

DevOps notes

None

None

None

Table of contents

1. 🙋 hey there	4
2. 03 IaC	5
2.1 3.1 Yandex cloud	5
2.2 3.2 Terraform	7
2.3 3.3 Terragrunt	14
2.4 3.4 Ansible	0
3. 04 DevOps and CICD	0
3.1 4.1 Docker	0
3.2 4.2 Технология непрерывной поставки ПО	0
4. 05 Kubernetes	0
4.1 5.1 Основные компоненты k8s	0
4.2 5.2 Основные объекты кластера K8s	0
4.3 5.3 Services and Ingress	0
4.4 Ingress	0
4.5 5.4 Helm	0
4.6 5.7 Безопасность в K8s	0
4.7 5.8 KodeKloud CKAD	0
5. 06 ServiceMesh	0
5.1 6.1 Микросервисная архитектура	0
5.2 6.2 Istio	0
6. 07 Observability	0
6.1 7.1 Мониторинг	0
6.2 7.2 Логгирование	0
6.3 7.3 Трейсинг	0
7. 10 Security	0
7.1 10.1 TLS and mTLS	0
8. 20 React	0
8.1 20.1 React	0
8.2 20.2 useState	0

 TELEGRAM

 GITHUB

 LINKEDIN

Profile views 218

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:

- [Golang - telegrambot dockerization and grpc integration](#)
- [Deep learning - deploy Convolutional neural network in Heroku](#)

• 📞 How to reach me: [Telegram](#) [Pavel](#)

Languages and Tools:



2. 03 IaC

2.1 3.1 Yandex cloud

Установка CLI yandex cloud

Инструкция

```
1  curl -sSL https://storage.yandexcloud.net/yandexcloud-yc/install.sh | bash
2
3  yc init
4
5  yc config list
```

Создание сети

```
1  # Посмотрите описание команд CLI для работы с облачными сетями:
2  yc vpc network --help
3
4  # Создайте облачную сеть в каталоге, указанном в вашем профиле CLI:
5  yc vpc network create \
6  --name my-yc-network \
7  --labels my-label=my-value \
8  --description "my first network via yc"
9
10 # Создайте подсеть в облачной сети my-yc-network
11 yc vpc subnet create \
12 --name my-yc-subnet-a \
13 --zone ru-central1-a \
14 --range 10.1.2.0/24 \
15 --network-name my-yc-network \
16 --description "my first subnet via yc"
17
18 # Получите список всех облачных сетей в каталоге, указанном в вашем профиле CLI
19 yc vpc network list
20
21 yc vpc network list --format yaml
```

Создание VM

```
1  # Создайте VM Linux
2  yc compute instance create \
3  --name my-yc-instance \
4  --network-interface subnet-name=my-yc-subnet-a,nat-ip-version=ipv4 \
5  --zone ru-central1-a \
6  --ssh-key ~/.ssh/id_ed25519.pub
7
8  # Узнайте публичный IP-адрес VM. Для этого посмотрите подробную информацию о вашей VM
9  yc compute instance get my-yc-instance
10
11 # Подключиться
12 ssh yc-user@xxx.xxx.xxx
```

Удалить VM, сеть и подсеть

```
1  yc compute instance delete my-yc-instance
2
3  yc vpc subnet delete my-yc-subnet-a
4
5  yc vpc network delete my-yc-network
```

Получить временный токен

```
1  export YC_TOKEN="$(yc iam create-token)"
```

Использовать в gitlab-ci pipeline

- Необходимо указать переменную YC_SERVICE_ACCOUNT_KEY_FILE

2.2 3.2 Terraform

Знакомство с Terraform

- Инструмент для декларативного описания инфраструктуры
- Описание инфраструктуры хранится в конфигурационных .tf-файлах
- State
- Ваши ресурсы не знают о том что они созданы через терраформ
- Терраформ без state'a не знает ничего о вашей инфраструктуре

```
1 terraform -h
2
3 terraform init
4 terraform plan
5 terraform apply
6 terraform destroy
7 terraform show
```

Note

Провайдер это уже кем то написанная программа которая позволяет вам на языке Терраформа общаться с какой то системой где вы будете делать свою инфраструктуру

```
1 provider "name" {
2     key = value
3 }
4
5 resource "type" "name" {
6     key = value
7 }
8
9 variable "name" {
10     type = ""
11     default = ""
12 }
```

Troubleshooting

- terraform validate
- terraform console
- TF_LOG="DEBUG"
- TF_LOG_FILE

Мета аргументы

У каждого провайдера свои ресурсы и у каждого ресурса свой набор аргументов

Но есть общие мета аргументы

- **provisioner** (запуск чего то сразу после выполнения например ansible (лучше использовать cloud init))
- **depends_on** (неявные зависимости порядок создания ресурсов)
- **count** (счетчик создание нескольких одинаковых ресурсов)
- **for_each** (циклы с разными переменными)
- **provider**
- **lifecycle** (Внутри блока lifecycle имеются следующие аргументы create_before_destroy, prevent_destroy, ignore_changes, и replace_triggered_by.)

