

# Автоматизация администрирования инфраструктуры: Подъем инфраструктуры в Яндекс.Облаке




Александр  
Зубарев



## **Александр Зубарев**

Председатель цикловой комиссии “Информационной  
безопасности инфокоммуникационных систем”

АКТ (ф) СПбГУТ

 Александр Зубарев



# Вспоминаем прошлые занятия

**Вопрос:** Что такое Terraform?



# Вспоминаем прошлые занятия

**Вопрос:** Что такое Terraform?

**Ответ:** программный инструмент с открытым исходным кодом, который помогает управлять инфраструктурой.



# Вспоминаем прошлые занятия

**Вопрос:** Какие особенности имеет Terraform?

---

# Вспоминаем прошлые занятия

**Вопрос:** Какие особенности имеет Terraform?

**Ответ:**

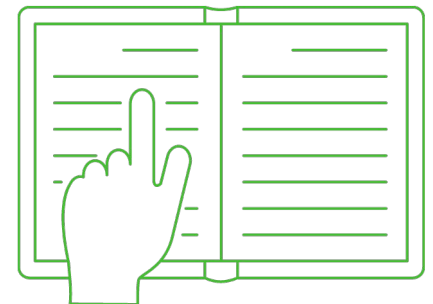
- оркестрирование;
- построение неизменяемой инфраструктуры;
- декларативный код;
- архитектура, работающая на стороне клиента.

---

# Предисловие

## На этом занятии мы:

- поговорим о том, как работает механизм развертывания инфраструктуры в облаке;
- познакомимся с системой связи методов быстрой развертки и масштабирования инфраструктуры в облаке.





# План занятия

1. [Конфигурация Terraform](#)
2. [Yandex.Cloud](#)
3. [Подготовка облака](#)
4. [Создание инфраструктуры](#)
5. [Создание пользователей](#)
6. [Создание и удаление ресурсов](#)
7. [Итоги](#)
8. [Домашнее задание](#)





# Конфигурация Terraform

---

## Простейшая конфигурация состоит из четырех файлов:

- **main.tf** — основной конфиг, описывающий какие инстансы нам нужны;
- **variables.tf** — конфиг с описанием переменных и значениями по умолчанию, если дефолтных значений нет, то они являются обязательными;
- **terraform.tfvars** — конфиг со значением переменных, часто является секретным, нужно с осторожностью пушить в публичные репозитории;
- **outputs.tf** — описание выходных переменных, необязательный файл, но очень удобно вычленять нужные параметры из созданного инстанса, например IP созданного в облаке инстанса.

# Пример конфигурационного файла main.tf

```
provider "yandex" {  
  token      = var.oauth_token  
  cloud_id   = var.cloud_id  
  folder_id  = var.folder_id  
  zone       = "ru-central1-a"  
}
```

# Пример конфигурационного файла main.tf

**Providers** (конкретное облако — GCP, DO, AWS, YANDEX и др.):

- содержат настройки аутентификации и подключения к платформе или сервису,
- предоставляют набор ресурсов для управления,
- установка провайдера производится командой:  
`terraform init`.

# Пример конфигурационного файла main.tf

```
resource "yandex_vpc_subnet" "subnet_a" {
  zone           = "ru-central1-a"
  network_id     = yandex_vpc_network.test_network.id
  v4_cidr_blocks = ["10.1.0.0/24"]
}
resource "yandex_vpc_subnet" "subnet_b" {
  zone           = "ru-central1-b"
  network_id     = yandex_vpc_network.test_network.id
  v4_cidr_blocks = ["10.2.0.0/24"]
}
```

---

# Пример конфигурационного файла main.tf

## Resources:

- определяются типом провайдера;
- позволяют управлять компонентами платформы или сервиса;
- могут иметь обязательные и необязательные аргументы;
- могут ссылаться на другие ресурсы;
- комбинация тип ресурса + имя уникально идентифицирует ресурс в рамках данной конфигурации.

# Input-переменные `variables.tf`

Позволяют параметризировать конфигурационные файлы.

**Четыре типа:**

- `string`
- `map`
- `list`
- `boolean`

Можно передать из файла, из окружения, из командной строки или интерактивно.

