# **Задание 1**

**Выполните действия, приложите скриншот скриптов, скриншот выполненного проекта.**

От заказчика получено задание: при помощи Terraform и Ansible собрать виртуальную инфраструктуру и развернуть на ней веб-ресурс.

В инфраструктуре нужна одна машина с ПО ОС Linux, двумя ядрами и двумя гигабайтами оперативной памяти.

Требуется установить nginx, залить при помощи Ansible конфигурационные файлы nginx и веб-ресурса.

Секретный токен от yandex cloud должен вводится в консоли при каждом запуске terraform.

Для выполнения этого задания нужно сгенерировать SSH-ключ командой ssh-keygen. Добавить в конфигурацию Terraform ключ в поле:

metadata = {

user-data = "${file("./meta.txt")}"

}

В файле meta прописать:

users:

- name: user

groups: sudo

shell: /bin/bash

sudo: ['ALL=(ALL) NOPASSWD:ALL']

ssh-authorized-keys:

- ssh-rsa xxx

Где xxx — это ключ из файла /home/"name\_ user"/.ssh/id\_rsa.pub. Примерная конфигурация Terraform:

terraform {

required\_providers {

yandex = {

source = "yandex-cloud/yandex"

}

}

}

variable "yandex\_cloud\_token" {

type = string

description = "Введите секретный токен от yandex\_cloud"

}

provider "yandex" {

token = var.yandex\_cloud\_token #секретные данные должны быть в сохранности!! Никогда не выкладывайте токен в публичный доступ.

cloud\_id = "xxx"

folder\_id = "xxx"

zone = "ru-central1-a"

}

resource "yandex\_compute\_instance" "vm-1" {

name = "terraform1"

resources {

cores = 2

memory = 2

}

boot\_disk {

initialize\_params {

image\_id = "fd87kbts7j40q5b9rpjr"

}

}

network\_interface {

subnet\_id = yandex\_vpc\_subnet.subnet-1.id

nat = true

}

metadata = {

user-data = "${file("./meta.txt")}"

}

}

resource "yandex\_vpc\_network" "network-1" {

name = "network1"

}

resource "yandex\_vpc\_subnet" "subnet-1" {

name = "subnet1"

zone = "ru-central1-b"

network\_id = yandex\_vpc\_network.network-1.id

v4\_cidr\_blocks = ["192.168.10.0/24"]

}

output "internal\_ip\_address\_vm\_1" {

value = yandex\_compute\_instance.vm-1.network\_interface.0.ip\_address

}

output "external\_ip\_address\_vm\_1" {

value = yandex\_compute\_instance.vm-1.network\_interface.0.nat\_ip\_address

}

В конфигурации Ansible указать:

* внешний IP-адрес машины, полученный из output external\_ ip\_ address\_ vm\_1, в файле hosts;
* доступ в файле plabook \*yml поля hosts.

- hosts: 138.68.85.196

remote\_user: user

tasks:

- service:

name: nginx

state: started

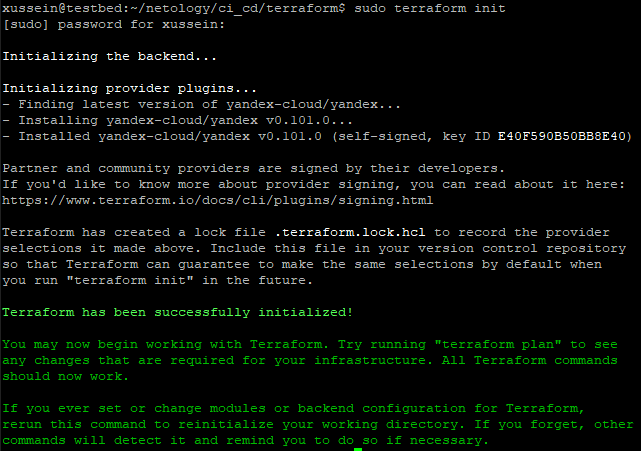
become: yes

become\_method: sudo

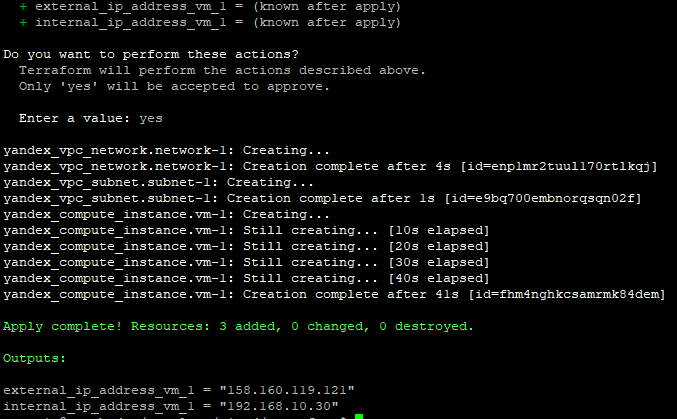
Провести тестирование.

# **Решение 1**

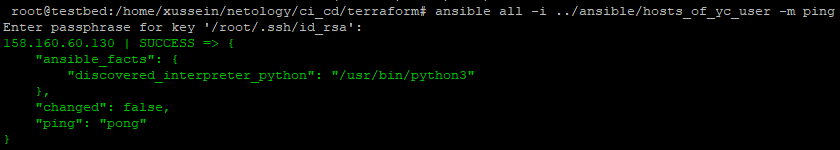
Инициализация терраформа:



После apply’я main.tf:



ВМ заработала. Теперь пинг через ансибл, для проверки (внешний адрес другой, да, несколько раз переустанавливал ВМки):



Запускаем следующие таски из роли (остальное не привожу так как роль в основном пустая, но беру за практику сразу создавать роль, а не сольные плейбуки):

---

# tasks file for yc\_nginx

- name: apt install nginx

apt:

name: nginx

state: present

- name: start nginx service

service:

name: nginx

state: started

- name: check website availability

uri:

url: http://localhost

method: GET

register: website\_response

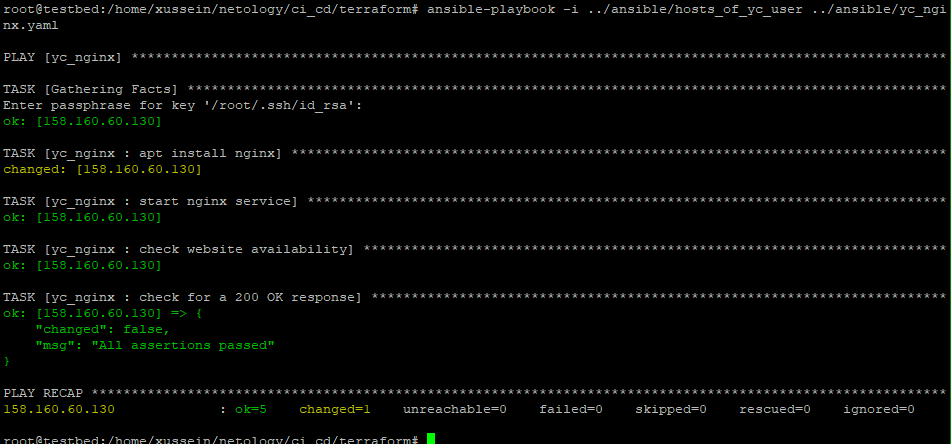
- name: check for a 200 OK response

assert:

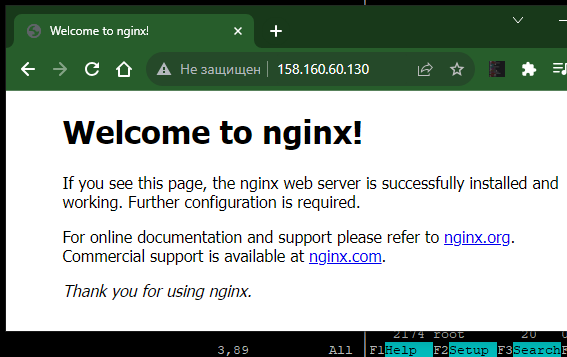
that: "website\_response.status == 200"

fail\_msg: "The website is unavailable or returned a response code: {{ website\_response.status }}"

Результат работы:



То есть nginx установился, запустился и прошел проверку на локалхосте и получил 200й ответ. Проверим, действительно ли он крутится:



Все ОК.

