DevOps notes

None

None

None

Table of contents

TELEGRAM GITHUB LINKEDIN

Profile views 365

1. 👋 hey there

🕵 About Me

I am a DevOps engineer from Saint-Petersburg

✅ Completed courses:

- KodeKloud - Certified Kubernetes Application Developer (CKAD)
- Yandex practicum - DevOps
- ITMO - DevOps engineer
- Yandex practicum - Algorithm and data structure
- MIPT - Deep Learning School

🐶 Pet projects:

- React+Xterm - Interactive Browser Terminal playgrounds
- Python+React - Flashcard Learning Service
- Golang - telegrambot dockerization and grpc integration
- Deep learning - deploy Convolutional neural network in Heroku
- 📫 How to reach me: Telegram Pavel

🛠 Languages and Tools:

2. 01 YandexCloud

2.1 Утилита YC

Изначально создаем или изменяем профиль

Профиль — это локальная конфигурация CLI (yc), где хранятся: cloud-id, folder-id token или service-account-key.json endpoint и т.д. То есть профиль — это не сущность в Yandex Cloud, а просто набор настроек на твоём компьютере.

```
yc init
```

Потом можно выбрать нужный

```
❯ yc config profile list
default
hardway ACTIVE
terraform

❯ yc config profile activate terraform
Profile 'terraform' activated
```

Создаем сервисный аккаунт Сервисный аккаунт (Service Account, SA) — это учётка, принадлежащая не человеку, а приложению. У него: свой уникальный идентификатор (serviceAccountId), свои IAM-права (Identity and Access Management) (например storage.editor), и может быть создан статический ключ, авторизационный ключ или IAM-токен для доступа.

```
❯ yc iam service-account create --name terra

done (1s)
id: ajefj388645earvs656i
folder_id: b1gt98o9j856o777k9r5
created_at: "2025-11-03T14:24:19.206036538Z"
name: terra
```

И добавляем ему права

```
❯ yc resource-manager folder add-access-binding \
  --service-account-name=terra \
  --role=admin \
  --name=default

done (2s)
effective_deltas:
  - action: ADD
    access_binding:
      role_id: admin
      subject:
        id: ajefj388645earvs656i
        type: serviceAccount
```

Добавить зеркало для провайдера Yandex vim ~/.terraformrc

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

Добавляем переменные окружения для дальнейшей работы

```
export YC_TOKEN=$(yc iam create-token)
export YC_CLOUD_ID=$(yc config get cloud-id)
export YC_FOLDER_ID=$(yc config get folder-id)
```

2.2 Terraform

2.2.1 Базовые понятия

Основные компоненты Terraform:

```
Конфигурационные файлы, или Configuration Files
Провайдеры, или Providers
Ресурсы, или Resources
Провиженеры, или Provisioners
Состояние, или State
```

2.2.2 Рабочий процесс

1️⃣ Write или Init 2️⃣ Plan 3️⃣ Apply

Инициализация директории Код для развёртывания Managed Service for YDB:

main.tf **provider.tf**

```
# Создание VPC и подсети
resource "yandex_vpc_network" "this" {
  name = "private"
}

resource "yandex_vpc_subnet" "private" {
  name           = "private"
  zone           = "ru-central1-a"
  v4_cidr_blocks = ["192.168.10.0/24"]
  network_id     = yandex_vpc_network.this.id
}

# Создание диска и виртуальной машины
resource "yandex_compute_disk" "boot_disk" {
  name     = "boot-disk"
  zone     = "ru-central1-a"
  image_id = "fd8ba9d5mfvlncknt2kd" # Ubuntu 22.04 LTS
  size     = 15
}

resource "yandex_compute_instance" "this" {
  name                      = "linux-vm"
  allow_stopping_for_update = true
  platform_id               = "standard-v3"
  zone                      = "ru-central1-a"

  resources {
    cores  = "2"
    memory = "4"
  }

  boot_disk {
    disk_id = yandex_compute_disk.boot_disk.id
  }

  network_interface {
    subnet_id = yandex_vpc_subnet.private.id
  }
}

# Создание Yandex Managed Service for YDB
resource "yandex_ydb_database_serverless" "this" {
  name        = "test-ydb-serverless"
  location_id = "ru-central1"
}

# Создание сервисного аккаунта
resource "yandex_iam_service_account" "bucket" {
  name = "bucket-sa"
}

# Назначение роли сервисному аккаунту
resource "yandex_resourcemanager_folder_iam_member" "storage_editor" {
  folder_id = "b1gt98o9j856o777k9r5"
  role      = "storage.editor"
  member    = "serviceAccount:${yandex_iam_service_account.bucket.id}"
}

# Создание статического ключа доступа
resource "yandex_iam_service_account_static_access_key" "this" {
  service_account_id = yandex_iam_service_account.bucket.id
  description        = "static access key for object storage"
}

# Создание бакета
resource "yandex_storage_bucket" "this" {
  bucket     = "test-s3-bucket-fm3oijfiwf3oro23dffoi93"
  access_key = yandex_iam_service_account_static_access_key.this.access_key
  secret_key = yandex_iam_service_account_static_access_key.this.secret_key

  depends_on = [yandex_resourcemanager_folder_iam_member.storage_editor]
}
```

```
terraform {
  required_providers {
    yandex = {
      source = "yandex-cloud/yandex"
    }
  }
  required_version = ">= 1.00"
}

provider "yandex" {
  zone      = "ru-central1-a"
  folder_id = "b1gt98o9j856o777k9r5"
}
```

Выполнение кода

```
terraform init
terraform plan
terraform apply

# Заменить образ
terraform apply -replace yandex_compute_disk.boot_disk
```

Удалить ресурсы

```
terraform destroy
```

2.2.3 Добавляем провайдеры

```
terraform {
  required_providers {
    random = {
      source = "hashicorp/random"
    }
  }
  required_version = ">= 0.13"
}

resource "random_string" "bucket_name" {
  length  = 8
  special = false
  upper   = false
}

# Создание бакета
resource "yandex_storage_bucket" "this" {
  bucket     = "terraform-bucket-${random_string.bucket_name.result}"
  access_key = yandex_iam_service_account_static_access_key.this.access_key
  secret_key = yandex_iam_service_account_static_access_key.this.secret_key

  depends_on = [ yandex_resourcemanager_folder_iam_member.storage_editor ]
}
```

2.2.4 Переменные в Terraform

Простые типы данных ⭐ String — последовательность символов Unicode, представляющая текст (например, hello). ⭐ Number — числовое значение, которое может представлять целые числа (например, 15) и дробные значения (например, 6,283185). ⭐ Bool — логическое значение (true либо false), которое может использоваться в условной логике

Сложные типы данных 🌟 Коллекции — для группировки схожих значений. 🌟 Структурные типы — для группировки потенциально несхожих значений.

Коллекции делятся на три группы: 1️⃣ Коллекция list(...) 2️⃣ Коллекция map(...) 3️⃣ Коллекция set(...)

Структурные типы данных делятся на два вида: 1️⃣ Тип данных object(...) 2️⃣ Тип данных tuple(...)

> ⚠️ **Warning**
>
> ```
> ❗ Вы не можете назначить переменной зарезервированные имена: source, version, providers, count, for_each, lifecycle, depends_on, locals.
> ```

Info

```
Аргумент *sensitive* Ограничивает вывод интерфейса Terraform при использовании переменной в конфигурации.
```

Аргумент *nullable* Контролирует, может ли переменной быть присвоено значение null.

```
variable "image_id" {
  type = string
}

variable "security_group_ids" {
  description = "(Optional) - List of security group IDs."
  type        = list(string)
  default     = []
}

variable "pg_version" {
  description = "PostgreSQL version"
  type        = string
  default     = "15"
  validation {
    condition     = contains(["12", "12-1c", "13", "13-1c", "14", "14-1c", "15", "15-1c", "16"], var.pg_version)
    error_message = "Allowed PostgreSQL versions are 12, 12-1c, 13, 13-1c, 14, 14-1c, 15, 15-1c, 16."
  }
}

variable "access_policy" {
  description = "Access policy from other services to the PostgreSQL cluster."
  type = object({
    data_lens     = optional(bool, null)
    web_sql       = optional(bool, null)
    serverless    = optional(bool, null)
    data_transfer = optional(bool, null)
  })
  default = {}
}

variable "example" {
  type     = string
  nullable = false
}

variable "user_information" {
  type = object({
    name    = string
    address = string
  })
  sensitive = true
}

resource "some_resource" "a" {
  name    = var.user_information.name
  address = var.user_information.address
}
```