---

# Input-переменные variables.tf

```
variable project {  
    description = "Project ID"  
}  
  
variable region {  
    description = "ru-central1-c"  
    default = "europe-west1"  
}
```



## Задание переменных через файл

Указываем переменные в файле:

```
my-vars.tfvars: project = "infra-179015" public_key_path = "~/.ssh/appuser.pub"
```

Указываем путь до файла при запуске команд terraform:

```
$ terraform apply -var-file=my-vars.tfvars
```

Если файл называется terraform.tfvars, то переменные загружаются автоматически.

---

## Output-переменные `outputs.tf`

- позволяют сохранить выходные значения после создания ресурсов;
- облегчают процедуру поиска нужных данных;
- используются в модулях как входные переменные для других модулей.

# Output-переменные outputs.tf

```
outputs.tf

output "app_external-ip {

value="${yandex_compute_instance.app.network_interface.0.access_config.0.assigned_nat_ip}"

}

// просмотр

$ terraform output

app_external_ip = 104.199.54.241
```

## Основные команды

Для приведения системы в целевое состояние используется команда:

`terraform -auto-approve=true apply` — идемпотентна!

Для просмотра изменений, которые будут применены:

`terraform plan`

Для обновления конфигурации: `terraform refresh`

Для просмотра выходных переменных: `terraform output`

Для пересоздания ресурса: `terraform taint  
yandex_compute_instance.app`

---

## Основные команды

Для просмотра выходных переменных: `terraform output`

Для пересоздания ресурса: `terraform taint  
yandex_compute_instance.app`



# Yandex.Cloud



# Yandex.Cloud

— это набор связанных сервисов, доступных через интернет, которые помогают быстро и безопасно взять в аренду вычислительные мощности в тех объемах, в которых это необходимо.

Такой подход к потреблению вычислительных ресурсов называется облачные вычисления.



# Yandex.Cloud

Облачные вычисления заменяют и дополняют традиционные дата-центры, расположенные на территории потребителя.

Yandex.Cloud берет на себя задачи по поддержанию работоспособности и производительности аппаратного и программного обеспечения облачной платформы.





# Yandex.Cloud

Yandex.Cloud предлагает различные категории облачных ресурсов: например, виртуальные машины, диски, базы данных.

Управлять ресурсами каждой категории можно с помощью соответствующего сервиса.

[Список сервисов в составе Yandex.Cloud.](#)

Инфраструктура Yandex.Cloud защищена в соответствии с Федеральным законом Российской Федерации «О персональных данных» № 152-ФЗ.

# Метки ресурсов сервисов

**Метка** — это пара ключ-значение в формате:

<имя метки>=<значение метки>.

Вы можете использовать метки для логического разделения ресурсов.

На метку накладывается следующее ограничение:

- Максимальное количество меток на один ресурс: 64.

На ключ:

- Длина — от 1 до 63 символов.

# API сервисов Yandex.Cloud

Yandex.Cloud дают вам возможность создавать системы автоматизации, плагины и другие приложения для управления облачными ресурсами на базе API.

- Infrastructure:
  - [Yandex Compute Cloud](#)
  - [Yandex Object Storage](#)
  - [Yandex Virtual Private Cloud](#)
  - [Yandex Identity and Access Management](#)
  - [Yandex Resource Manager](#)
  - [Yandex Certificate Manager](#)
  - [Yandex Key Management Service](#)
  - [Yandex Network Load Balancer](#)
  - [Yandex Container Registry](#)
  - [Yandex Managed Service for Kubernetes®](#)
  - [Yandex Monitoring](#)




# Подготовка облака

# Подготовка облака к работе

Перед тем, как разворачивать инфраструктуру, зарегистрируйтесь в Yandex.Cloud и создайте платежный аккаунт:


1. Перейдите в консоль управления, затем войдите в Yandex.Cloud.
2. На странице биллинга убедитесь, что у вас подключен платежный аккаунт, и он находится в статусе ACTIVE или TRIAL\_ACTIVE.
3. Далее вы можете создать или выбрать каталог, в котором будет работать ваша виртуальная машина, на странице облака.