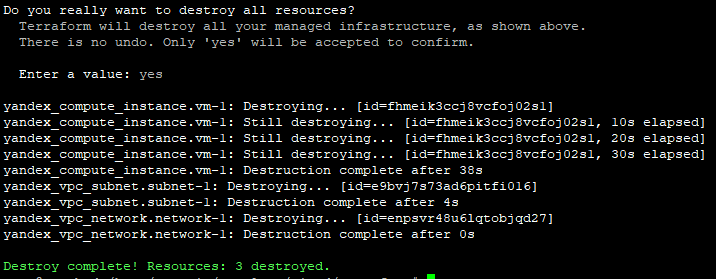
# **Задание 2**

Выполните действия, приложите скриншот скриптов, скриншот выполненного проекта.

1. Перестроить инфраструктуру и добавить в неё вторую виртуальную машину.
2. Установить на вторую виртуальную машину базу данных.
3. Выполнить проверку состояния запущенных служб через Ansible.

# **Решение 2**

Ломаем всё:



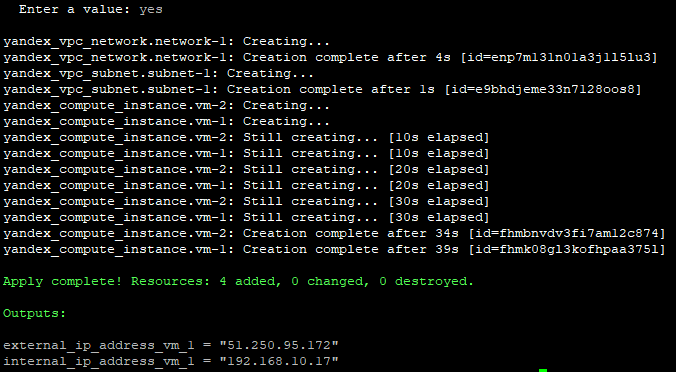
Новый main.tf:

1. terraform {
2. required\_providers {
3. yandex = {
4. source = "yandex-cloud/yandex"
5. }
6. }
7. }
8. variable "yandex\_cloud\_token" {
9. type = string
10. description = "Введите секретный токен от yandex\_cloud"
11. }
12. provider "yandex" {
13. token = var.yandex\_cloud\_token
14. cloud\_id = "b1grulihhlu8ubggpm9k"
15. folder\_id = "b1ghitn6n7m4ui0tnhfd"
16. zone = "ru-central1-a"
17. }
18. resource "yandex\_compute\_instance" "vm-1" {
19. name = "terraform1"
20. resources {
21. cores = 4
22. memory = 4
23. }
24. boot\_disk {
25. initialize\_params {
26. image\_id = "fd8go38kje4f6v3g2k4q"
27. }
28. }
29. network\_interface {
30. subnet\_id = yandex\_vpc\_subnet.subnet-1.id
31. nat = true
32. }
34. metadata = {
35. user-data = "${file("./meta.yaml")}"
36. }
37. }
38. resource "yandex\_compute\_instance" "vm-2" {
39. name = "terraform2"
40. resources {
41. cores = 2
42. memory = 2
43. }
44. boot\_disk {
45. initialize\_params {
46. image\_id = "fd8go38kje4f6v3g2k4q"
47. }
48. }
49. network\_interface {
50. subnet\_id = yandex\_vpc\_subnet.subnet-1.id
51. nat = true
52. }
54. metadata = {
55. user-data = "${file("./meta.yaml")}"
56. }
57. }
58. resource "yandex\_vpc\_network" "network-1" {
59. name = "network1"
60. }
61. resource "yandex\_vpc\_subnet" "subnet-1" {
62. name = "subnet1"
63. zone = "ru-central1-a"
64. network\_id = yandex\_vpc\_network.network-1.id
65. v4\_cidr\_blocks = ["192.168.10.0/24"]
66. }

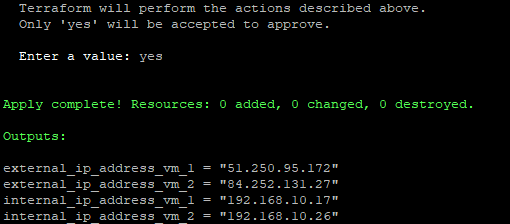
output.tf:

1. output "internal\_ip\_address\_vm\_1" {
2. value = yandex\_compute\_instance.vm-1.network\_interface.0.ip\_address
3. }
4. output "external\_ip\_address\_vm\_1" {
5. value = yandex\_compute\_instance.vm-1.network\_interface.0.nat\_ip\_address
6. }
7. output "internal\_ip\_address\_vm\_2" {
8. value = yandex\_compute\_instance.vm-2.network\_interface.0.ip\_address
9. }
10. output "external\_ip\_address\_vm\_2" {
11. value = yandex\_compute\_instance.vm-2.network\_interface.0.nat\_ip\_address
12. }

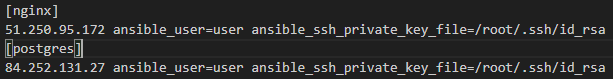
Результат:



Не вывел правильно адреса, исправляюсь (отдельно вынес output.tf и добавил вторую виртуалку):



Вносим внешние адреса в инвентарь ансибла:



Для ансибла создал вторую роль yc\_postgres. Обе файла тасков ролей:

# tasks file for yc\_nginx

- name: apt install nginx

apt:

name: nginx

state: present

- name: start nginx service

service:

name: nginx

state: started

- name: check website availability

uri:

url: http://localhost

method: GET

register: website\_response

- name: check for a 200 OK response

assert:

that: "website\_response.status == 200"

fail\_msg: "The website is unavailable or returned a response code: {{ website\_response.status }}"

---

# tasks file for yc\_postgres

- name: apt install postgres

apt:

name: postgresql

state: present

- name: start postgres service

service:

name: postgresql

state: started

- name: postgres status check

command: pg\_isready -h localhost -U postgres

ignore\_errors: yes

register: pg\_isready\_output

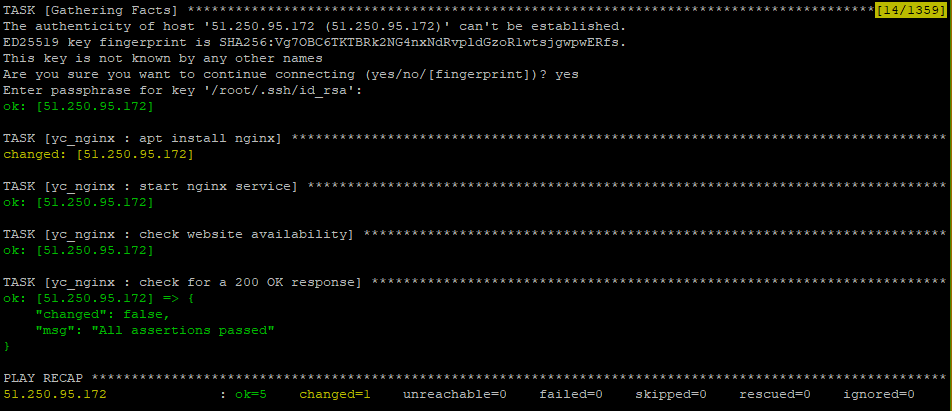
- name: postgres status output

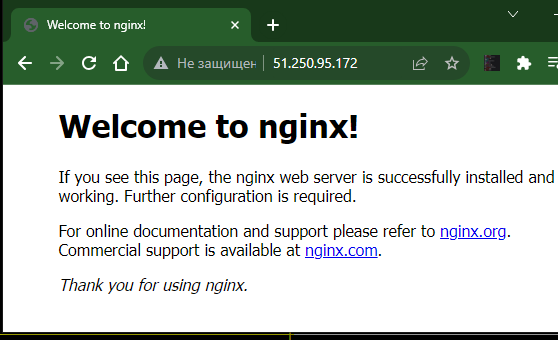
fail:

msg: "PostgreSQL не запущен"

