Раздел 1. Установка docker-compose

До начала работ убедитесь, что у вас установлен docker (подробное описание установки со ссылкой на дистрибутив находится в методическом материале по работе с ansible) и произведите установку docker-compose

Ссылка на дистрибутив и мануал по установке:
- docker-compose: https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-compose-on-ubuntu-20-04

--
Устанавливаем docker-compose (ubuntu):

загружаем релиз:

```
$ sudo curl -L
"https://github.com/docker/compose/releases/download/1.29.2/docker
-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

назначаем права:
```
$ sudo chmod +x /usr/local/bin/docker-compose
```

проверяем успешность установки:
```
$ docker-compose -version
```

Раздел 2. Пример по работе с terraform

Проинициализируем рабочее пространство terraform:

```
$ terraform init
```

```
root@filipp-notebook:/home/filipp/Desktop/tf-edu# terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp.com/edu/hashicups from the dependency l
ock file
- Using previously-installed hashicorp.com/edu/hashicups v0.3.1

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Появилась директория **terraform.d/** - в ней находится провайдер
plugins - hasicorp.com - edu - hashicups - 0.3.1 - linux_amd64 - сам плагин (terraform-provider-hashicups)

Появилась директория **.terraform/**, в которой хранятся проинициализированные файлы, в т.ч. и файлы провайдера (если захотим использовать новую версию провайдера, папку .terraform/ надо будет чистить и затем снова выполнять terraform init)

Создался файл **.terraform.lock.hcl** - содержит кэш используемых провайдеров и плагинов (этот файл тоже надо будет удалять при обновлении)

```
root@filipp-notebook:/home/filipp/Desktop/tf-edu# ll
total 40
drwxrwxrwx  5 filipp filipp 4096 ноя  3 17:21 ./
drwxr-xr-x 10 filipp filipp 4096 ноя  3 13:27 ../
drwxrwxrwx  2 filipp filipp 4096 окт 25 20:05 docker_compose/
-rwxrwxrwx  1 filipp filipp 1933 ноя  2 16:35 main.tf*
drwxr-xr-x  3 root   root   4096 ноя  3 17:21 .terraform/
drwxrwxrwx  3 filipp filipp 4096 окт 26 10:39 terraform.d/
-rw-r--r--  1 root   root    274 ноя  3 17:21 .terraform.lock.hcl
-rw-rw-r--  1 filipp filipp 7160 ноя  2 16:33 tf.txt
-rwxrwxrwx  1 filipp filipp 2323 окт 26 14:18 инструкция*
root@filipp-notebook:/home/filipp/Desktop/tf-edu#
```

--
Поднимаем инфраструктуру в docker-compose, с которой будет работать через terraform:

```
$ docker-compose up
```

```
root@filipp-notebook:/home/filipp/Desktop/tf-edu# cd docker_compose/
root@filipp-notebook:/home/filipp/Desktop/tf-edu/docker_compose# docker-compose
 up
Starting docker_compose_db_1 ... done
Recreating docker_compose_api_1 ... done
Attaching to docker_compose_db_1, docker_compose_api_1
db_1   |
db_1   | PostgreSQL Database directory appears to contain a database; Skipping
initialization
db_1   |
db_1   | 2022-11-03 14:23:52.221 UTC [1] LOG:  listening on IPv4 address "0.0.0
.0", port 5432
db_1   | 2022-11-03 14:23:52.221 UTC [1] LOG:  listening on IPv6 address "::",
port 5432
db_1   | 2022-11-03 14:23:52.222 UTC [1] LOG:  listening on Unix socket "/var/r
un/postgresql/.s.PGSQL.5432"
db_1   | 2022-11-03 14:23:52.233 UTC [27] LOG:  database system was shut down a
t 2022-11-02 15:04:19 UTC
db_1   | 2022-11-03 14:23:52.241 UTC [1] LOG:  database system is ready to acce
pt connections
api_1  | 2022-11-03T14:23:52.530Z [INFO]  Starting service: bind=0.0.0.0:9090 m
etrics=localhost:9102
```

Проверим готовность инфраструктуры:

```
$ curl localhost:19090/health
```

```
root@filipp-notebook:/home/filipp/Desktop/tf-edu# curl local
host:19090/health
okroot@filipp-notebook:/home/filipp/Desktop/tf-edu#
```

Создадим пользователя (назовем его, к примеру, student):

```
$ curl -X POST localhost:19090/signup -d '{"username":"student",
"password":"test123"}'
```
(не забудьте сверить свой username в этом запросе!)