Способы задания переменных

Terraform загружает переменные в следующем порядке: 1️⃣ Переменные окружения. 2️⃣ Файл terraform.tfvars, если он есть. 3️⃣ Файл terraform.tfvars.json, если он есть. 4️⃣ Любые файлы *.auto.tfvars или*.auto.tfvars.json, обрабатываемые в алфавитном порядке их имён. 5️⃣ Любые опции -var и -var-file в командной строке в порядке их предоставления.

В командной строке

```
terraform apply -var="image_id=fd8ne6e3etbrr2ve9nlc"
terraform apply -var='image_id_list=["fd8ne6e3etbrr2ve9nlc","fd8fsjddp35jvb4e4jo7"]' -var="cores=4"
terraform apply -var='subnet_map={"ru-central1-a":"subnet-a","ru-central1-b":"subnet-b"}'
```

В файле

```
terraform apply -var-file="testing.tfvars"

image_id = "fd8ne6e3etbrr2ve9nlc"
zone_names = [
  "ru-central1-a",
  "ru-central1-b",
]
```

Переменные окружения Terraform ищет переменные окружения (начинающиеся с TF_VAR_), за которым следует имя объявленной переменной.

```
export TF_VAR_image_id=fd8ne6e3etbrr2ve9nlc
terraform plan
```

Локальные переменные

```
locals {
  boot_disk_name      = "${var.name_prefix}-boot-disk"
  linux_vm_name       = "${var.name_prefix}-linux-vm"

  labels = {
      owner = "DevOps"
      env   = "dev"
  }
}

resource "yandex_compute_disk" "boot_disk" {
  name = local.boot_disk_name

...

labels = local.labels
}

resource "yandex_compute_instance" "this" {
  name = local.linux_vm_name

...

  labels = local.labels
}

resource "yandex_vpc_subnet" "this" {
...

labels = local.labels
}
```

Выходные переменные/значения

```
output "boot_disk_id" {
  description = "The ID of the boot disk created for the instance."
  value       = yandex_compute_disk.boot_disk.id
}

output "instance_id" {
  description = "The ID of the Yandex Compute instance."
  value       = yandex_compute_instance.this.id
}

output "subnet_id" {
  description = "The ID of the VPC subnet used by the Yandex Compute instance."
  value       = yandex_vpc_subnet.private.id
}

output "ydb_id" {
  description = "The ID of the Yandex Managed Service for YDB instance."
  value       = yandex_ydb_database_serverless.this.id
}

output "service_account_id" {
  description = "The ID of the Yandex IAM service account."
  value       = yandex_iam_service_account.bucket.id
}

output "bucket_name" {
  description = "The name of the Yandex Object Storage bucket."
  value       = yandex_storage_bucket.this.bucket
}

output "access_key" {
  description = "access key"
  sensitive   = true
  value       = yandex_iam_service_account_static_access_key.this
}
```

Sensitive значения

```
❯ terraform output access_key
```

2.2.5 Функции

Встроенные функции Terraform

Работа с числами - min, max, pow, floor, ceil Работа со строками - format, join, split, substr Работа с коллекциями - length, lookup, flatten, element, map, merge Кодирование - base64decode, base64encode, jsonencode Операции с файловой системой - file, fileexists, abspath, templatefile Работа с датой и временем - formatdate, timestamp Шифрование и хеш-функции - base64sha512, bcrypt Работы с сетевой CIDR - cidrsubnet, cidrhost Конвертирование типов - tobool, tomap, tolist

```
variable "value" {
  default = "hello"
}

output "transformed_value" {
  value = upper(format("transformed: %s", var.value))
}
```

```
terraform console

cidrnetmask("172.16.0.0/12")
```

2.2.6 Управляющие структуры

Условные выражения

```
condition ? true_value : false_value

var.name == "" ? "default-name" : var.name
```

Операторы

```
a == b
a != b
a < b
a <= b
a > b
a >= b
a || b
a && b
!a
```

Циклы

```
[for s in var.list : upper(s)]

[for k, v in var.map : length(k) + length(v)]
```

```
variable "servers_count" {
  default = {
    "db" = 3,
    "frontend" = 2,
    "backend" = 5,
    "balancer" = 1
  }
}

output "servers" {
  value = [for k, v in var.servers_count:
    "${k} has ${v} servers"
  ]
}

Outputs:

servers = [
  "backend has 5 servers",
  "balancer has 1 servers",
  "db has 3 servers",
  "frontend has 2 servers",
]
```

Тип данных результата Скобки вокруг выражения for определяют, каким будет тип результирующего объекта:

```
квадратные скобки [ ] – tuple;
фигурные скобки { } – object.
```

Фильтрация значений

```
[for s in var.list : upper(s) if s != ""]
```

2.2.7 Создание нескольких ресурсов

Count

Info

```
Если блок ресурса или модуля включает аргумент count, значение которого является целым числом. Terraform создаст указанное количество экземпляров.
```

Warning

```
❗ Значение count должно быть известно до выполнения кода Terraform. Таким образом, нельзя использовать в качестве значения атрибуты, неизвестные до применения изменений.
```

```
resource "yandex_compute_instance" "this" {
  count = 5

  name        = "Server ${count.index}"
  platform_id = "standard-v1"
  zone        = "ru-central1-a"

  resources {
    cores  = 2
    memory = 4
  }

  boot_disk {
    initialize_params {
      image_id = "fd833v6c5tb0udvk4jo6"
    }
  }

  network_interface {
    subnet_id = yandex_vpc_subnet.this.id
  }
}
```

For_each

Если ваши экземпляры практически идентичны, использование count уместно. Если некоторые из аргументов нельзя задать с помощью целых чисел count.index, то безопаснее использовать for_each. Он лишён недостатка count, поскольку

экземпляры в нём идентифицируются строкой. Аргумент for_each принимает типы данных map или set(string) и создаёт экземпляр для каждого элемента этих типов данных.

```
variable "instances" {
  type = map(object({
    zone = string
    resources = object({
      cores  = number
      memory = number
    })
  }))
  default = {
    "server-1" = {
      zone = "ru-central1-a"
      resources = {
        cores  = 2
        memory = 2
      }
    }
    "server-2" = {
      zone = "ru-central1-b"
      resources = {
        cores  = 4
        memory = 4
      }
    }
  }
}

resource "yandex_compute_instance" "this" {
  for_each = var.instances

  name = each.key
  zone = each.value["zone"]
  resources {
    cores  = each.value.resources.cores
    memory = each.value.resources.memory
  }

...

}
```

```
variable "databases" {
  description = "List of PostgreSQL databases."

  type = list(object({
    name                = string
    owner               = string
    lc_collate          = optional(string, null)
    lc_type             = optional(string, null)
    template_db         = optional(string, null)
    deletion_protection = optional(bool, null)
    extensions          = optional(list(string), [])
  }))
}

resource "yandex_mdb_postgresql_database" "database" {
  for_each = length(var.databases) > 0 ? { for db in var.databases : db.name => db } : {}

  name = each.value.name
...
}
```