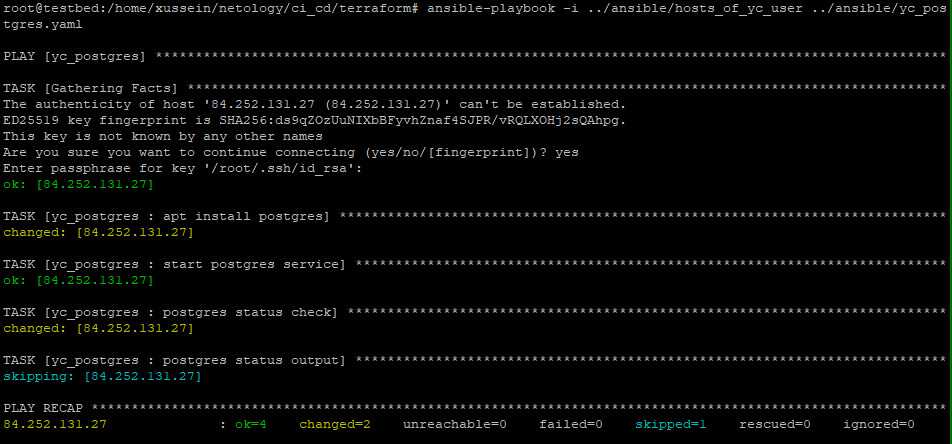
when: "'no response' in pg\_isready\_output.stdout"

nginx поднялся:

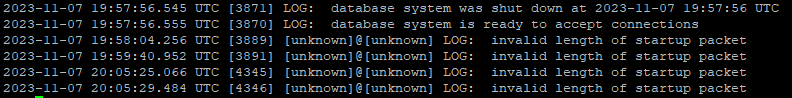




postgres поднялся:



Только с постгресом немного сжульничал, напрочь забыл что у него по-умолчанию закрыт доступ извне, только локалхост. То есть нужно править файлик /etc/postgres/какая-то-там-версия/что-то-там-main/postgresql.conf, добавить строку listen\_addresses = “\*”. То есть поправил ручками, так как уже не хотелось в очередной раз все пересобирать для красивой картинки на скринах 🙂. При обновленном файле конфы он легко телнетится, в логах это отражается:



Вообще там в роль добавляется таска вида:

- name: restart apache2 if config changes

blockinfile:

path: /etc/apache2/sites-available/000-default.conf

block: |

DirectoryIndex output.html

insertbefore: '^</VirtualHost>'

notify: restart apache2

С последующим перезапуском сервиса.

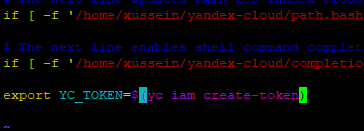
Но ночь уже глубокая, а завтра снова к станку. “Уж полночь близится, а Германа все нет”.

# **Задание 3**

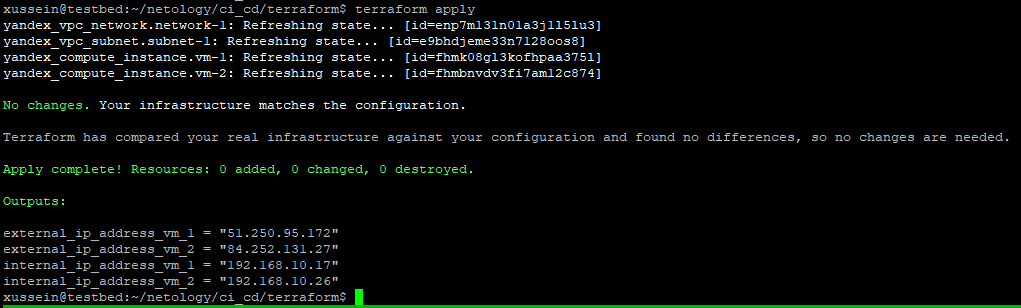
Изучите [инструкцию](https://cloud.yandex.ru/docs/tutorials/infrastructure-management/terraform-quickstart) yandex для terraform. Добейтесь работы паплайна с безопасной передачей токена от облака в terraform через переменные окружения. Для этого:

1. Настройте профиль для yc tools по инструкции.
2. Удалите из кода строчку "token = var.yandex\_cloud\_token". Terraform будет считывать значение ENV переменной YC\_TOKEN.
3. Выполните команду export YC\_TOKEN=$(yc iam create-token) и в том же shell запустите terraform.
4. Для того чтобы вам не нужно было каждый раз выполнять export - добавьте данную команду в самый конец файла ~/.bashrc

# **Решение 3**



Ну и, после запуска еще раз, уже не требует ключа:



Выведем например список хостов на аккаунте:

