | None        | IPv6      | ::/0      |             | egress    | None                                 |
| a7217f79-4a8c-4069-bac0-bb7306ce1c9f | None        | IPv6      | ::/0      |             | egress    | None                                 |
| ad88027a-77d7-4a24-90dc-03b86b081360 | icmp        | IPv4      | 0.0.0.0/0 |             | ingress   | None                                 |
| dd94a646-810c-457e-89d1-64a72cd5a35a | None        | IPv6      | ::/0      |             | egress    | None                                 |
| f19ad581-757f-4a3a-abff-8962b1b9cbd5 | None        | IPv4      | 0.0.0.0/0 |             | egress    | None                                 |
+--------------------------------------+-------------+-----------+-----------+-------------+-----------+--------------------------------------+
[altlinux@controlvm bin]$
```

- Проверяем наличие созданных ресурсов средствами **веб-интерфейса**:
  - Правила в группах безопасности:





- В файл **network.tf** добавляем следующий код:
  - Назначая ранее созданные группы безопасности на ранее созданные порты для каждого инстанса, обращаясь по идентификаторам **id** созданных ранее ресурсов;
  - На порт для инстанса **WebADM** назначаем группы безопасности **ICMP, SSH** (явное требование задания) и **VPN**, т.к. в дальнейшем данный инстанс будет выступать в роли VPN-сервера и должен принимать соединения на порт указанный в правиле для данной группы безопасности;
  - На порт для инстансов **WEB1** и **WEB2** назначаем группы безопасности **ICMP, SSH** (явное требование задания) и **WEB**, т.к. в дальнейшем на данных инстансах будет развёрнуто веб-приложение и должны приниматься соединения на порты указанные в правилах для данной группы безопасности;

```
# Назначим группы (ICMP, SSH, VPN) безопасности на порт для инстанса WebADM
resource "openstack_networking_port_secgroup_associate_v2" "security_group_associate_webadm" {
    port_id            = openstack_networking_port_v2.port_webadm.id
    enforce            = true
    security_group_ids = [
      openstack_networking_secgroup_v2.secgroup_1.id,
      openstack_networking_secgroup_v2.secgroup_2.id,
      openstack_networking_secgroup_v2.secgroup_4.id
    ]
}

# Назначим группы (ICMP, SSH, WEB) безопасности на порт для инстанса WEB1
resource "openstack_networking_port_secgroup_associate_v2" "security_group_associate_web1" {
    port_id            = openstack_networking_port_v2.port_web1.id
    enforce            = true
    security_group_ids = [
      openstack_networking_secgroup_v2.secgroup_1.id,
      openstack_networking_secgroup_v2.secgroup_2.id,
      openstack_networking_secgroup_v2.secgroup_3.id
    ]
}

# Назначим группы (ICMP, SSH, WEB) безопасности на порт для инстанса WEB2
resource "openstack_networking_port_secgroup_associate_v2" "security_group_associate_web2" {
    port_id            = openstack_networking_port_v2.port_web2.id
    enforce            = true
    security_group_ids = [
      openstack_networking_secgroup_v2.secgroup_1.id,
      openstack_networking_secgroup_v2.secgroup_2.id,
      openstack_networking_secgroup_v2.secgroup_3.id
    ]
}
```

- Проверяем конфигурацию и план **terraform validate** и **terraform plan** и запускаем развёртывание ресурсов **terraform apply**:
  - Результат:

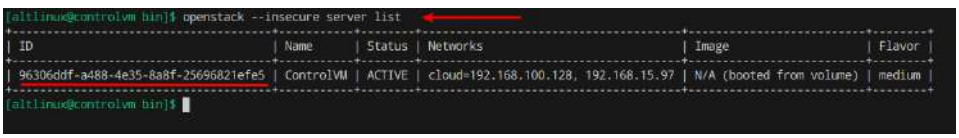
- Создаём файл **instance.tf** и описываем **последовательно** часть для развёртывания необходимых инстансов (виртуальных машин) с **поэтапным запуском** и **наблюдением** созданных ресурсов:

```
vim instance.tf
```

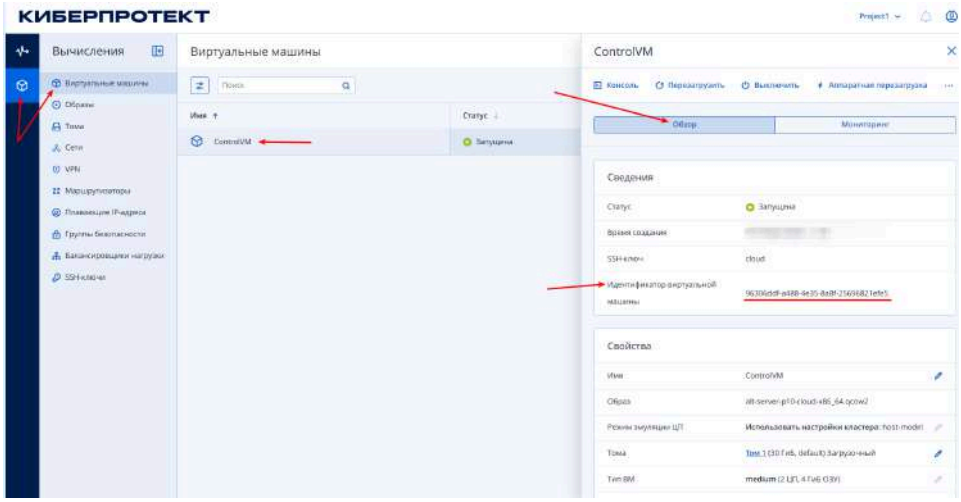
  - Помещаем следующее содержимое:
    - Т.к. все ресурсы и сети должны быть созданы в соответствие с топологией по условиям задания, а на топологии **ControlVM** также должна быть подключена к сети **INTERNET**;
    - То добавляем к инстансу **ControlVM** ещё один сетевой интерфейс из подсети **INTERNET** и назначаем фиксированный IP-адрес;
    - Необходимо указать идентификатор **id** виртуальной машины **ControlVM**

```
# Подключаем в ControlVM интерфейс из подсети "INTERNET" в соответствие с топологией
resource "openstack_compute_interface_attach_v2" "controlvm" {
  instance_id = "96306ddf-a488-4e35-8a8f-25696821efe5"
  network_id  = openstack_networking_network_v2.network.id
  fixed_ip = "192.168.200.10"
  depends_on = [ openstack_networking_subnet_v2.subnet ]
}
```

  - **instance_id** (идентификатор) можно получить средствами **openstack-cli**:



  - **instance_id** (идентификатор) можно получить средствами **веб-интерфейса**:



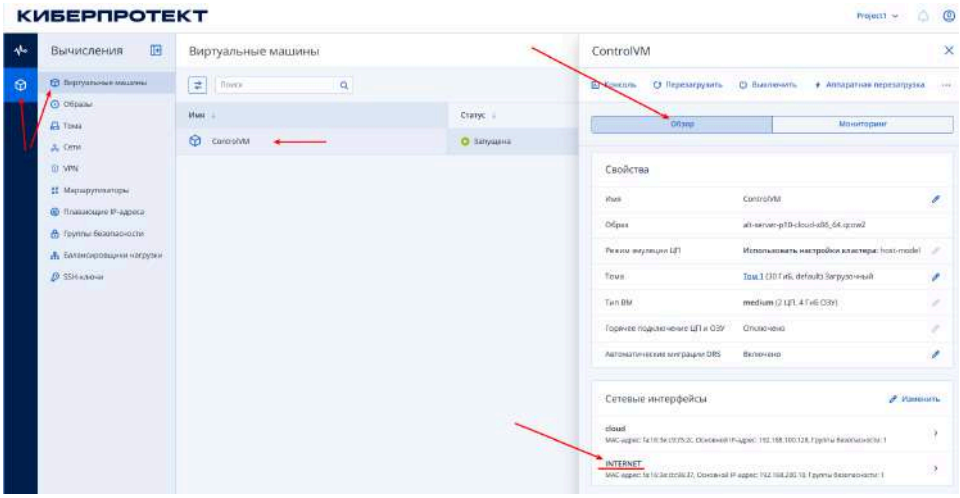
- Проверяем конфигурацию и план **terraform validate** и **terraform plan** и запускаем развёртывание ресурсов **terraform apply**:
  - Результат:

```
  Enter a value: yes

openstack_compute_interface_attach_v2.controlvm: Creating...
openstack_compute_interface_attach_v2.controlvm: Still creating... [10s elapsed]
openstack_compute_interface_attach_v2.controlvm: Creation complete after 17s [id=96306ddf-a488-4e35-8a8f-25696821efe5/bea9153c-e7cd-41f7-8592-35047e452ed3]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[altlinux@controlvm bin]$
```

  - Проверяем наличие второго сетевого интерфейса из подсети **INTERNET** средствами **openstack-cli**:

```
[altlinux@controlvm bin]$ openstack --insecure server show ControlVM | grep "address"
| addresses                   | INTERNET=192.168.200.10; cloud=192.168.100.128, 192.168.15.97

[altlinux@controlvm bin]$
```

  - Проверяем наличие второго сетевого интерфейса из подсети **INTERNET** средствами **веб-интерфейса**:



- В файл **instance.tf** добавляем следующий код:
  - Описывая конфигурацию инстансов **WebADM, WEB1** и **WEB2**;
  - См. ниже информацию о переменных **var.*** и файле **cloud-init.yml**;

```
# Создаём инстанс с именем "WebADM"
resource "openstack_compute_instance_v2" "webadm" {
  name            = "WebADM"
  flavor_id       = var.flavor_id
  key_pair        = var.key_pair
  user_data       = file("cloud-init.yml")

  block_device {
    uuid                  = var.image_id
    source_type           = "image"
    volume_size           = 10
    boot_index            = 0
    destination_type      = "volume"
    delete_on_termination = true
  }

  network {
    port = openstack_networking_port_v2.port_webadm.id
  }
}

# Создаём инстанс с именами "WEB1"
resource "openstack_compute_instance_v2" "web1" {
  name            = "WEB1"
  flavor_id       = var.flavor_id
  key_pair        = var.key_pair
  user_data       = file("cloud-init.yml")

  block_device {
    uuid                  = var.image_id
    source_type           = "image"
    volume_size           = 10
    boot_index            = 0
    destination_type      = "volume"
    delete_on_termination = true
  }

  network {
    port = openstack_networking_port_v2.port_web1.id
  }
}

# Создаём инстанс с именами "WEB2"
resource "openstack_compute_instance_v2" "web2" {
  name            = "WEB2"
  flavor_id       = var.flavor_id
  key_pair        = var.key_pair
  user_data       = file("cloud-init.yml")

  block_device {
    uuid                  = var.image_id
    source_type           = "image"
    volume_size           = 10
    boot_index            = 0
    destination_type      = "volume"
    delete_on_termination = true
  }

  network {
    port = openstack_networking_port_v2.port_web2.id
  }
}
```

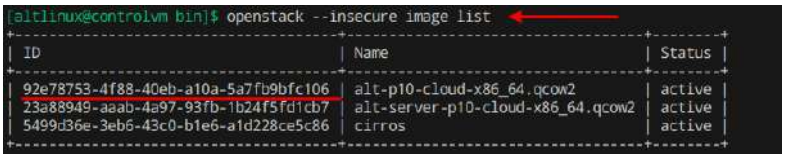
- В файл **variables.tf** добавляем следующий код:
  - Определяя значение переменных указывая идентификаторы **id** для соответствующих (существующих) ресурсов;
  - См. ниже где брать идентификаторы;
  - Учитывая основные требования задания для создаваемых инстансов:

. Основные характеристики:

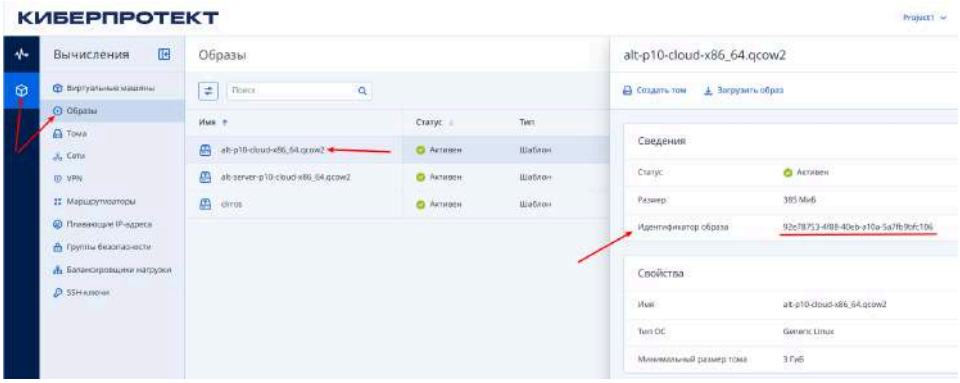
- i. Операционная система: Альт p10 StarterKit/~~Альт Сервер p10-cloud~~
- ii. Количество vCPU: 1.
- iii. Объём оперативной памяти: 1024 МБ.
- iv. Объём диска: 10 ГБ/~~30 ГБ~~

```
# ID для образа "alt-p10-cloud-x86_64.qcow2" (Starterkit)
variable "image_id" {
    type = string
    default = "92e78753-4f88-40eb-a10a-5a7fb9bfc106"
}

# ID для шаблона 1 vCPU 1 RAM (openstack --insecure flavor list)
variable "flavor_id" {
    type = string
    default = "1f64883c-1cdd-45e4-ac8a-82ab57a12fdf"
}

# Имя ssh-ключа
variable "key_pair" {
    type = string
    default = "cloud"
}
```

- Для переменной **image_id** значение можно получить:
  - Средствами **openstack-cli**:



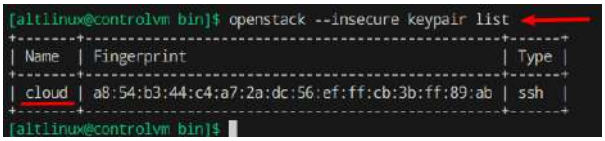
  - Средствами **веб-интерфейса**:



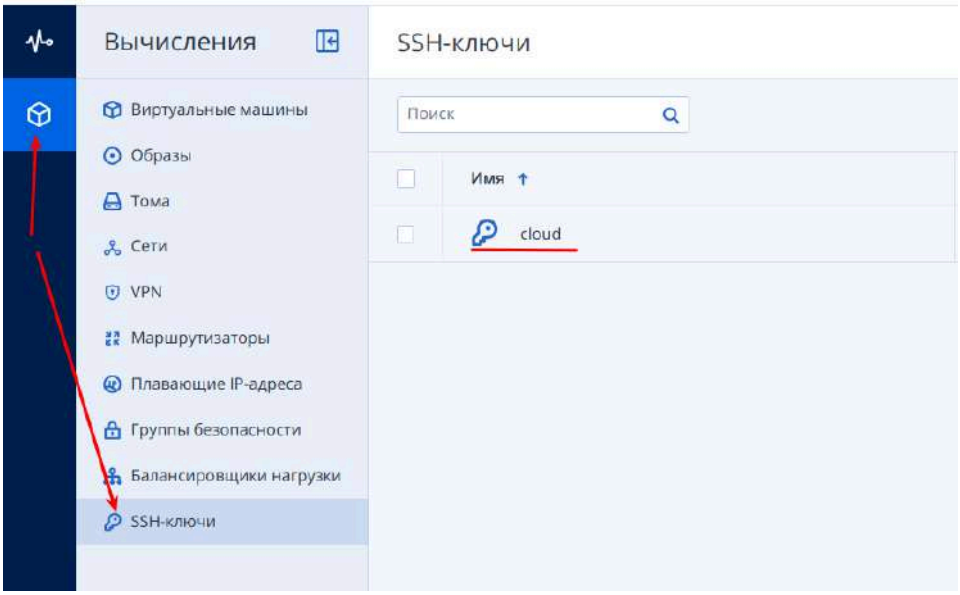
- Для переменной **flavor_id** значение можно получить:
  - Средствами **openstack-cli**:

  - Средствами **веб-интерфейса** нет возможности;

- Для переменной **key_pair** значение можно получить:
  - Средствами **openstack-cli**:



  - Средствами **веб-интерфейса**:



- Средствами **Terraform** передаём именно тот ключ, на основе которого осуществляется доступ с рабочего места до **ControlVM**
  - Чтобы при необходимости можно было подключиться по SSH к каждому инстансу по "публичному" (Плавающему) IP-адресу, что также является требованиями задания;

- Создаём файл **cloud-init.yml**:

```
vim cloud-init.yml
```

  - Помещаем следующее содержимое:
    - Задавая пароль по требованиям задания **P@ssw0rd** для каждого создаваемого инстанса;
    - Также передаём значение публичной части SSH-ключа (генерация ssh-ключей см. ниже), чтобы в дальнейшем был доступ с **ControlVM** до каждого инстанса по SSH;
    - И именно по ключам, т.к. по условиям задания подключение по SSH на основе открытых ключей;
    - Доступ по SSH с **ControlVM** до каждого инстанса необходим для дальнейшей работы **Ansible**;

```
#cloud-config
chpasswd:
  expire: false
  users:
  - {name: altlinux, password: P@ssw0rd, type: text}
  ssh_pwauth: false

users:
  - name: altlinux
    sudo: ALL=(ALL) NOPASSWD:ALL
    groups: wheel
    shell: /bin/bash
    ssh_authorized_keys:
      - ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDFzYMn859ToNndaRRTlXoDnBDgME7rTPNvPlh9hmo4QsvmMdF3lKgQTR6p53cP1zUgxfstkKCfvMlAf
NCE3iJWGxaY8B8tM4fO6jC8eMKe93cIK74lFk1cZnrHHUJSvM4dsD+hsoZ06VPOWSoUtipi9B7AgOF0EsL63GTx0EeajJCOerQUlA6qJM785N
fcRNS23vSHhlRyV84xrfBhP/0w2jiHWy+jNCVtpO/M7lAQ5EtcVegSZupzDoJ6K/4GpgjLoP4eUfr8EVNMLGZ5AJlJmlFL5hTsOwa/42CN+Tw
FSC4G6PFXtWLdZJebPJ2nlcKpydmwSDEeXTy4vPAZWXBD altlinux@controlvm
```

- Генерируем на **ControlVM** ключевую пару и смотрим содержимое публичного (открытого) ключа, которое необходимо указать в файле **cloud-init.yml**:

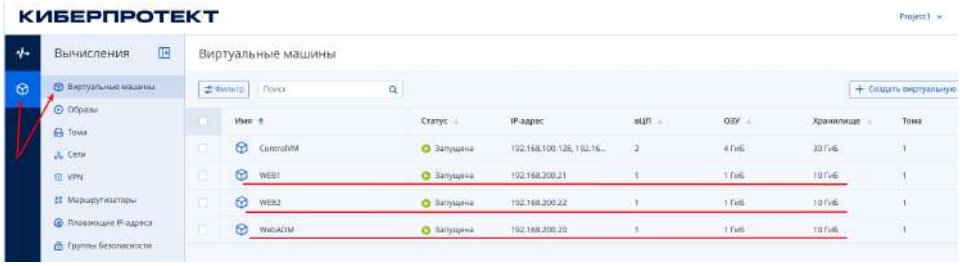
- Проверяем конфигурацию и план **terraform validate** и **terraform plan** и запускаем развёртывание ресурсов **terraform apply**:
  - Результат:

```
Plan: 3 to add, 0 to change, 0 to destroy.

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

openstack_compute_instance_v2.web1: Creating...
openstack_compute_instance_v2.webadm: Creating...
openstack_compute_instance_v2.web2: Creating...
openstack_compute_instance_v2.web1: Still creating... [10s elapsed]
openstack_compute_instance_v2.webadm: Still creating... [10s elapsed]
openstack_compute_instance_v2.web2: Still creating... [10s elapsed]
openstack_compute_instance_v2.web1: Still creating... [20s elapsed]
openstack_compute_instance_v2.webadm: Still creating... [20s elapsed]
openstack_compute_instance_v2.web2: Still creating... [20s elapsed]
openstack_compute_instance_v2.web2: Creation complete after 25s [id=88398bf3-2c63-48c9-817b-5ded169725c7]
openstack_compute_instance_v2.web1: Creation complete after 25s [id=74e2bb78-e034-463c-a2ad-ca0d843d47d3]
openstack_compute_instance_v2.webadm: Creation complete after 25s [id=c4ce7221-58a9-4804-a190-89f438526921]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
[altlinux@controlvm bin]$
```

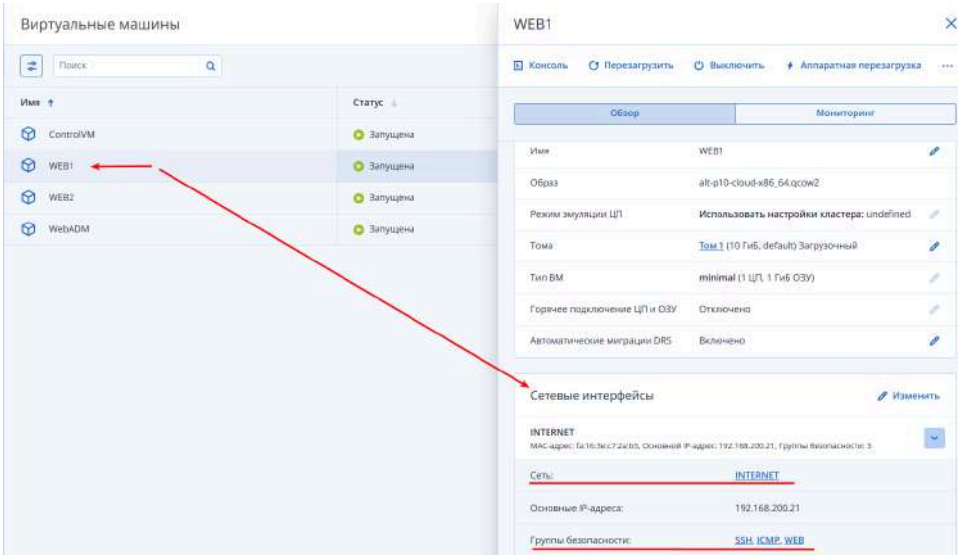
- Проверяем наличие созданных инстансов средствами **openstack-cli**:

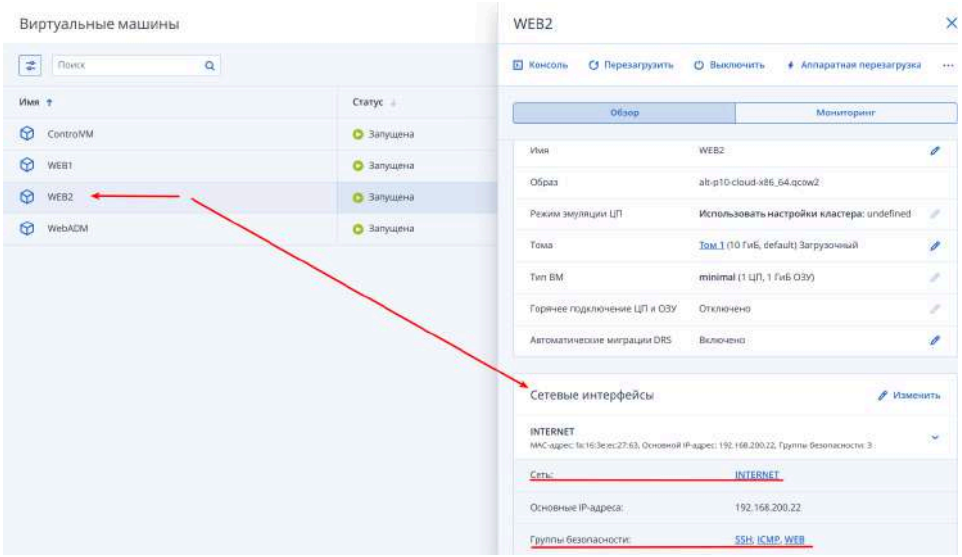
```
[altlinux@controlvm bin]$ openstack --insecure server list
+--------------------------------------+-----------+--------+-----------------------------------------------------------------------+--------------------------------+---------+
| ID                                   | Name      | Status | Networks                                                              | Image                          | Flavor  |
+--------------------------------------+-----------+--------+-----------------------------------------------------------------------+--------------------------------+---------+
| 74e2bb78-e034-463c-a2ad-ca0d843d47d3 | WEB1      | ACTIVE | INTERNET=192.168.15.176, 192.168.200.21                               | N/A (booted from volume)       | minimal |
| 88398bf3-2c63-48c9-817b-5ded169725c7 | WEB2      | ACTIVE | INTERNET=192.168.15.161, 192.168.200.22                               | N/A (booted from volume)       | minimal |
| c4ce7221-58a9-4804-a190-89f438526921 | WebADM    | ACTIVE | INTERNET=192.168.15.99, 192.168.200.20                                | N/A (booted from volume)       | minimal |
| 96306ddf-a488-4e35-8a8f-25696821efe5 | ControlVM | ACTIVE | INTERNET=192.168.200.10; cloud=192.168.100.128, 192.168.15.97         | N/A (booted from volume)       | medium  |
+--------------------------------------+-----------+--------+-----------------------------------------------------------------------+--------------------------------+---------+
[altlinux@controlvm bin]$
```

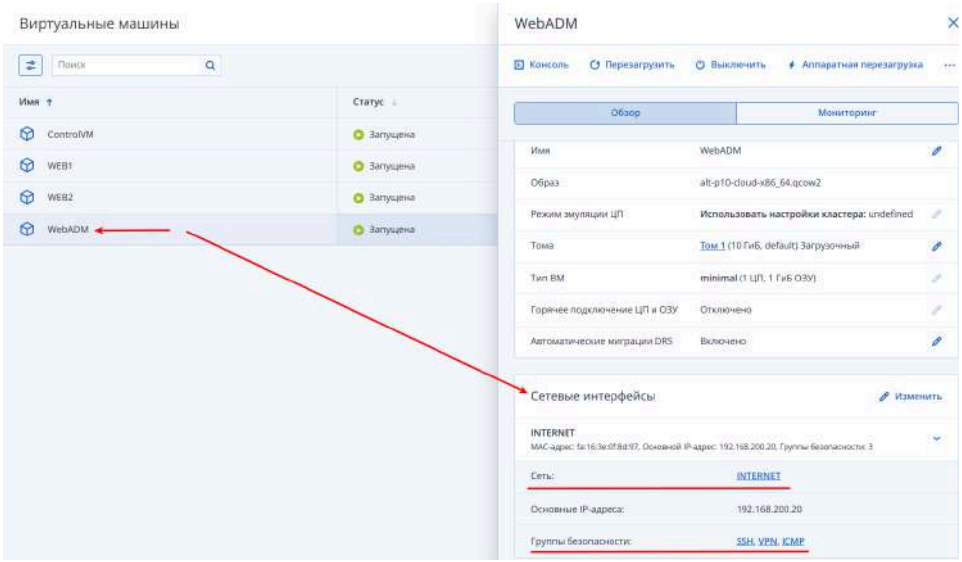
- Проверяем наличие созданных инстансов средствами **веб-интерфейса**:



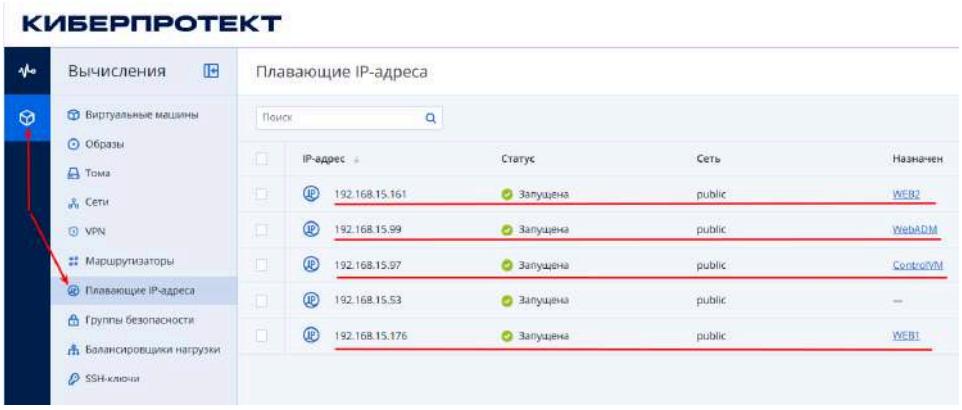
- Также проверяем что:
  - Каждый инстанс подключён к сети **INTERNET** с неоходимыми группами безопасности:







- Каждый Плавающий IP-адрес назначем на инстансы (за исключением 1-го для балансировки нагрузки):



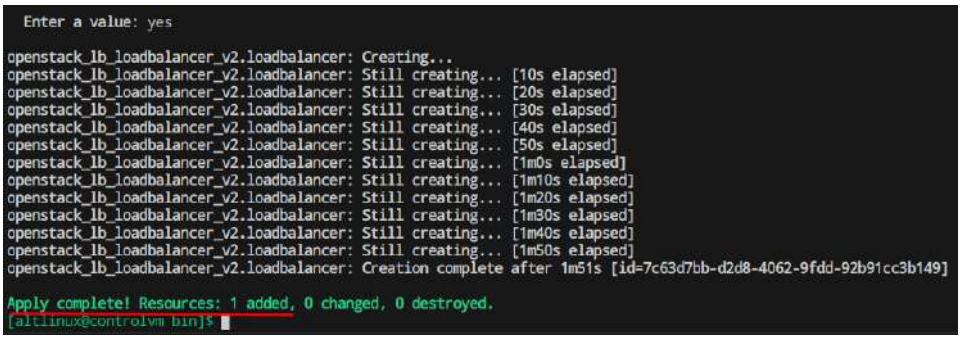
- Создаём файл **loadbalancer.tf** и описываем **последовательно** часть для развёртывания балансировщика нагрузки с **поэтапным запуском** и **наблюдением** созданных ресурсов:

```
vim loadbalancer.tf
```

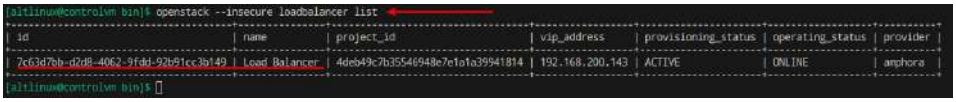
  - Помещаем следующее содержимое:
    - Создадим балансировщик нагрузки, который будет подключён в соответствие с топологией к подсети **INTERNET**;

```
# Создаём балансировщик нагрузки с именем "Load Balancer"
resource "openstack_lb_loadbalancer_v2" "loadbalancer" {
  name          = "Load Balancer"
  vip_subnet_id = openstack_networking_subnet_v2.subnet.id

  depends_on = [
    openstack_compute_instance_v2.web1,
    openstack_compute_instance_v2.web2
   ]
}
```

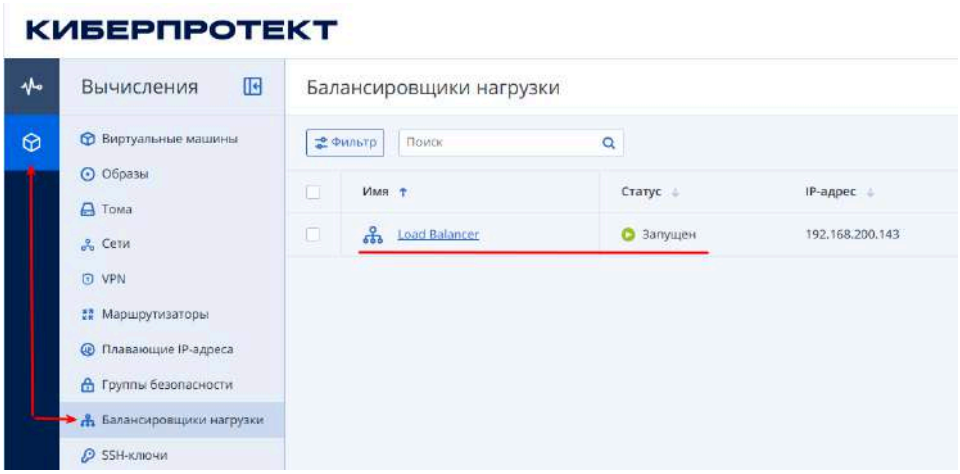
- Проверяем конфигурацию и план **terraform validate** и **terraform plan** и запускаем развёртывание ресурсов **terraform apply**:
  - Результат:



- Проверяем наличие созданного балансировщика нагрузки средствами **openstack-cli**:



- Проверяем наличие созданного балансировщика нагрузки средствами **веб-интерфейса**:



- В файл **loadbalancer.tf** добавляем следующий код:
  - Определяя правила в ранее созданном балансировщике нагрузки для HTTP и HTTPS трафика;

```
# Создаём правило в балансировщике нагрузки для HTTP
resource "openstack_lb_listener_v2" "listener_http" {
  name            = "HTTP"
  protocol        = "TCP"
  protocol_port   = "80"
  loadbalancer_id = openstack_lb_loadbalancer_v2.loadbalancer.id
}

# Создаём правило в балансировщике нагрузки для HTTPS
resource "openstack_lb_listener_v2" "listener_https" {
  name            = "HTTPS"
  protocol        = "TCP"
  protocol_port   = "443"
  loadbalancer_id = openstack_lb_loadbalancer_v2.loadbalancer.id
}
```

- Проверяем конфигурацию и план **terraform validate** и **terraform plan** и запускаем развёртывание ресурсов **terraform apply**:
  - Результат:

```
  Enter a value: yes

openstack_lb_listener_v2.listener_https: Creating...
openstack_lb_listener_v2.listener_http: Creating...
openstack_lb_listener_v2.listener_https: Still creating... [10s elapsed]
openstack_lb_listener_v2.listener_http: Still creating... [10s elapsed]
openstack_lb_listener_v2.listener_https: Creation complete after 14s [id=8955b49d-1201-4a4c-ad2d-1105a2b9728a]
openstack_lb_listener_v2.listener_http: Creation complete after 18s [id=a087d0b8-bcf5-40b5-b5a3-1f6b3ec942a5]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
[altlinux@controlvm bin]$
```

- В файл **loadbalancer.tf** добавляем следующий код:
  - Создавая целевые группы для HTTP и HTTPS указывая соответствующий протокол и алгоритм балансировки в соответствие с заданием **round robin**;

```
# Создаём целевую группу для HTTP с указанием алгоритма балансировки "ROUND ROBIN"
resource "openstack_lb_pool_v2" "pool_http" {
  name        = "HTTP"
  protocol    = "HTTP"
  lb_method   = "ROUND_ROBIN"
  listener_id = openstack_lb_listener_v2.listener_http.id
}

# Создаём целевую группу для HTTPS с указанием алгоритма балансировки "ROUND ROBIN"
resource "openstack_lb_pool_v2" "pool_https" {
  name        = "HTTPS"
  protocol    = "HTTPS"
  lb_method   = "ROUND_ROBIN"
  listener_id = openstack_lb_listener_v2.listener_https.id
}
```

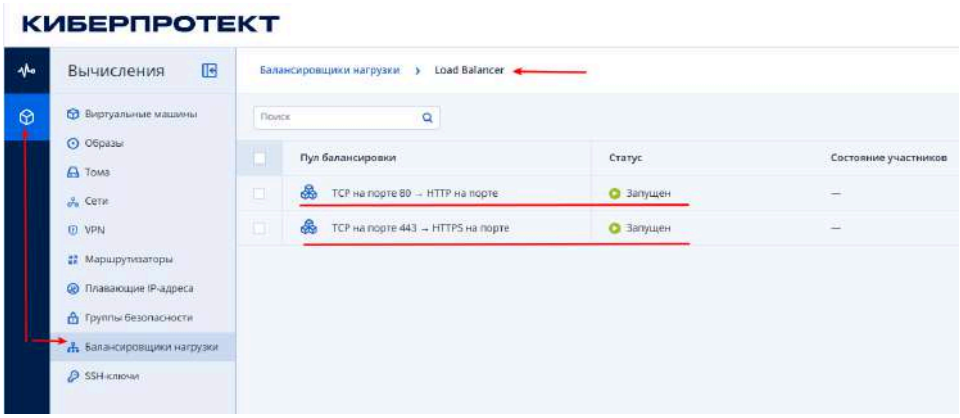
- Проверяем конфигурацию и план **terraform validate** и **terraform plan** и запускаем развёртывание ресурсов **terraform apply**:
  - Результат:

```
  Enter a value: yes

openstack_lb_pool_v2.pool_https: Creating...
openstack_lb_pool_v2.pool_http: Creating...
openstack_lb_pool_v2.pool_https: Creation complete after 8s [id=d77602e2-34a9-4422-86dc-a795652eb21b]
openstack_lb_pool_v2.pool_http: Creation complete after 8s [id=924b756e-d0a6-4ea5-9f5e-730241dbd0ce]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
[altlinux@controlvm bin]$
```

- Проверяем созданные целевые группы средствами **openstack-cli**:

```
[altlinux@controlvm bin]$ openstack --insecure loadbalancer pool list
+--------------------------------------+-------+----------------------------------+---------------------+----------+---------------+----------------+
| id                                   | name  | project_id                       | provisioning_status | protocol | lb_algorithm  | admin_state_up |
+--------------------------------------+-------+----------------------------------+---------------------+----------+---------------+----------------+
| d77602e2-34a9-4422-86dc-a795652eb21b | HTTPS | 4deb49c7b35546948e7e1a1a39941814 | ACTIVE              | HTTPS    | ROUND_ROBIN   | True           |
| 924b756e-d0a6-4ea5-9f5e-730241dbd0ce | HTTP  | 4deb49c7b35546948e7e1a1a39941814 | ACTIVE              | HTTP     | ROUND_ROBIN   | True           |
+--------------------------------------+-------+----------------------------------+---------------------+----------+---------------+----------------+
[altlinux@controlvm bin]$
```

- Проверяем созданные целевые группы средствами **веб-интерфейса**:



- В файл **loadbalancer.tf** добавляем следующий код:
  - Добавляя инстансы **WEB1** и **WEB2** в ранее созданные целевые группы для HTTP и HTTPS;

```
# Добавляем инстанс WEB1  в целевую группу HTTP
resource "openstack_lb_member_v2" "member_web1_http" {
  name          = "WEB1"
  subnet_id     = openstack_networking_subnet_v2.subnet.id
  pool_id       = openstack_lb_pool_v2.pool_http.id
  address       = "192.168.200.21"
  protocol_port = "80"
}

# Добавляем инстанс WEB2  в целевую группу HTTP
resource "openstack_lb_member_v2" "member_web2_http" {
  name          = "WEB2"
  subnet_id     = openstack_networking_subnet_v2.subnet.id
  pool_id       = openstack_lb_pool_v2.pool_http.id
  address       = "192.168.200.22"
  protocol_port = "80"
}

# Добавляем инстанс WEB1  в целевую группу HTTPS
resource "openstack_lb_member_v2" "member_web1_https" {
  name          = "WEB1"
  subnet_id     = openstack_networking_subnet_v2.subnet.id
  pool_id       = openstack_lb_pool_v2.pool_https.id
  address       = "192.168.200.21"
  protocol_port = "80"
}

# Добавляем инстанс WEB2  в целевую группу HTTPS
resource "openstack_lb_member_v2" "member_web2_https" {
  name          = "WEB2"
  subnet_id     = openstack_networking_subnet_v2.subnet.id
  pool_id       = openstack_lb_pool_v2.pool_https.id
  address       = "192.168.200.22"
  protocol_port = "80"
}
```

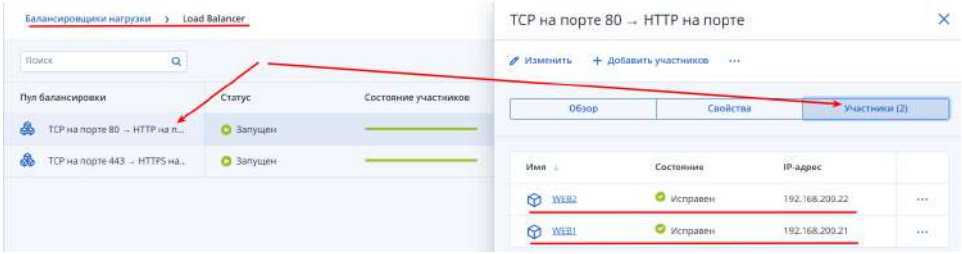
- Проверяем конфигурацию и план **terraform validate** и **terraform plan** и запускаем развёртывание ресурсов **terraform apply**:
  - Результат:

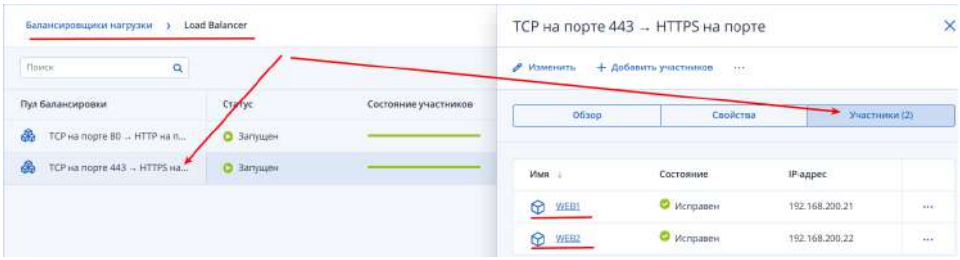
```
  Enter a value: yes

openstack_lb_member_v2.member_web1_http: Creating...
openstack_lb_member_v2.member_web1_https: Creating...
openstack_lb_member_v2.member_web2_https: Creating...
openstack_lb_member_v2.member_web2_http: Creating...
openstack_lb_member_v2.member_web1_http: Still creating... [10s elapsed]
openstack_lb_member_v2.member_web1_https: Still creating... [10s elapsed]
openstack_lb_member_v2.member_web2_https: Still creating... [10s elapsed]
openstack_lb_member_v2.member_web2_http: Still creating... [10s elapsed]
openstack_lb_member_v2.member_web2_https: Creation complete after 18s [id=3af7c156-4ab3-499f-8439-5ade4c52a664]
openstack_lb_member_v2.member_web1_https: Creation complete after 18s [id=161a06b6-0dd5-4b99-978d-ffd75935e7b2]
openstack_lb_member_v2.member_web1_http: Still creating... [20s elapsed]
openstack_lb_member_v2.member_web2_http: Still creating... [20s elapsed]
openstack_lb_member_v2.member_web2_http: Creation complete after 22s [id=3ed740cd-8ce4-4222-ab32-fc27b8fd86dc]
openstack_lb_member_v2.member_web1_http: Creation complete after 24s [id=12e61e9f-2925-4841-b98c-930a6c628457]

Apply complete! Resources: 4 added, 0 changed, 0 destroyed.
[altlinux@controlvm bin]$
```

- Проверяем наличие инстансов в целевых группах средствами **openstack-cli**:

```
[altlinux@controlvm bin]$ openstack --insecure loadbalancer member list HTTP
+--------------------------------------+------+----------------------------------+---------------------+--------------+---------------+------------------+--------+
| id                                   | name | project_id                       | provisioning_status | address      | protocol_port | operating_status | weight |
+--------------------------------------+------+----------------------------------+---------------------+--------------+---------------+------------------+--------+
| 3ed740cd-8ce4-4222-ab32-fc27b8fd86dc | WEB2 | 4deb49c7b35546948e7e1a1a39941814 | ACTIVE              | 192.168.200.22 |            80 | NO_MONITOR       |      1 |
| 12e61e9f-2925-4841-b98c-930a6c628457 | WEB1 | 4deb49c7b35546948e7e1a1a39941814 | ACTIVE              | 192.168.200.21 |            80 | NO_MONITOR       |      1 |
+--------------------------------------+------+----------------------------------+---------------------+--------------+---------------+------------------+--------+
[altlinux@controlvm bin]$ openstack --insecure loadbalancer member list HTTPS
+--------------------------------------+------+----------------------------------+---------------------+--------------+---------------+------------------+--------+
| id                                   | name | project_id                       | provisioning_status | address      | protocol_port | operating_status | weight |
+--------------------------------------+------+----------------------------------+---------------------+--------------+---------------+------------------+--------+
| 161a06b6-0dd5-4b99-978d-ffd75935e7b2 | WEB1 | 4deb49c7b35546948e7e1a1a39941814 | ACTIVE              | 192.168.200.21 |            80 | NO_MONITOR       |      1 |
| 3af7c156-4ab3-499f-8439-5ade4c52a664 | WEB2 | 4deb49c7b35546948e7e1a1a39941814 | ACTIVE              | 192.168.200.22 |            80 | NO_MONITOR       |      1 |
+--------------------------------------+------+----------------------------------+---------------------+--------------+---------------+------------------+--------+
```

- Проверяем наличие инстансов в целевых группах средствами **веб-интерфейса**:

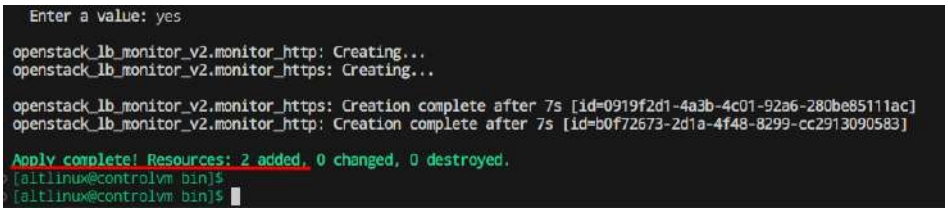




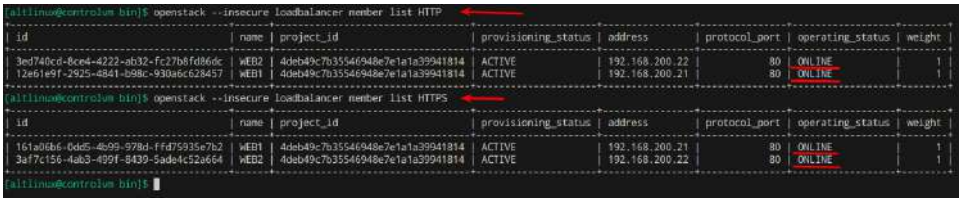
- В файл **loadbalancer.tf** добавляем следующий код:
  - Добавляя проверку доступности инстансов в каждой целевой группе;

