DevOps notes

None

None

None

Table of contents

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

```
curl -sSL https://storage.yandexcloud.net/yandexcloud-yc/install.sh | bash

yc init

yc config list
```

Создание сети

```
# Посмотрите описание команд CLI для работы с облачными сетями:
yc vpc network --help

# Создайте облачную сеть в каталоге, указанном в вашем профиле CLI:
yc vpc network create \
  --name my-yc-network \
  --labels my-label=my-value \
  --description "my first network via yc"

# Создайте подсеть в облачной сети my-yc-network
yc vpc subnet create \
  --name my-yc-subnet-a \
  --zone ru-central1-a \
  --range 10.1.2.0/24 \
  --network-name my-yc-network \
  --description "my first subnet via yc"

# Получите список всех облачных сетей в каталоге, указанном в вашем профиле CLI
yc vpc network list

yc vpc network list --format yaml
```

Создание ВМ

```
# Создайте ВМ Linux
yc compute instance create \
  --name my-yc-instance \
  --network-interface subnet-name=my-yc-subnet-a,nat-ip-version=ipv4 \
  --zone ru-central1-a \
  --ssh-key ~/.ssh/id_ed25519.pub

# Узнайте публичный IP-адрес ВМ. Для этого посмотрите подробную информацию о вашей ВМ
yc compute instance get my-yc-instance

# Подключиться
ssh yc-user@xxx.xxx.xxx.xxx
```

Удалить ВМ, сеть и подсеть

```
yc compute instance delete my-yc-instance

yc vpc subnet delete my-yc-subnet-a

yc vpc network delete my-yc-network
```

Получить временный токен

```
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'а не знает ничего о вашей инфраструктуре

```
terraform -h

terraform init
terraform plan
terraform apply
terraform destroy
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

```
provider "name" {
    key = value
}

resource "type" "name" {
    key = value
}

variable "name" {
    type = ""
    default = ""
}
```

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

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

main.tf terraform.tfvars variables.tf

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

provider "yandex" {
  cloud_id = var.cloud_id
  folder_id = var.folder_id
  zone = var.zone
  # token = var.yc_token
}

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

  resources {
    cores = 2
    memory = 2
  }

  boot_disk {
    initialize_params {
      image_id = var.image_id
    }
  }

  network_interface {
    subnet_id = yandex_vpc_subnet.subnet-1.0.id
    nat = true # чтобы машине был выдан внешний IP-адрес
  }

  metadata = {
    ssh-keys = "ubuntu:${file("~/.ssh/id_rsa.pub")}"
  }

  lifecycle {
    prevent_destroy = true
    ignore_changes = [boot_disk]
  }

}

resource "yandex_vpc_network" "network-1" {
  name = "network1"
}

resource "yandex_vpc_subnet" "subnet-1" {
  count = 3
  name = "subnet-${count.index}"
  zone = "ru-central1-a"
  network_id = yandex_vpc_network.network-1.id
  v4_cidr_blocks = ["192.168.${count.index+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
}
```

```
zone = "ru-central1-a"
folder_id = "xxxxxx"
cloud_id = "xxxxxx"
image_id = "fd8tkfhqgbht3sigr37c"
```

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

variable "cloud_id"{
  type = string
}

variable "folder_id"{
  type = string
}

variable "image_id"{
  type = string
}

# variable "yc_token"{
  # type = string
# }
```

Дополнительные ресурсы

- local
- random
- null

```
resource "random_password" "rnd" {
  length = 16
}

output "password" {
  value = random_password.rnd.result
  sensitive = true
}
```

Строковые шаблоны

Например можно сгенерировать инвентори для ансибла

`hosts.tpl` `variables.tf` `main.tf`

```
[lbs]
%{ for i in range(length(names)) ~}
%{ if names[i] ==  "lb" ~}
${names[i]} ansible_host=${addrs[i]} ansible_user=${user}
%{ endif ~}
%{ endfor ~}

[apps]
%{ for i in range(length(names)) ~}
%{ if split("-", names[i])[0] ==  "app" ~}
${names[i]} ansible_host=${addrs[i]} ansible_user=${user}
%{ endif ~}
%{ endfor ~}
```

```
variable "instances" {
  type = list(string)
  default = [
    "app-1",
    "app-2",
    "lb"
  ]
}

variable "user" {
  type    = string
  default = "ubuntu"
}
```

```
resource "yandex_compute_instance" "vm-app" {
  for_each = toset(var.instances)
  name     = each.key

  allow_stopping_for_update = true
  scheduling_policy {
    preemptible = true
  }

  resources {
    core_fraction = 5
    cores         = 2
    memory        = 4
  }

  boot_disk {
    initialize_params {
      image_id = var.image_id
    }
  }

  network_interface {
    subnet_id = var.subnet_id
    nat       = true
  }

  metadata = {
    ssh-keys = "ubuntu:${file("~/.ssh/id_rsa.pub")}"
  }

  connection {
    type        = "ssh"
    host        = self.network_interface.0.nat_ip_address
    user        = "ubuntu"
    agent       = false
    private_key = file("~/.ssh/id_rsa")
  }
}

locals {
  names = values(yandex_compute_instance.vm-app)[*].name
  ips   = values(yandex_compute_instance.vm-app)[*].network_interface.0.nat_ip_address
}

resource "local_file" "ansible_inventory" {
  content = templatefile("hosts.tpl", {
      names = local.names,
      addrs = local.ips,
      user = var.user
  })
  filename = "inventory"
}
```

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

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

Модули

modules/app/main.tf prod/main.tf

```
resource "google_compute_instance" "app" {
    name = "example-instance-${count.index + 1}"
    machine_type = "f1-micro"
    count = var.instances_count
    boot_disk {
        initialize_params {
            image = var.app_image
        }
    }
    network_interface {
        network = "default"
        access_config {}
    }
}
```

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

depends_on

- Список ресурсов и модулей от которых зависит текущий ресурс или модуль
- Указывается если нельзя указать агрумент 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

```
resource "aws_instance" "web" {
# ...

  provisioner "file" {
    source      = "script.sh"
    destination = "/tmp/script.sh"
  }

  provisioner "remote-exec" {
    inline = [
      "chmod +x /tmp/script.sh",
      "/tmp/script.sh args",
    ]
  }
}
```

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**

```
locals {
  servers = {
    "master" = {
      memory         = 4
      cores          = 2
      zone           = "ru-central1-a"
      boot_disk_size = 20
      boot_disk_type = "network-ssd"
    }
    "slave" = {
      memory         = 4
      cores          = 2
      zone           = "ru-central1-a"
      boot_disk_size = 20
      boot_disk_type = "network-ssd"
    }
  }
  s3_access_key = get_env("AWS_ACCESS_KEY_ID")
  s3_secret_key = get_env("AWS_SECRET_ACCESS_KEY")
  s3_bucket     = "otus-tfstate"
  s3_region     = "ru-central1"

  yc_token = get_env("YC_TOKEN")
  yc_cloud_id  = "b1ga9aooiodscmmouobm"
  yc_folder_dev_id = "b1g416evp4dl2eef88nt"
  yc_folder_test_id = "b1gms5goflgecu065agg"
  yc_zone      = "ru-central1-a"

}

generate "provider" {
  path      = "provider_gen.tf"
  if_exists = "overwrite"
  contents  = <<EOF
provider "yandex" {
  token     = "${local.yc_token}"
  cloud_id  = "${local.yc_cloud_id}"
  folder_id = "${local.yc_folder_dev_id}"
  zone      = "${local.yc_zone}"
}
provider "yandex" {
  alias     = "test"
  token     = "${local.yc_token}"
  cloud_id  = "${local.yc_cloud_id}"
  folder_id = "${local.yc_folder_test_id}"
  zone      = "${local.yc_zone}"
}
EOF
}

generate "backend" {
  path      = "backend_gen.tf"
  if_exists = "overwrite"
  contents  = <<EOF
terraform {
  backend "s3" {
    endpoint                    = "storage.yandexcloud.net"
    bucket                      = "${local.s3_bucket}"
    region                      = "${local.s3_region}"
    key                         = "${path_relative_to_include()}/terraform.tfstate"
    access_key                  = "${local.s3_access_key}"
    secret_key                  = "${local.s3_secret_key}"
    dynamodb_endpoint           = "${local.dynamodb_endpoint}"
    dynamodb_table              = "tfstate"

    skip_region_validation      = true
    skip_credentials_validation = true
  }
}
EOF
}
```

```
terraform {
  source = "../../../terraform/modules/yandex"
}

include "root" {
  path = find_in_parent_folders()
  expose = true
}

dependency "vpc" {
  config_path = "../vpc"
  mock_outputs = {
    vpc_id = "dummy_vpc"
  }

  mock_outputs_allowed_terraform_commands = ["init", "validate", "plan"]
}

inputs = {
  servers      = include.root.locals.servers
  vpc_id       = dependency.vpc.outputs.vpc_id
  cidr_blocks  = ["10.51.21.0/24"]
  network_name = "test_network"
}
```