Вернется примерно следующий ответ:

```
root@filipp-notebook:/home/filipp/Desktop/tf-edu# curl -X POST localhost:19090/
signup -d '{"username":"student", "password":"test123"}'
{"UserID":5,"Username":"student","token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJleHAiOjE2Njc1NzIwODEsInVzZXJfaWQiOjUsInVzZXJuYW1lIjoic3R1ZGVudCJ9.e43gGgjSp8
v2XPRat-6JA0zIjkFxtT9K5Bw-M0MLXjI"}
```

Полученный токен необходимо добавить в переменные окружения терминала,
из которого будет проводится работа с терраформом.
Для Linux это можно сделать так (не забудьте подставить в команду тот токен, который вам вернулся в предыдущей команде):

```
$ export
HASHICUPS_TOKEN=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE1O
TEwNzgwODUsInVzZXJfaWQiOjIsInVzZXJuYW1lIjoiZWR1Y2F0aW9uIn0.CguceCN
ILKdjOQ7Gx0u4UAMlOTaH3Dw-fsll2iXDrYU
```

```
root@filipp-notebook:/home/filipp/Desktop/tf-edu# export HASHICUPS_TOKEN=eyJhbG
ciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2Njc1NzIwODEsInVzZXJfaWQiOjUsInVzZXJu
YW1lIjoic3R1ZGVudCJ9.e43gGgjSp8v2XPRat-6JA0zIjkFxtT9K5Bw-M0MLXjI
root@filipp-notebook:/home/filipp/Desktop/tf-edu# echo $HASHICUPS_TOKEN
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2Njc1NzIwODEsInVzZXJfaWQiOjUsIn
VzZXJuYW1lIjoic3R1ZGVudCJ9.e43gGgjSp8v2XPRat-6JA0zIjkFxtT9K5Bw-M0MLXjI
```

Для Windows (PowerShell):

```
$env:HASHICUPS_TOKEN="eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHA
iOjE1OTEwNzgwODUsInVzZXJfaWQiOjIsInVzZXJuYW1lIjoiZWR1Y2F0aW9uIn0.C
guceCNILKdjOQ7Gx0u4UAMlOTaH3Dw-fsll2iXDrYU"
```

Благодаря тому, что провайдер может использовать в том числе переменные окружения, мы можем держать секреты в относительной безопасности

--

Получим список кофе с помощью terraform.

Комментируем в **main.tf** все data, resource и output кроме data "получение списка кофе":

```
#получение списка кофе
data "hashicups_coffees" "coffees" {
  #без параметров
}

/*
#заказ кофе, состоит из списка кофе
data "hashicups_order" "order_info" {
  #если заказов нет - пустой список
  id = hashicups_order.order.id
}



#блок создания, редактирования заказа
resource "hashicups_order" "order" {
  #блок, описывающий заказ
  items {
    coffee {
      #id кофе
      id = 3
    }
    #кол-во кофе
    quantity = 5
  }

  items {
    coffee {
      id = 2
    }
    quantity = 2
  }

}



output "test_order" {
  value = hashicups_order.order
}



output "test_order_info" {
  value = data.hashicups_order.order_info
}
*/

output "test_coffees" {
  value = data.hashicups_coffees.coffees
}
```

```
$ terraform plan
```

```
root@filipp-notebook:/home/filipp/Desktop/tf-edu# terraform plan
data.hashicups_coffees.coffees: Reading...
data.hashicups_coffees.coffees: Read complete after 0s [id=1667486061]

Changes to Outputs:
  + test_coffees = {
      + coffees = [
          + {
              + description = ""
              + id          = 1
              + image       = "/packer.png"
              + ingredients = [
                  + {
                      + ingredient_id = 1
                    },
                  + {
                      + ingredient_id = 2
                    },
                  + {
                      + ingredient_id = 4
                    },
                ]
```

Проверки того, что будет происходить (при работе с data (datasource) это всегда получение данных, то есть операция чтения)

Применяем изменения (хоть и операция чтения) через terraform:

```
$ terraform apply
```

```
root@filipp-notebook:/home/filipp/Desktop/tf-edu# terraform apply
data.hashicups_coffees.coffees: Reading...
data.hashicups_coffees.coffees: Read complete after 0s [id=166748617

Changes to Outputs:
  + test_coffees = {
      + coffees = [
          + {
              + description = ""
              + id          = 1
              + image       = "/packer.png"
              + ingredients = [
                  + {
                      + ingredient_id = 1
                    },
```

Используемый output создаст файл **terraform.tfstate**:

```
root@filipp-notebook:/home/filipp/Desktop/tf-edu# ll
total 48
drwxrwxrwx  5 filipp filipp 4096 ноя  3 17:36 ./
drwxr-xr-x 10 filipp filipp 4096 ноя  3 13:27 ../
drwxrwxrwx  2 filipp filipp 4096 окт 25 20:05 docker_compose/
-rwxrwxrwx  1 filipp filipp 1936 ноя  3 17:31 main.tf*
drwxr-xr-x  3 root   root   4096 ноя  3 17:21 .terraform/
drwxrwxrwx  3 filipp filipp 4096 окт 26 10:39 terraform.d/
-rw-r--r--  1 root   root    274 ноя  3 17:21 .terraform.lock.hcl
-rw-r--r--  1 root   root   6704 ноя  3 17:36 terraform.tfstate
-rw-rw-r--  1 filipp filipp 7381 ноя  3 17:33 tf.txt
-rwxrwxrwx  1 filipp filipp 2323 окт 26 14:18 инструкция*
```

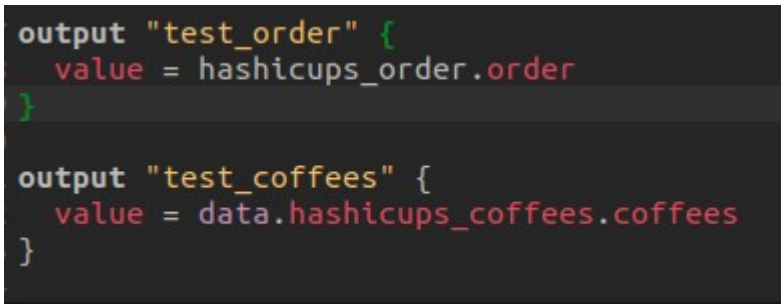
Содержимое **tfstate**-файла определяется провайдером. Создание самого этого файла выполняет terraform
--

Приведем пример с типами resource и data:

`resource "hashicups_order" "order" { ... }`, где
resource (или **data**) - это собственно тип (resource или data)
hashicups_order - наименование resource (или data)
order - переменная, в которой хранится инфомрация, к которой можно обращаться

```
#блок создания, редактирования заказа
resource "hashicups_order" "order" {
  #блок, описывающий заказ
  items {
    coffee {
      #id кофе
      id = 3
    }
    #кол-во кофе
    quantity = 5
  }

  items {
    coffee {
      id = 2
    }
    quantity = 2
  }

}
```

Обращение к типам в **output**:



```
output "test_order" {
  value = hashicups_order.order
}

output "test_coffees" {
  value = data.hashicups_coffees.coffees
}
```

обращение к **resource** (ключевое слово “resource” не указывается):

```
output "test_order" {
  value = hashicups_order.order
}
```

обращение к **data** (ключевое слово “data” указывается):

```
output "test_coffees" {
  value = data.hashicups_coffees.coffees
}
```

--

Раскомментируем блок с редактированием заказов:

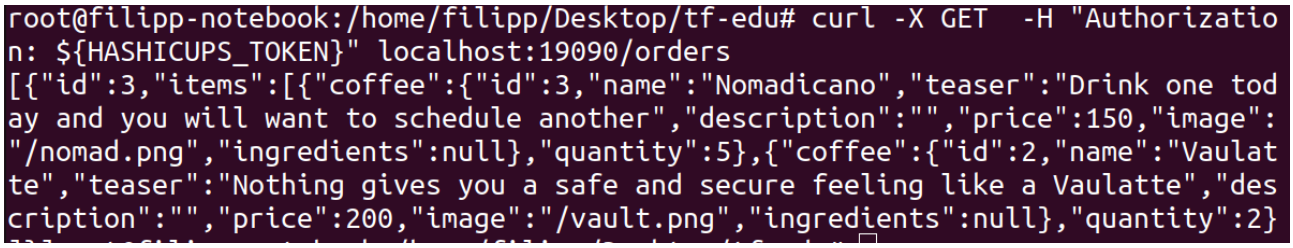
```
#получение списка кофе
data "hashicups_coffees" "coffees" {
  #без параметров
}

/*
#заказ кофе, состоит из списка кофе
data "hashicups_order" "order_info" {
  #если заказов нет - пустой список
  id = hashicups_order.order.id
}
*/


#блок создания, редактирования заказа
resource "hashicups_order" "order" {
  #блок, описывающий заказ
  items {
    coffee {
      #id кофе
      id = 3
    }
    #кол-во кофе
    quantity = 5
  }

  items {
    coffee {
      id = 2
    }
    quantity = 2
  }

}

/*
output "test_order_info" {
  value = data.hashicups_order.order_info
}
*/

output "test_order" {
  value = hashicups_order.order
}

output "test_coffees" {
  value = data.hashicups_coffees.coffees
}
```