# Подготовка облака к работе

[← Стартовая страница](#)  
**Биллинг**  
Общий раздел  
 **Список аккаунтов**

## Список аккаунтов

Фильтр по имени

Имя	Статус	Баланс	Дата создания	
AlexanderZ	Active	0,00 ₽	12 мая 2021, в 21:41	

---

# Подготовка облака к работе

В стоимость поддержки этой инфраструктуры входит:

- плата за хранение данных;
- плата за диски и постоянно запущенные виртуальные машины;
- плата за использование динамических публичных IP-адресов.



# Создание инфраструктуры




\_\_\_\_\_


## CONCLUSIONS


\_\_\_\_\_


default

Каталог

 Дашборд каталога

 Сервисные аккаунты

 Федерации

 Уведомления об инцидентах

Имя сервисного аккаунта	Роли в каталоге	Дата создания	
user-1	editor	13 мая 2021, в 17:36	...

# Создание инфраструктуры

Вторая: получить [статический ключ](#) доступа. Сохранить идентификатор ключа и секретный ключ, они понадобятся в следующих разделах инструкции.

Каталог

Сервисный аккаунт

Обзор

Обзор

Идентификатор

aje352u82t37pri2t5m9

Имя сервисного аккаунта

user-1

Дата создания

13 мая 2021, в 17:36

Роли в каталоге

editor

Ключи доступа

Создать новый ключ

Создать статический ключ доступа

Для Object Storage, Message Queue и Yandex Database

Создать API-ключ


Для упрощенной аутентификации, вместо IAM-токена

Создать авторизованный ключ

Для запроса IAM-токена

# Создание инфраструктуры

← Каталог

Сервисный аккаунт 

🚩 Обзор

Обзор + Создать новый ключ

Идентификатор

aje352u82t37pri2t5m9

Имя сервисного аккаунта

user-1

Дата создания

13 мая 2021, в 17:36

Роли в каталоге

editor

Ключи доступа

Идентификатор	Описание	Дата создания	
<div></div>	default-test	10 июня 2021, в 19:35	...

# Создание инфраструктуры

Создаем файл main.tf

```
terraform {  
  required_providers {  
    yandex = {  
      source = "yandex-cloud/yandex"  
    }  
  }  
}  
  
provider "yandex" {  
  token      = "AxxxxxxQ"  
  cloud_id   = "P9-PFNhiz9b4STN6tpvA"  
  folder_id  = "b1gquhcfohdpq9q9r64k"  
  zone       = "ru-central1-a"  
}
```

# Создание инфраструктуры

Запускаем terraform init и получаем:

```
Initializing the backend...
```

```
Initializing provider plugins...
```

- Reusing previous version of yandex-cloud/yandex from the dependency lock file
- Using previously-installed yandex-cloud/yandex v0.59.0

```
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.
```

---

## Подготовьте план инфраструктуры

С помощью Terraform в Yandex.Cloud можно создавать облачные ресурсы всех типов: виртуальные машины, диски, образы и т.д.

По плану будут созданы:

- две виртуальные машины: terraform1 и terraform2,
- облачная сеть network-1 с подсетью subnet-1.

---

# Подготовьте план инфраструктуры

У машин будут **разные** количества ядер и объемы памяти:

- terraform1: 2 ядра и 2 Гб оперативной памяти,
- terraform2: 4 ядра и 4 Гб оперативной памяти.

Машины автоматически получают публичные IP-адреса и приватные IP-адреса из диапазона 192.168.10.0/24 в подсети subnet-1, находящейся в зоне доступности ru-central1-a и принадлежащей облачной сети network-1.

# Подготовьте план инфраструктуры

Конфигурации ресурсов задаются сразу после конфигурации провайдера:

```
resource "yandex_compute_instance" "vm-1" {
  name = "terraform1"

  resources {
    cores = 2
    memory = 2
  }
  boot_disk {
    initialize_params {
      image_id = "fd87va5cc00gaq2f5qfb"
    }
  }
}
```





# Создание пользователей

# Создайте пользователей

Чтобы добавить пользователя на создаваемую ВМ, в блоке metadata укажите параметр user-data с пользовательскими метаданными. Для этого создаем файл с мета информацией:

```
#cloud-config
users:
  - name: user
    groups: sudo
    shell: /bin/bash
    sudo: ['ALL=(ALL) NOPASSWD:ALL']
    ssh-authorized-keys:
      - ssh-rsa xxxxxxxxxxxx xxxx@example.com
      - ssh-rsa yyyyyyyyyyyy yyyy@desktop
```