Пример создания ВМ на Яндекс Облаке

```
1 export YC_TOKEN="$(yc iam create-token)"
```


main.tf terraform.tfvars variables.tf

```

1 terraform {
2   required_providers {
3     yandex = {
4       source = "yandex-cloud/yandex"
5     }
6   }
7 }
8
9 provider "yandex" {
10   cloud_id = var.cloud_id
11   folder_id = var.folder_id
12   zone = var.zone
13   # token = var.yc_token
14 }
15
16 resource "yandex_compute_instance" "vm-1" {
17   name = "terraform1"
18
19   resources {
20     cores = 2
21     memory = 2
22   }
23
24   boot_disk {
25     initialize_params {
26       image_id = var.image_id
27     }
28   }
29
30   network_interface {
31     subnet_id = yandex_vpc_subnet.subnet-1.0.id
32     nat = true # чтобы машине был выдан внешний IP-адрес
33   }
34
35   metadata = {
36     ssh-keys = "ubuntu:${file("~/ssh/id_rsa.pub")}"
37   }
38
39   lifecycle {
40     prevent_destroy = true
41     ignore_changes = [boot_disk]
42   }
43 }
44
45 resource "yandex_vpc_network" "network-1" {
46   name = "network1"
47 }
48
49 resource "yandex_vpc_subnet" "subnet-1" {
50   count = 3
51   name = "subnet-${count.index}"
52   zone = "ru-central1-a"
53   network_id = yandex_vpc_network.network-1.id
54   v4_cidr_blocks = ["192.168.${count.index+10}.0/24"]
55 }
56
57 output "internal_ip_address_vm_1" {
58   value = yandex_compute_instance.vm-1.network_interface.0.ip_address
59 }
60
61 output "external_ip_address_vm_1" {
62   value = yandex_compute_instance.vm-1.network_interface.0.nat_ip_address
63 }
64

```

```

1 zone = "ru-central1-a"
2 folder_id = "xxxxxx"
3 cloud_id = "xxxxxx"
4 image_id = "fd8tkfhqgbht3sigr37c"

```

```

1 variable "zone"{
2   type = string
3 }
4
5 variable "cloud_id"{
6   type = string
7 }
8
9 variable "folder_id"{
10  type = string
11 }
12
13 variable "image_id"{
14  type = string
15 }
16
17 # variable "yc_token"{
18 #   type = string
19 # }

```




Дополнительные ресурсы

- local
- random
- null

```
1 resource "random_password" "rnd" {  
2   length = 16  
3 }  
4  
5 output "password" {  
6   value = random_password.rnd.result  
7   sensitive = true  
8 }
```

Строковые шаблоны

Например можно сгенерировать инвентори для ансибла

 **hosts.tpl**  **variables.tf**  **main.tf**

```

1  [lbs]
2  %{ for i in range(length(names)) ~}
3  %{ if names[i] ==  "lb" ~}
4  ${names[i]} ansible_host=${addrs[i]} ansible_user=${user}
5  %{ endif ~}
6  %{ endfor ~}
7
8  [apps]
9  %{ for i in range(length(names)) ~}
10 %{ if split("-", names[i])[0] ==  "app" ~}
11 ${names[i]} ansible_host=${addrs[i]} ansible_user=${user}
12 %{ endif ~}
13 %{ endfor ~}

```

```

1  variable "instances" {
2    type = list(string)
3    default = [
4      "app-1",
5      "app-2",
6      "lb"
7    ]
8  }
9
10 variable "user" {
11   type   = string
12   default = "ubuntu"
13 }

```

```

1  resource "yandex_compute_instance" "vm-app" {
2    for_each = toset(var.instances)
3    name     = each.key
4
5    allow_stopping_for_update = true
6    scheduling_policy {
7      preemptible = true
8    }
9
10   resources {
11     core_fraction = 5
12     cores         = 2
13     memory        = 4
14   }
15
16   boot_disk {
17     initialize_params {
18       image_id = var.image_id
19     }
20   }
21
22   network_interface {
23     subnet_id = var.subnet_id
24     nat       = true
25   }
26
27   metadata = {
28     ssh-keys = "ubuntu:${file("~/ssh/id_rsa.pub")}"
29   }
30
31   connection {
32     type        = "ssh"
33     host        = self.network_interface.0.nat_ip_address
34     user        = "ubuntu"
35     agent       = false
36     private_key = file("~/ssh/id_rsa")
37   }
38 }
39
40 locals {
41   names = values(yandex_compute_instance.vm-app)[*].name
42   ips   = values(yandex_compute_instance.vm-app)[*].network_interface.0.nat_ip_address
43 }
44
45 resource "local_file" "ansible_inventory" {
46   content = templatefile("hosts.tpl", {
47     names = local.names,
48     addrs = local.ips,
49     user  = var.user
50   })
51   filename = "inventory"
52 }

```

Управление стейтом

- Хранение state в s3
- Использовать блокировки для предотвращения перезатирания
- Можно указывать remote state как data source

Data-sources

- Получает информацию о других ресурсах в облаке созданных не нами



prod/main.tf

```
1 data "yandex_vpc_network" "default" {
2   name = var.network_name
3 }
```

Модули



modules/app/main.tf



prod/main.tf

```
1 resource "google_compute_instance" "app" {
2   name = "example-instance-${count.index + 1}"
3   machine_type = "f1-micro"
4   count = var.instances_count
5   boot_disk {
6     initialize_params {
7       image = var.app_image
8     }
9   }
10  network_interface {
11    network = "default"
12    access_config {}
13  }
14 }
```

```
1 module "app" {
2   source = "../modules/app/main.tf"
3   instances_count = 2
4   app_image = "myapp-centos-8"
5 }
```

depends_on

- Список ресурсов и модулей от которых зависит текущий ресурс или модуль
- Указывается если нельзя указать аргумент зависящий от другого ресурса

lifecycle

- create_before_destroy
- prevent_destroy (запретить удаление)
- ignore_changes

provisioner

- Выполнение действий выходящих за рамки апи ресурсов

- Также для многих облаков существует специальный аргумент `userdata`, `metadata`

`prod/main.tf`

```
1 resource "aws_instance" "web" {
2   # ...
3
4   provisioner "file" {
5     source      = "script.sh"
6     destination = "/tmp/script.sh"
7   }
8
9   provisioner "remote-exec" {
10    inline = [
11      "chmod +x /tmp/script.sh",
12      "/tmp/script.sh args",
13    ]
14  }
15 }
```

2.3 3.3 Terragrunt

2.3.1 Преимущества

- Он позволяет повторно использовать конфигурационные параметры и поддерживает многоуровневые конфигурации и зависимости
- Расширяет возможности терраформа
- Следует принципу DRY (Don't Repeat Yourself)

Пример

 terraform.hcl  yandex/terragrunt.hcl

```

1  locals {
2    servers = {
3      "master" = {
4        memory      = 4
5        cores       = 2
6        zone        = "ru-central1-a"
7        boot_disk_size = 20
8        boot_disk_type = "network-ssd"
9      }
10     "slave" = {
11       memory      = 4
12       cores       = 2
13       zone        = "ru-central1-a"
14       boot_disk_size = 20
15       boot_disk_type = "network-ssd"
16     }
17   }
18   s3_access_key = get_env("AWS_ACCESS_KEY_ID")
19   s3_secret_key = get_env("AWS_SECRET_ACCESS_KEY")
20   s3_bucket     = "otus-tfstate"
21   s3_region     = "ru-central1"
22
23   yc_token = get_env("YC_TOKEN")
24   yc_cloud_id = "b1ga9aooiodscmmouobm"
25   yc_folder_dev_id = "b1g416evp4d12eef88nt"
26   yc_folder_test_id = "b1gms5goflgecu065agg"
27   yc_zone        = "ru-central1-a"
28 }
29
30
31 generate "provider" {
32   path      = "provider_gen.tf"
33   if_exists = "overwrite"
34   contents  = <<EOF
35   provider "yandex" {
36     token      = "${local.yc_token}"
37     cloud_id    = "${local.yc_cloud_id}"
38     folder_id   = "${local.yc_folder_dev_id}"
39     zone        = "${local.yc_zone}"
40   }
41   provider "yandex" {
42     alias      = "test"
43     token      = "${local.yc_token}"
44     cloud_id    = "${local.yc_cloud_id}"
45     folder_id   = "${local.yc_folder_test_id}"
46     zone        = "${local.yc_zone}"
47   }
48   EOF
49 }
50
51 generate "backend" {
52   path      = "backend_gen.tf"
53   if_exists = "overwrite"
54   contents  = <<EOF
55   terraform {
56     backend "s3" {
57       endpoint      = "storage.yandexcloud.net"
58       bucket        = "${local.s3_bucket}"
59       region        = "${local.s3_region}"
60       key            = "${path_relative_to_include()}/terraform.tfstate"
61       access_key     = "${local.s3_access_key}"
62       secret_key     = "${local.s3_secret_key}"
63       dynamodb_endpoint = "${local.dynamodb_endpoint}"
64       dynamodb_table = "tfstate"
65
66       skip_region_validation = true
67       skip_credentials_validation = true
68     }
69   }
70   EOF
71 }

```

```

1  terraform {
2    source = "../../terraform/modules/yandex"
3  }
4
5  include "root" {
6    path = find_in_parent_folders()
7    expose = true
8  }
9
10 dependency "vpc" {
11   config_path = "../vpc"
12   mock_outputs = {
13     vpc_id = "dummy_vpc"
14   }
15 }
16 mock_outputs_allowed_terraform_commands = ["init", "validate", "plan"]
17
18
19 inputs = {
20   servers      = include.root.locals.servers
21   vpc_id       = dependency.vpc.outputs.vpc_id
22   cidr_blocks  = ["10.51.21.0/24"]
23   network_name = "test_network"
24 }

```