Выполняем terraform plan и terraform apply

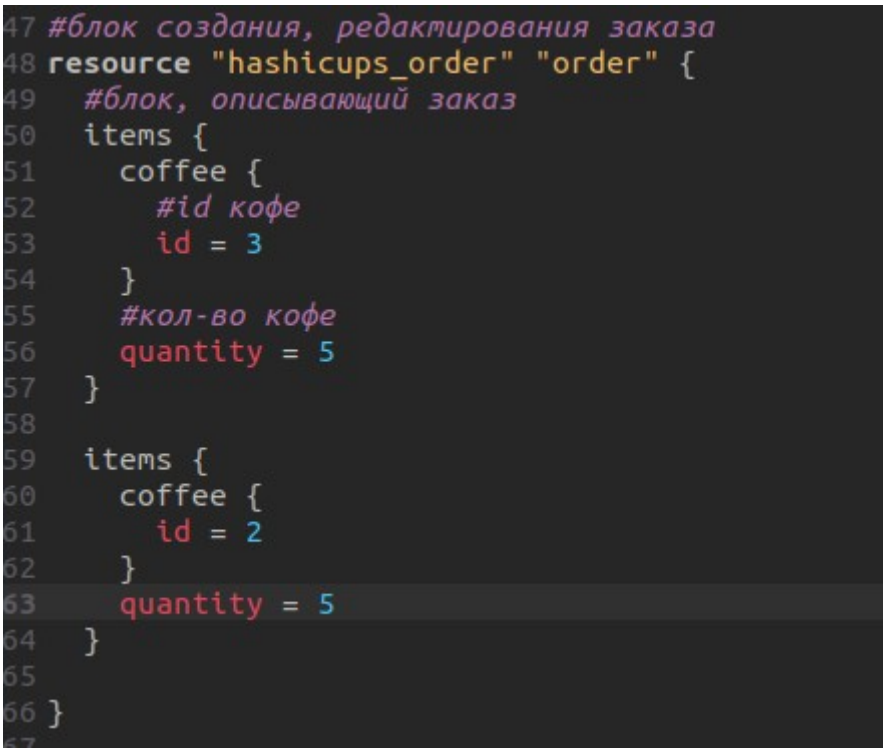
```
$ terraform plan

$ terraform apply
```

Проверяем успешность выполненной операции запросом:

```
$ curl -X GET  -H "Authorization: ${HASHICUPS_TOKEN}"
localhost:19090/orders
```



Изменим количество кофе
`quantity = 2 -> 5` (вносим изменение в **main.tf** файл)




```
$ terraform plan
```

```
root@filipp-notebook:/home/filipp/Desktop/tf-edu# terraform plan
data.hashicups_coffees.coffees: Reading...
hashicups_order.order: Refreshing state... [id=3]
data.hashicups_coffees.coffees: Read complete after 0s [id=166748

Terraform used the selected providers to generate the following ex
plan. Resource actions are indicated with the following symbols:
  ~ update in-place

Terraform will perform the following actions:

  # hashicups_order.order will be updated in-place
  ~ resource "hashicups_order" "order" {
        id = "3"

      ~ items {
          ~ quantity = 2 -> 5
```

```
$ terraform apply
```

Проверяем успешность выполненной операции запросом:

```
$ curl -X GET  -H "Authorization: ${HASHICUPS_TOKEN}"
localhost:19090/orders
```

```
root@filipp-notebook:/home/filipp/Desktop/tf-edu# curl -X GET  -H "Authorizatio
n: ${HASHICUPS_TOKEN}" localhost:19090/orders
[{"id":3,"items":[{"coffee":{"id":3,"name":"Nomadicano","teaser":"Drink one tod
ay and you will want to schedule another","description":"","price":150,"image":
"/nomad.png","ingredients":null},"quantity":5},{"coffee":{"id":2,"name":"Vaulat
te","teaser":"Nothing gives you a safe and secure feeling like a Vaulatte","des
cription":"","price":200,"image":"/vault.png","ingredients":null},"quantity":5}
]}]root@filipp-notebook:/home/filipp/Desktop/tf-edu#
```