## Создайте пользователей

В файле main.tf вместо ssh-keys задайте параметр user-data и укажите путь к файлу с метаданными:

```
# metadata = {  
#   ssh-key = "ubuntu:{file("~/ssh/id_rsa.pub)}"  
#}  
metadata = {  
  user-data = "${file("./meta.txt")}"  
}
```

# Проверьте и отформатируйте файлы конфигураций

Проверьте конфигурацию командой: `terraform validate`

Если конфигурация является допустимой, появится сообщение:

```
root@Netalogy:/home/user/terraform/test# terraform validate Success! The configuration is valid.
```

Отформатируйте файлы конфигураций в текущем каталоге и подкаталогах: `terraform fmt`

```
root@Netalogy:/home/user/terraform/test# terraform fmt main.tf
```



# Создание и удаление ресурсов

# Создание ресурсов

После проверки конфигурации выполните команду: `terraform plan`

Если будут ошибки, на них укажут видом:

Ошибка: синтаксическая ошибка в файле конфигурации в строке main.tf 6, в переменной "имя\_группы ресурсов": 6: тип = строка

Если нет ошибок, то увидите следующее:

**Plan:** 4 to add, 0 to change, 0 to destroy.

**Changes to Outputs:**

```
+ external_ip_address_vm_1 = (known after apply)
+ external_ip_address_vm_2 = (known after apply)
+ internal_ip_address_vm_1 = (known after apply)
+ internal_ip_address_vm_2 = (known after apply)
```

## Создание ресурсов





Чтобы создать ресурсы выполните команду: `terraform apply`

Вывод терминала должен выдать:

```
Apply complete! Resources: 4 added, 0 changed, 0 destroyed.
```

# Создание ресурсов

В итоге получаем:

<input type="checkbox"/>	Имя	Статус	ОС	Платформа	vCPU	Доля vCPU	RAM	Прерываемая	Размер дисков	Зона доступности	⚙
<input type="checkbox"/>	terraform2	 Running		Intel Broadwell	4	100 %	4 ГБ	нет	2.2 ГБ	ru-central1-a	...
<input type="checkbox"/>	terraform1	 Running		Intel Broadwell	2	100 %	2 ГБ	нет	2.2 ГБ	ru-central1-a	...





## Удаление ресурсов

Чтобы удалить ресурсы, созданные с помощью Terraform:

Выполните команду: `terraform destroy`



# Ansible

# Ansible - создание инфраструктуры

Для выполнения процесса подключения к инфраструктуре нужно знать два важных параметра

- 1) ip адрес — указывается в файле `/etc/ansible/hosts`
- 2) ключ для подключения к инфраструктуре :
  - `ssh-keygen`
  - `ssh-copy-id [xxx.xxx.xxx.xxx]`, x -ip хоста инфраструктуры

Или прописать через консоль на яндекс облаке.



## Проверка выполнения

Для этого требуется создать файл с расширением .yaml (YAML) и описать playbook.

Где указать какой хост, какой пользователь, какие тесты надо выполнить.

Для справки [HandBook](#)



# Итоги

---

# Итоги

## Сегодня мы:

- научились разворачивать облачную инфраструктуру, используя инструмент Terraform на платформе Yandex.Cloud;
- использовать Ansible для подключения к инфраструктуре.



---

## Дополнительные материалы по работе с Yandex.Cloud

- [Начало работы с Terraform | Yandex.Cloud](#)
- [Загрузка состояний Terraform в Object Storage | Yandex.Cloud](#)
- [Начало работы с Packer | Yandex.Cloud](#)
- [Docs overview | yandex-cloud/yandex](#)



# Домашнее задание





## Домашнее задание

Давайте посмотрим ваше [домашнее задание](#).

- Вопросы по домашней работе задавайте **в чате** мессенджера Slack.
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

**Задавайте вопросы и  
пишите отзыв о лекции!**

**Имя Фамилия**