2.2.8 Динамические блоки

Некоторые типы ресурсов содержат повторяемые вложенные блоки в своих аргументах, обычно они представляют собой отдельные объекты. Вы можете динамически конструировать вложенные блоки с помощью dynamic-блоков. Они поддерживаются внутри блоков resource, data, provider и provisioner.

```
resource "yandex_vpc_security_group" "this" {
...
  ingress {
    protocol       = "TCP"
    description    = "HTTP"
    v4_cidr_blocks = ["10.0.1.0/24"]
    port           = 80
  }

  ingress {
    protocol       = "TCP"
    description    = "HTTPS"
    v4_cidr_blocks = ["0.0.0.0/0"]
    port           = 443
  }
...
}
```

```
variable "ingress_rules" {
  type = list(object({
    protocol       = string
    description    = string
    v4_cidr_blocks = list(string)
    port           = number
  }))
  default = [
    {
      protocol       = "TCP"
      description    = "HTTP"
      v4_cidr_blocks = ["10.0.1.0/24"]
      port           = 80
    },
    {
      protocol       = "TCP"
      description    = "HTTPS"
      v4_cidr_blocks = ["0.0.0.0/0"]
      port           = 443
    }
  ]
}

resource "yandex_vpc_security_group" "this"
...
  dynamic "ingress" {
    for_each = var.ingress_rules
    content {
      protocol       = ingress.value.protocol
      description    = ingress.value.description
      port           = ingress.value.port
      v4_cidr_blocks = ingress.value.v4_cidr_blocks
    }
...
}
```

```
  dynamic "network_interface" {
    for_each = var.network_interfaces
    content {
      subnet_id          = network_interface.value.ubnet_id
      nat                = network_interface.value.nat
      nat_ip_address     = network_interface.value.nat_ip_address
      security_group_ids = network_interface.value.security_group_ids
      ip_address         = network_interface.value.ip_address

      dynamic "dns_record" {
        for_each = network_interface.value.dns_record
        content {
          fqdn        = dns_record.value.fqdn
          dns_zone_id = dns_record.value.dns_zone_id
          ttl         = dns_record.value.ttl
          ptr         = dns_record.value.ptr
        }
      }
    }
  }
```

2.2.9 Указание зависимости

В большинстве случаев Terraform сам управляет зависимостями. Вам нужно явно указать зависимость только в том случае, когда ресурс или модуль зависит от поведения другого ресурса, но не обращается к данным этого ресурса в своих аргументах.

```
variable "databases" {
...
  type = list(object({
    name = string
  }))
...
}

variable "users" {
...
  type = list(object({
    name                  = string
    password              = optional(string, null)
    authentication_plugin = optional(string, null)
    global_permissions    = optional(list(string), [])
    connection_limits = optional(object({
      max_questions_per_hour   = optional(number, -1)
      max_updates_per_hour     = optional(number, -1)
      max_connections_per_hour = optional(number, -1)
      max_user_connections     = optional(number, -1)
    }), null)
    permissions = optional(list(object({
      database_name = string
      roles         = optional(list(string), ["ALL"])
    })), [])
  }))
...
}

resource "yandex_mdb_mysql_database" "database" {
  for_each = length(var.databases) > 0 ? { for db in var.databases : db.name => db } : {}

  cluster_id = yandex_mdb_mysql_cluster.this.id
  name       = lookup(each.value, "name", null)
}

resource "yandex_mdb_mysql_user" "user" {
...

  dynamic "permission" {
    for_each = lookup(each.value, "permissions", [])
    content {
      database_name = permission.value.database_name
      roles         = permission.value.roles
    }
  }
...
  # Create databases before users
  depends_on = [
    yandex_mdb_mysql_database.database
  ]
}
```

2.2.10 State-файл в Terraform

Изменения в файле backend.tf требуют повторной инициализации

Решение: используйте команду terraform init -reconfigure.

В момент применения конфигурации пропала связь с бэкендом

Решение: исправьте командой terraform force-unlock LOCK_ID.

Пример LOCK_ID: yandex_storage_bucket.dev.

Блоке import

Блок import — это новый декларативный способ описывать импорт ресурсов прямо в Terraform-коде, без ручного запуска terraform import ... в консоли.

🔹 Проще говоря: ты описываешь в коде, какие ресурсы нужно импортировать в state, и Terraform сам делает импорт при terraform apply.

```
import {
  to = yandex_compute_instance.testvm
  id = "<< VM ID >>"
}

resource "yandex_compute_instance" "testvm" {
  name                      = "testvm"
  platform_id               = "standard-v3"
  allow_stopping_for_update = true

  resources {
    cores  = 2
    memory = 2
  }

  scheduling_policy {
    preemptible = true
  }

  boot_disk {
    disk_id = yandex_compute_disk.testvm.id
    auto_delete = false
  }

  network_interface {
    subnet_id = yandex_vpc_subnet.dev.id
    nat       = true
  }

  metadata = {
    enable-oslogin = true
  }
}
```

2.2.11 Блокировку состояния инфраструктуры