--

Выполним read-операцию по получению заказов.
Раскомментируем в **main.tf**:

```
$ terraform plan
```

```
#получение списка кофе
data "hashicups_coffees" "coffees" {
  #без параметров
}


#заказ кофе, состоит из списка кофе
data "hashicups_order" "order_info" {
  #если заказов нет - пустой список
  id = hashicups_order.order.id
}



#блок создания, редактирования заказа
resource "hashicups_order" "order" {
  #блок, описывающий заказ
  items {
    coffee {
      #id кофе
      id = 3
    }
    #кол-во кофе
    quantity = 5
  }

  items {
    coffee {
      id = 2
    }
    quantity = 5
  }

}


output "test_order_info" {
  value = data.hashicups_order.order_info
}


output "test_order" {
  value = hashicups_order.order
}

output "test_coffees" {
  value = data.hashicups_coffees.coffees
}
```

```
$ terraform apply
```

Получаем вывод заказов (также эта информация записалась в **state**-файл)

Можно перепроверить себя, выполнив команду по получению ордера по id:

```
$ curl -X GET  -H "Authorization: ${HASHICUPS_TOKEN}"
localhost:19090/orders/3
```
(подставьте сюда номер вашего заказа, в данном примере это 3 заказ, у вас может быть 1-й - если ранее не делали заказ)

```
{}root@filipp-notebook:/home/filipp/Desktop/tf-edu# curl -X GET  -H "Authorizat
ion: ${HASHICUPS_TOKEN}" localhost:19090/orders/3
{"id":3,"items":[{"coffee":{"id":3,"name":"Nomadicano","teaser":"Drink one toda
y and you will want to schedule another","description":"","price":150,"image":"
/nomad.png","ingredients":null},"quantity":5},{"coffee":{"id":2,"name":"Vaulatt
e","teaser":"Nothing gives you a safe and secure feeling like a Vaulatte","desc
ription":"","price":200,"image":"/vault.png","ingredients":null},"quantity":5}]
}root@filipp-notebook:/home/filipp/Desktop/tf-edu#
```

--

Удалим один **item** из **main.tf:**

```
#блок создания, редактирования заказа
resource "hashicups_order" "order" {
  #блок, описывающий заказ
  items {
    coffee {
      #id кофе
      id = 3
    }
    #кол-во кофе
    quantity = 5
  }

/*
  items {
    coffee {
      id = 2
    }
    quantity = 5
  }
*/

}
```

```
$ terraform plan
```

```
$ terraform apply
```

По существу это не удаление как таковое, а обновление инфраструктуры

Проверить успешность выполнения операции можно запросом:

```
$ curl -X GET  -H "Authorization: ${HASHICUPS_TOKEN}"
localhost:19090/orders/3
```

```
root@filipp-notebook:/home/filipp/Desktop/tf-edu# curl -X GET  -H "Authorizatio
n: ${HASHICUPS_TOKEN}" localhost:19090/orders/3
{"id":3,"items":[{"coffee":{"id":3,"name":"Nomadicano","teaser":"Drink one toda
y and you will want to schedule another","description":"","price":150,"image":"
/nomad.png","ingredients":null},"quantity":5}]}root@filipp-notebook:/home/filip
p/Desktop/tf-edu#
```

```
$ curl -X GET  -H "Authorization: ${HASHICUPS_TOKEN}"
localhost:19090/orders
```

```
root@filipp-notebook:/home/filipp/Desktop/tf-edu# curl -X GET  -H "Authorizatio
n: ${HASHICUPS_TOKEN}" localhost:19090/orders
[{"id":3,"items":[{"coffee":{"id":3,"name":"Nomadicano","teaser":"Drink one tod
ay and you will want to schedule another","description":"","price":150,"image":
"/nomad.png","ingredients":null},"quantity":5}]}]root@filipp-notebook:/home/fil
ipp/Desktop/tf-edu#
root@filipp-notebook:/home/filipp/Desktop/tf-edu#
root@filipp-notebook:/home/filipp/Desktop/tf-edu# curl -X GET  -H "Authorizatio
n: ${HASHICUPS_TOKEN}" localhost:19090/orders/3
{"id":3,"items":[{"coffee":{"id":3,"name":"Nomadicano","teaser":"Drink one toda
y and you will want to schedule another","description":"","price":150,"image":"
/nomad.png","ingredients":null},"quantity":5}]}root@filipp-notebook:/home/filip
p/Desktop/tf-edu#
root@filipp-notebook:/home/filipp/Desktop/tf-edu#
```