```
# Создаём проверку доступности инстансов в целевой группе HTTP
resource "openstack_lb_monitor_v2" "monitor_http" {
  name        = "monitor HTTP"
  pool_id     = openstack_lb_pool_v2.pool_http.id
  type        = "PING"
  delay       = "10"
  timeout     = "4"
  max_retries = "5"
}

# Создаём проверку доступности инстансов в целевой группе HTTPS
resource "openstack_lb_monitor_v2" "monitor_https" {
  name        = "monitor HTTPS"
  pool_id     = openstack_lb_pool_v2.pool_https.id
  type        = "PING"
  delay       = "10"
  timeout     = "4"
  max_retries = "5"
}
```

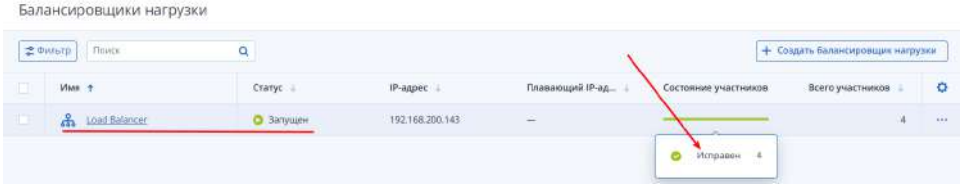
- Проверяем конфигурацию и план **terraform validate** и **terraform plan** и запускаем развёртывание ресурсов **terraform apply**:
  - Результат:



- Проверяем доступность мониторинга инстансов в целевых группах средствами **openstack-cli**:



- Проверяем доступность мониторинга инстансов в целевых группах средствами **веб-интерфейса**:



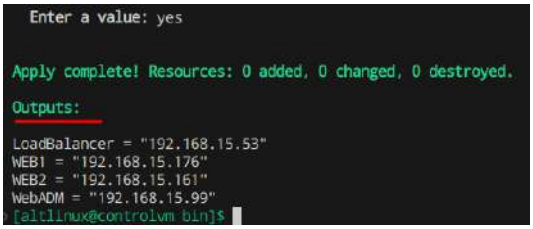
- Создаём файл **output.tf**:

```
vim output.tf
```

  - Помещаем следующее содержимое:
    - Для печати на экран "внешний (публичные)" плавающие IP-адреса всех инстансов;
    - Понадобится в дальнейшем для сохранения вывода в файл по требованиям задания;

```
output "WebADM" {
  value = openstack_networking_floatingip_v2.floatingip_webadm.address
  depends_on = [ openstack_compute_instance_v2.webadm ]
}

output "WEB1" {
  value = openstack_networking_floatingip_v2.floatingip_web1.address
  depends_on = [ openstack_compute_instance_v2.web1 ]
}

output "WEB2" {
  value = openstack_networking_floatingip_v2.floatingip_web2.address
  depends_on = [ openstack_compute_instance_v2.web2 ]
}

output "LoadBalancer" {
  value = openstack_networking_floatingip_v2.floatingip_loadbalancer.address
}
```

- Проверяем конфигурацию и план **terraform validate** и **terraform plan** и запускаем развёртывание ресурсов **terraform apply**:
  - Результат:



- На текущий момент вся инфраструктура развёрнута в соответствие с требованиями задания, за исключением настройки VPN-туннеля и SSH по паролю в рамках туннельного соединения
- Реализуем недостающий функционал средствами связки **Terraform** и **Ansible**

- Создаём файл **templates.tf**:

```
vim templates.tf
```

  - Помещаем следующее содержимое:
    - Поскольку нельзя предугадать какие IP-адреса будут выданы инстансам в роли "внешних" (Плавающих) реализуем возможность генерации автоматического инвентаря для **Ansible** средствами **Terraform;**
    - Т.к. в конечном варианте ожидается запуск единого скрипта который развёрнёт инфраструктуру а затем её настроит;

```
data "template_file" "inventory" {
    template = file("/home/altlinux/bin/_templates/inventory.tpl")

    vars = {
        user = "altlinux"
        webadm = join("", [openstack_compute_instance_v2.webadm.name, " ansible_host=",
openstack_networking_floatingip_v2.floatingip_webadm.address])
        web1 = join("", [openstack_compute_instance_v2.web1.name, " ansible_host=",
openstack_networking_floatingip_v2.floatingip_web1.address])
        web2 = join("", [openstack_compute_instance_v2.web2.name, " ansible_host=",
openstack_networking_floatingip_v2.floatingip_web2.address])
    }
}

resource "local_file" "save_inventory" {
   content  = data.template_file.inventory.rendered
   filename = "/home/altlinux/bin/ansible/inventory"
}
```

- Создаём директорию "**_templates**" где будет хранить шаблоны:

```
mkdir _templates
```

- Создаём сам файл шаблона для автоматической генерации инвентаря для **Ansible**:

```
vim _templates/inventory.tpl
```

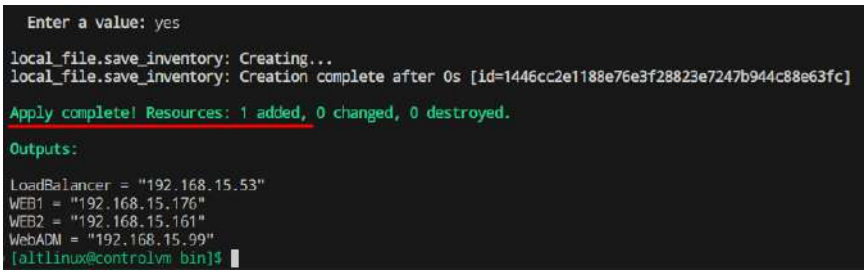
  - Помещаем в него следующее содержимое:
    - В результате **Terraform** должен будет генерировать инвентарь для Ansible по пути **/home/altlinux/bin/ansible/inventory** описанный в ini-формате;

```
${webadm}
${web1}
${web2}

[all:vars]
ansible_ssh_user = ${user}
ansible_ssh_extra_args = '-o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no'
ansible_python_interpreter = /usr/bin/python3
```

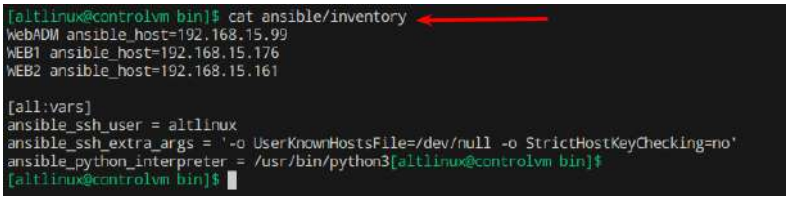
- Создаём директорию для **Ansible**:

```
mkdir ansible
```

- Проверяем конфигурацию и план **terraform validate** и **terraform plan** и запускаем развёртывание ресурсов **terraform apply**:
  - Результат:



- Проверяем наличие файла **/home/altlinux/bin/ansible/inventory** и его содержимое:



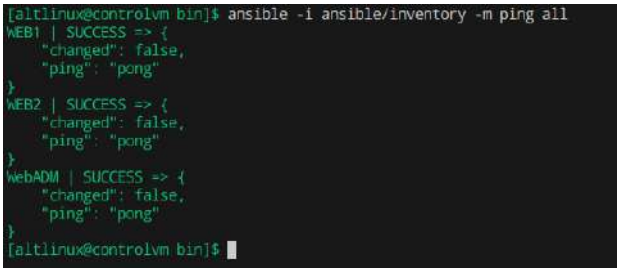
- Устанавливаем **ansible**:

```
sudo apt-get update && sudo apt-get install -y ansible
```

- Проверяем корректность сгенерированного инвентарного файла:

```
ansible -i ansible/inventory -m ping all
```

  - Результат:
    - **Ansible** успешно может подключиться с **ControlVM** до всех инстансов для дальнейшей их настройки;



- В файл **templates.tf** добавляем следующий код:
  - Добавляя возможность **Terraform** автоматически генерировать конфигурационные файлы для **WireGuard**;
  - Поскольку нет информации о "внешнем" (Плавающим) IP-адресе который будет назначем **WebADM** при новом развёртывании проекта;
  - Он необходим для конфигурационного файла **WireGuard** для параметра **Endpoint**;

```
data "template_file" "web1_wg0" {
    template =  file("/home/altlinux/bin/_templates/web1_wg0.tpl")

    vars = {
      floatingip_webadm = join("",[openstack_networking_floatingip_v2.floatingip_webadm.address])
    }
}

resource "local_file" "save_web1_wg0" {
   content  = data.template_file.web1_wg0.rendered
   filename = "/home/altlinux/bin/ansible/wireguard/web1_wg0.conf"
}

data "template_file" "web2_wg0" {
    template =  file("/home/altlinux/bin/_templates/web2_wg0.tpl")

    vars = {
      floatingip_webadm = join("",[openstack_networking_floatingip_v2.floatingip_webadm.address])
    }
}

resource "local_file" "save_web2_wg0" {
   content  = data.template_file.web2_wg0.rendered
   filename = "/home/altlinux/bin/ansible/wireguard/web2_wg0.conf"
}
```

- Создаём сам файл шаблона конфигурационного файла туннельного соединения для инстанса **WEB1**:

```
vim _templates/web1_wg0.tpl
```

    - Помещаем в него следующее содержимое:

```
[Interface]
Address = 10.20.30.2/29
PrivateKey = +OoP8y4QuTZL7P5Esr7YV7p/GqleqSTE5Mf6R1E6y38=

# WebAdm
[Peer]
PublicKey = Pn1bzrpZoBob/VeAvWDQfazwrZh18WarKSKzid2K1wc=
AllowedIPs = 10.20.30.0/29
PersistentKeepalive = 10
Endpoint = ${floatingip_webadm}:51820
```

- Создаём сам файл шаблона конфигурационного файла туннельного соединения для инстанса **WEB2**:

```
vim _templates/web2_wg0.tpl
```

    - Помещаем в него следующее содержимое:

```
[Interface]
Address = 10.20.30.3/29
PrivateKey = UEN3Lbpm+NyXsw+XMEZPKHAaMocFRP0KfDZvIiu+BUk=

# WebAdm
[Peer]
PublicKey = Pn1bzrpZoBob/VeAvWDQfazwrZh18WarKSKzid2K1wc=
AllowedIPs = 10.20.30.0/29
PersistentKeepalive = 10
Endpoint = ${floatingip_webadm}:51820
```

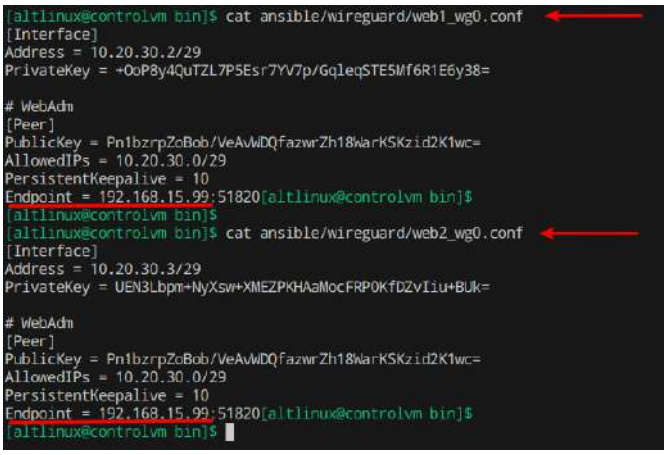
- Создаём директорию для куда **Terraform** сохранит сгенерированные конфигурационные файлы:

```
mkdir ansible/wireguard
```

- Проверяем конфигурацию и план **terraform validate** и **terraform plan** и запускаем развёртывание ресурсов **terraform apply**:
    - Результат:

```
  Enter a value: yes

local_file.save_web1_wg0: Creating...
local_file.save_web1_wg0: Creation complete after 0s [id=e48082e72e6a45f68c784df5a548a741909d4fd1]
local_file.save_web2_wg0: Creating...
local_file.save_web2_wg0: Creation complete after 0s [id=aae9a96c0222f4de114f5dc2b787ed93711b2127]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

Outputs:

LoadBalancer = "192.168.15.53"
WEB1 = "192.168.15.176"
WEB2 = "192.168.15.161"
WebADM = "192.168.15.99"
[altlinux@controlvm bin]$
```

- Проверяем наличие конфигурационных файлов для туннельных соединений и их содержимое:
    - Должен корректно подставляться параметр **Endpoint**:



- Создаём конфигурационный файл туннельного соединения для инстанса **WebADM**:

```
vim ansible/wireguard/webadm_wg0.conf
```

  - Помещаем в него следующее содержимое:
    - Т.к. с точки зрения **WireGuard** инстанс **WebADM** будет выступать сервером, то данный файл может бьть статическим и не требуем какой-либо динамически получаемой информации;

```
[Interface]
Address = 10.20.30.1/29
ListenPort = 51820
PrivateKey = wHpaZtEfAYGxVXhwzUa8OS1EbBXqrjDW8E1Z73pzTUI=

# WEB1
[Peer]
PublicKey = dZuyzPZVpJ4gyvMpJRCOI3PBGEIYe1x1MZuYyYHjfT0=
AllowedIPs = 10.20.30.2/32

# WEB2
[Peer]
PublicKey = ZSTT8CxQsxJ1KU9bO5PoSG6iLFBb8hQtIc4Kyox4AnU=
AllowedIPs = 10.20.30.3/32
```

- Создаём **playbook** который будет реализовывать настройку VPN-соединения средствами **WireGuard** между **WebADM** и **WEB1, WEB2**:

```
vim ansible/wireguard_playbook.yml
```

  - Помещаем в него следующее содержимое:

```
---
- name: Install packages
  hosts: all
  become: true

  tasks:
    - name: Install Wireguard
      community.general.apt_rpm:
        name:
          - wireguard-tools
          - wireguard-tools-wg-quick
        state: latest
        update_cache: true

- name: Settings WebAdm Wireguard server
  hosts: WebADM
  become: true

  tasks:
    - name: Create directory '/etc/wireguard'
      ansible.builtin.file:
        path: /etc/wireguard
        state: directory
        mode: '0755'

    - name: Copy file 'wg0.conf'
      ansible.builtin.copy:
        src: wireguard/webadm_wg0.conf
        dest: /etc/wireguard/wg0.conf

    - name: Started and enabled wg0
      ansible.builtin.systemd:
        name: wg-quick@wg0
        state: started
        enabled: true

- name: Settings WEB1 Wireguard client
  hosts: WEB1
  become: true

  tasks:
    - name: Create directory '/etc/wireguard'
      ansible.builtin.file:
        path: /etc/wireguard
        state: directory
        mode: '0755'

    - name: Copy file 'wg0.conf'
      ansible.builtin.copy:
        src: wireguard/web1_wg0.conf
        dest: /etc/wireguard/wg0.conf

    - name: Started and enabled wg0
      ansible.builtin.systemd:
        name: wg-quick@wg0
        state: started
        enabled: true

- name: Settings WEB2 Wireguard client
  hosts: WEB2
  become: true

  tasks:
    - name: Create directory '/etc/wireguard'
      ansible.builtin.file:
        path: /etc/wireguard
        state: directory
        mode: '0755'

    - name: Copy file 'wg0.conf'
      ansible.builtin.copy:
        src: wireguard/web2_wg0.conf
        dest: /etc/wireguard/wg0.conf

    - name: Started and enabled wg0
      ansible.builtin.systemd:
        name: wg-quick@wg0
        state: started
        enabled: true
```

- Устанавливаем коллекцию необходимую для работы модуля **community.general.apt_rpm:**

```
ansible-galaxy collection install community.general
```

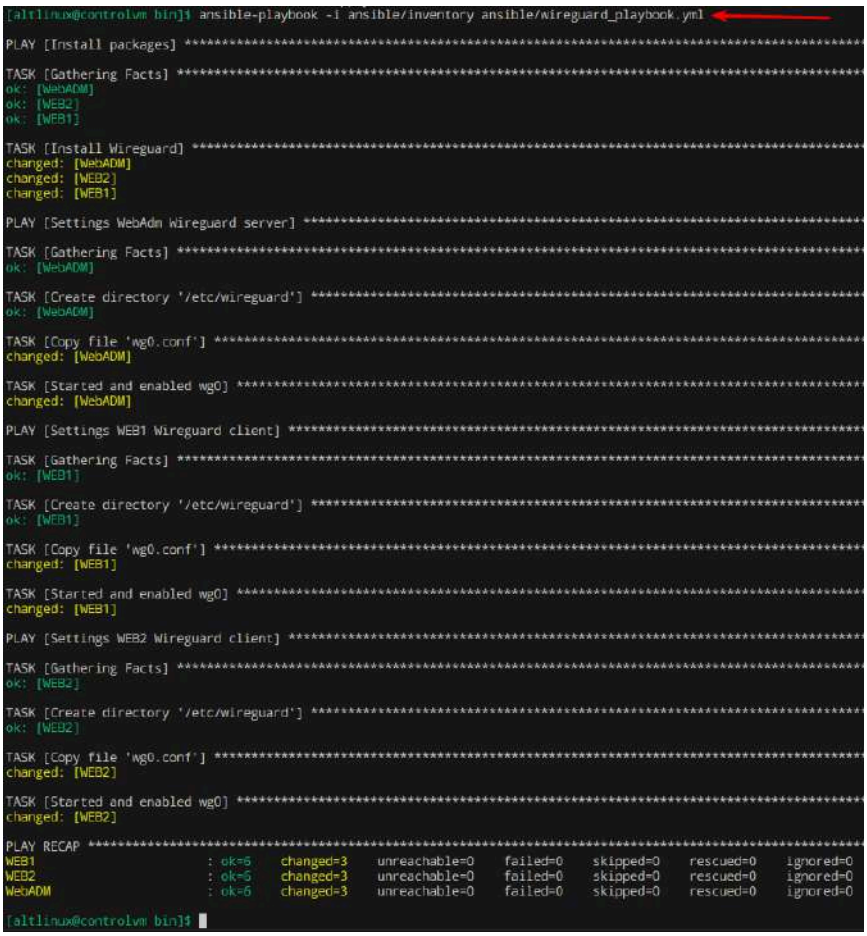
    - Результат:

```
[altlinux@controlvm bin]$ ansible-galaxy collection install community.general
Process install dependency map
Starting collection install process
Installing 'community.general:10.2.0' to '/home/altlinux/.ansible/collections/ansible_collections/community/general'
[altlinux@controlvm bin]$
```

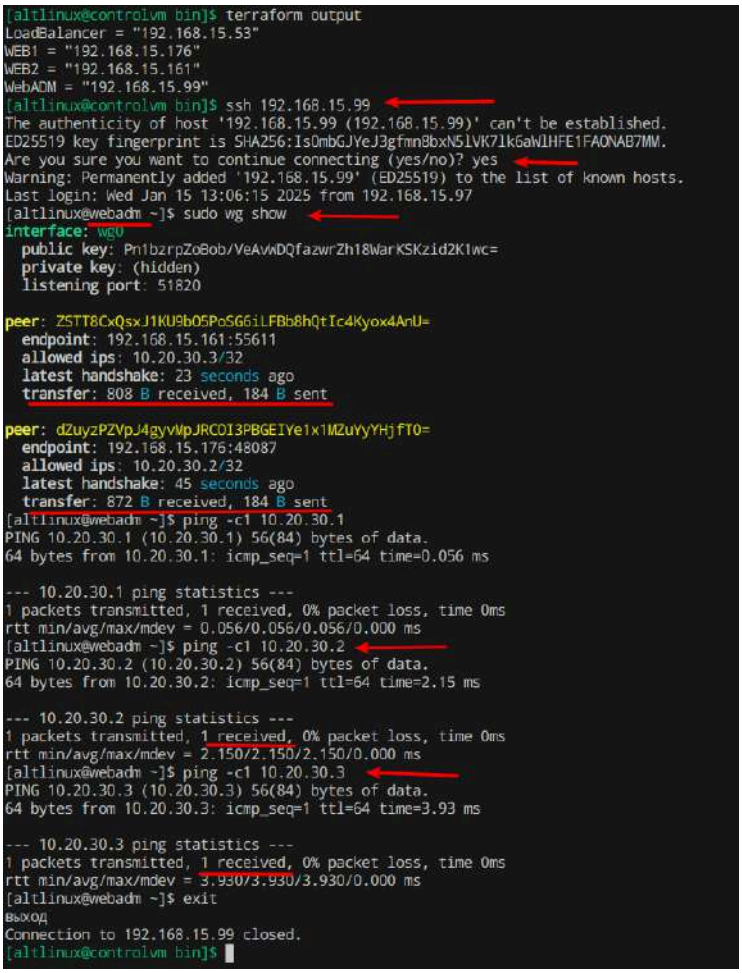
- Запускаем playbook-сценарий для настройки VPN-туннеля:

```
ansible-playbook -i ansible/inventory ansible/wireguard_playbook.yml
```

  - Результат:



- Проверяем наличие VPN-соединения:
  - Подключаемся по SSH с **ControlVM** на **WebADM** по Плавающему IP-адресу;
  - Смотрим туннельные соединения;
  - Проверяем связность;



- Создаём **playbook** который будет реализовывать:
  - **WebADM** так, чтобы она могла подключаться по SSH с использованием пользователя **altlinux** и пароля «**P@ssw0rd**» к инстансам **WEB1** и **WEB2** с помощью VPN туннеля

```
vim ansible/ssh_playbook.yml
```

  - Помещаем в него следующее содержимое:

```
---
- hosts: WEB1 WEB2
  become: true

  tasks:
    - name: Setting 'sshd_config' file
      ansible.builtin.lineinfile:
        line: "{{ item }}"
        path: /etc/openssh/sshd_config
        state: present
      with_items:
        - "PasswordAuthentication no"
        - "Match address 10.20.30.0/29"
        - "    PasswordAuthentication yes"

    - name: Restarted sshd
      ansible.builtin.systemd:
        name: sshd
        state: restarted
```

- Запускаем playbook-сценарий для настройки SSH по VPN-туннелю:

```
ansible-playbook -i ansible/inventory ansible/ssh_playbook.yml
```

  - Результат:

```
[altlinux@controlvm bin]$ ansible-playbook -i ansible/inventory ansible/ssh_playbook.yml

PLAY [WEB1 WEB2] ***************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************
ok: [WEB2]
ok: [WEB1]

TASK [Setting 'sshd_config' file] **********************************************************************************
ok: [WEB1] => (item=PasswordAuthentication no)
ok: [WEB2] => (item=PasswordAuthentication no)
changed: [WEB2] => (item=Match address 10.20.30.0/29)
changed: [WEB1] => (item=Match address 10.20.30.0/29)
changed: [WEB1] => (item=    PasswordAuthentication yes)
changed: [WEB2] => (item=    PasswordAuthentication yes)

TASK [Restarted sshd] **********************************************************************************************
changed: [WEB1]
changed: [WEB2]

PLAY RECAP *********************************************************************************************************
WEB1                       : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
WEB2                       : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[altlinux@controlvm bin]$
```

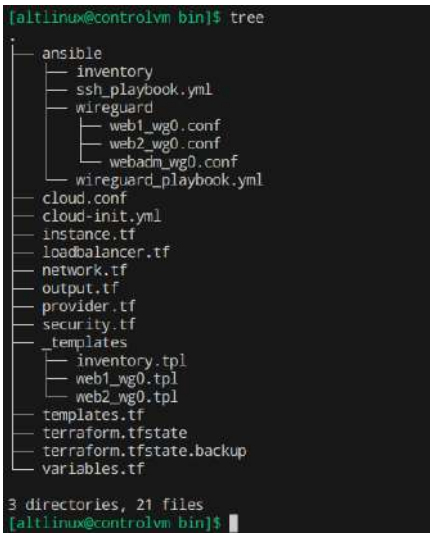
- Проверяем:
  - Подключение с **WebADM** по туннельным IP-адресам на основе пароля:

```
[altlinux@controlvm bin]$ ssh 192.168.15.99
Last login: Wed Jan 15 13:09:21 2025 from 192.168.15.97
[altlinux@webadm ~]$ ssh 10.20.30.2
The authenticity of host '10.20.30.2 (10.20.30.2)' can't be established.
ED25519 key fingerprint is SHA256:JT/BUDKbWVUqOPTOV9rGmN+DpyDwfBcITbasbFvGXM.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.20.30.2' (ED25519) to the list of known hosts.
altlinux@10.20.30.2's password:
Last login: Wed Jan 15 13:13:58 2025 from 192.168.15.97
[altlinux@web1 ~]$
[altlinux@web1 ~]$ exit
выход
Connection to 10.20.30.2 closed.
[altlinux@webadm ~]$
[altlinux@webadm ~]$ ssh 10.20.30.3
The authenticity of host '10.20.30.3 (10.20.30.3)' can't be established.
ED25519 key fingerprint is SHA256:bdhrhxsPT2SEeT9qkeqUppEpyDAo7YvMzBqYsHb//gk.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.20.30.3' (ED25519) to the list of known hosts.
altlinux@10.20.30.3's password:
Last login: Wed Jan 15 13:13:59 2025 from 192.168.15.97
[altlinux@web2 ~]$ exit
выход
Connection to 10.20.30.3 closed.
[altlinux@webadm ~]$
[altlinux@webadm ~]$ exit
выход
Connection to 192.168.15.99 closed.
[altlinux@controlvm bin]$
```

- Также проверяем что подключение по SSH из внешних сете (не по туннелю) доступно только по ключу:

```
[altlinux@controlvm bin]$ terraform output
LoadBalancer = "192.168.15.53"
WEB1 = "192.168.15.176"
WEB2 = "192.168.15.161"
WebADM = "192.168.15.99"
[altlinux@controlvm bin]$ ssh 192.168.15.99
Last login: Wed Jan 15 13:14:57 2025 from 192.168.15.97
[altlinux@webadm ~]$ ssh 192.168.15.176
The authenticity of host '192.168.15.176 (192.168.15.176)' can't be established.
ED25519 key fingerprint is SHA256:JT/BUDKbWVUqOPTOV9rGmN+DpyDwfBcITbasbFvGXM.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.15.176' (ED25519) to the list of known hosts.
ssh: altlinux@192.168.15.176: Permission denied (publickey).
[altlinux@webadm ~]$ ssh 192.168.15.99
The authenticity of host '192.168.15.99 (192.168.15.99)' can't be established.
ED25519 key fingerprint is SHA256:IsOmbGJYeJ3gfmn8bxN5lVK7lk6aWlHFE1FAONAB7VM.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.15.99' (ED25519) to the list of known hosts.
ssh: altlinux@192.168.15.99: Permission denied (publickey).
[altlinux@webadm ~]$ exit
выход
Connection to 192.168.15.99 closed.
[altlinux@controlvm bin]$
```

- Таким образом, на текущий момент получаем следующую структуру файлов и каталогов в рамках каталога **/home/altlinux/bin** на **ControlVM**:



Последнее изменение: среда, 15 января 2025, 16:18