Удаленного **item** (кофе) теперь нет в заказе

Раскомментируем обратно участок кода с **item** и снова проведем обновление инфраструктуры (item появится снова в заказе):

```
$ terraform plan

$ terraform apply

$ curl -X GET  -H "Authorization: ${HASHICUPS_TOKEN}"
localhost:19090/orders/3
```

Кофе в заказе снова появилось:

```
root@filipp-notebook:/home/filipp/Desktop/tf-edu# curl -X GET  -H "Authorizatio
n: ${HASHICUPS_TOKEN}" localhost:19090/orders/3
{"id":3,"items":[{"coffee":{"id":3,"name":"Nomadicano","teaser":"Drink one toda
y and you will want to schedule another","description":"","price":150,"image":"
/nomad.png","ingredients":null},"quantity":5},{"coffee":{"id":2,"name":"Vaulatt
e","teaser":"Nothing gives you a safe and secure feeling like a Vaulatte","desc
ription":"","price":200,"image":"/vault.png","ingredients":null},"quantity":5}]
}root@filipp-notebook:/home/filipp/Desktop/tf-edu#
```

--

Проведем операцию удаления инфраструктуры (terraform **destroy**):

```
$ terraform destroy
```

```
root@filipp-notebook:/home/filipp/Desktop/tf-edu# terraform destroy
data.hashicups_coffees.coffees: Reading...
hashicups_order.order: Refreshing state... [id=3]
data.hashicups_coffees.coffees: Read complete after 0s [id=1667487517]
data.hashicups_order.order_info: Reading...
data.hashicups_order.order_info: Read complete after 0s

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

Terraform will perform the following actions:

  # hashicups_order.order will be destroyed
  - resource "hashicups_order" "order" {
      - id           = "3" -> null
      - last_updated = "Thursday, 03-Nov-22 17:56:24 MSK" -> null

      - items {
          - quantity = 5 -> null
```

Обратите внимание, что удаляется инфраструктура, а не конфигурация.
Также удалился весь контент из **state**-файла (осталась только информация по version)

```
root@filipp-notebook:/home/filipp/Desktop/tf-edu# cat terraform.tfstate
{
  "version": 4,
  "terraform_version": "1.3.3",
  "serial": 17,
  "lineage": "1e16ec79-bafe-b0c2-7993-20016cb04b38",
  "outputs": {},
  "resources": [],
  "check_results": []
}
```

С помощью файла **terraform.tfstate.backup** можно восстановить **state** (провайдер должен поддерживать подобную процедуру)

```
root@filipp-notebook:/home/filipp/Desktop/tf-edu# ll
total 56
drwxrwxrwx  5 filipp filipp  4096 ноя  3 18:01 ./
drwxr-xr-x 10 filipp filipp  4096 ноя  3 13:27 ../
drwxrwxrwx  2 filipp filipp  4096 окт 25 20:05 docker_compose/
-rwxrwxrwx  1 filipp filipp  1926 ноя  3 17:56 main.tf*
drwxr-xr-x  3 root   root    4096 ноя  3 17:21 .terraform/
drwxrwxrwx  3 filipp filipp  4096 окт 26 10:39 terraform.d/
-rw-r--r--  1 root   root     274 ноя  3 17:21 .terraform.lock.hcl
-rw-r--r--  1 root   root     179 ноя  3 17:58 terraform.tfstate
-rw-r--r--  1 root   root   11943 ноя  3 17:58 terraform.tfstate.backup
-rw-rw-r--  1 filipp filipp  8007 ноя  3 18:01 tf.txt
-rwxrwxrwx  1 filipp filipp  2323 окт 26 14:18 инструкция*
root@filipp-notebook:/home/filipp/Desktop/tf-edu#
```

```
root@filipp-notebook:/home/filipp/Desktop/tf-edu# cat terraform.tfstate.backup
{
  "version": 4,
  "terraform_version": "1.3.3",
  "serial": 13,
  "lineage": "1e16ec79-bafe-b0c2-7993-20016cb04b38",
  "outputs": {
    "test_coffees": {
      "value": {
        "coffees": [
          {
            "description": "",
            "id": 1,
            "image": "/packer.png",
            "ingredients": [
              {
                "ingredient_id": 1
              },
```