Теория.

О терраформе

На предыдущих уроках вы создали образ ВМ, описав его текстовым файлом — спецификацией. Сейчас мы пойдём ещё дальше: познакомимся с программой Terraform, которая позволяет похожим образом создавать облачную инфраструктуру (не только ВМ, но и балансировщики, сети, базы данных, хранилища и т. д.). Подготовив один файл спецификации, вы автоматически развернёте из него готовую инфраструктуру. Риски ошибок ручной сборки сводятся к минимуму.
Вот так выглядит каркас спецификации для Terraform. Он состоит из описания **ресурсов**: ВМ, сетей, подсетей и т. д.

```
resource "yandex_compute_instance" "vm-1" {
  ...
}

resource "yandex_vpc_network" "network-1" {
  ...
}

resource "yandex_vpc_subnet" "subnet-1" {
  ...
}
```

Terraform позволяет предварительно посмотреть план выполнения: что будет создано и удалено в процессе работы. Благодаря этому вы можете удостовериться, что получите инфраструктуру нужной конфигурации, а ничего лишнего не появится и не пропадёт.
Вывод команды с проверкой создаваемых ресурсов:

```
Terraform will perform the following actions:

  # yandex_compute_instance.vm-1 will be created
  + resource "yandex_compute_instance" "vm-1" {
    ...
    }

  # yandex_vpc_network.network-1 will be created
  + resource "yandex_vpc_network" "network-1" {
```

```
    ...
    }

  # yandex_vpc_subnet.subnet-1 will be created
  + resource "yandex_vpc_subnet" "subnet-1" {
    ...
    }

Plan: ... to add, 0 to change, 0 to destroy.
```

В Terraform объекты можно связывать друг с другом. Например, можно подключить ВМ к сети, созданной в этой же спецификации.

```
resource "yandex_compute_instance" "vm-1" {
  ...
  network_interface {
    subnet_id = yandex_vpc_subnet.subnet-1.id
    nat       = true
  }
}

resource "yandex_vpc_subnet" "subnet-1" {
  ...
}
```

Спецификации Terraform

Terraform, как и Packer, разработала компания HashiCorp. Облачные провайдеры, в том числе Yandex Cloud, поддерживают спецификации Terraform. Обычно они пишутся на языке HCL и хранятся в файлах формата `.tf`. Для удобства таких файлов может быть несколько. При запуске Terraform просматривает все файлы в директории и воспринимает их как единую спецификацию.
Посмотрите пример файла спецификации. Привязка к провайдеру (в данном случае это Yandex Cloud) задаётся в секциях `required_providers` и `provider`:

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

provider "yandex" {
```

```
  token     = "<OAuth-токен>"
  cloud_id  = "<идентификатор_облака>"
  folder_id = "<идентификатор_каталога>"
  zone      = "<зона_доступности_по_умолчанию>"
}
```

Как и Packer, Terraform поддерживает различные способы аутентификации. В спецификации выше в параметре `token` задан OAuth-токен от Yandex Cloud. Другой способ аутентифицироваться — использовать переменную окружения `YC_TOKEN`, в которую можно записать не только OAuth-токен, но и IAM-токен.
Значения параметров или задаются в спецификации, или передаются в качестве **переменных**, чтобы адаптировать спецификацию для конкретных задач. Например, с помощью одной спецификации вы сможете развернуть одинаковую инфраструктуру в разных каталогах — для тестирования и для рабочей эксплуатации:

```
variable "folder-id" {
  type = string
}

provider "yandex" {
  token     = "<OAuth-токен>"
  cloud_id  = "<идентификатор_облака>"
  folder_id = var.folder-id
  zone      = "<зона_доступности_по_умолчанию>"
}
```

При этом ключевые ресурсы и зависимости остаются зафиксированы в спецификации и обеспечивают ее работоспособность.

Как использовать спецификации Terraform

Инфраструктура разворачивается в три этапа:

1. Команда `terraform init` инициализирует провайдеров, указанных в файле спецификации.
2. Команда `terraform plan` запускает проверку спецификации. Если есть ошибки — появятся предупреждения. Если ошибок нет, отобразится список элементов, которые будут созданы или удалены.
3. Команда `terraform apply` запускает развёртывание инфраструктуры.

Если инфраструктура больше не нужна, её можно удалить командой `terraform destroy`.

Оптимизация создания инфраструктуры

На самом деле Terraform не всегда создаёт заново все ресурсы, описанные в спецификации. Terraform ведёт реестр, в котором фиксирует состояние инфраструктуры в облаке. Этот реестр называется State (**стейт-файл**), он имеет формат JSON.
State поддерживает связь между описанием ресурсов в спецификации и реальными ресурсами в облаке. При запуске команд `plan` и `apply` стейт-файл сравнивается с ресурсами, которые нужно создать из спецификации. По итогам сравнения недостающие ресурсы создаются, лишние — удаляются, а некоторые изменяются на ходу. Такой подход позволяет существенно улучшить производительность операций развёртывания, особенно для масштабных инфраструктур.
После выполнения команды `apply` стейт-файл обновляется.
С помощью команд и стейт-файлов вы можете управлять конфигурацией облачной инфраструктуры: импортировать ее описание в стейт-файл (команда `terraform import`), исключить ресурсы из стейт-файла (`terraform state rm`), выгрузить описание (`terraform output` и `terraform show`).

Практическая работа. Создаём виртуальную машину из образа и базу данных

В этой практической работе вы установите Terraform и подготовите спецификацию, с помощью которой создадите виртуальную машину, а затем управляемую базу данных. Подсказки для создания спецификации смотрите в документации Yandex Cloud и в справочнике ресурсов (раздел Resources).

1. Дистрибутив для вашей платформы можно скачать из зеркала. После загрузки добавьте путь к папке, в которой находится исполняемый файл, в переменную `PATH`. Настройте провайдер.
2. Создайте файл спецификации `my-config.tf` и укажите в нём Yandex Cloud в качестве провайдера.

   Скопировать код

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

 provider "yandex" {
   token  =  "<OAuth-токен>"
   cloud_id  = "<идентификатор_облака>"
   folder_id = "<идентификатор_каталога>"
   zone      = "<зона_доступности_по_умолчанию>"
 }
```

3. *Далее мы будем считать, что в качестве зоны доступности по умолчанию выбрана* `ru-central1-a`.
4. Для создания ВМ используйте образ, созданный с помощью Packer в предыдущей практической работе.
   Можно использовать переменные в спецификации Terraform и передавать в них разные значения при запуске команд. Например, если сделать переменную для идентификатора образа `image-id`, тогда с помощью одного и того же файла спецификации вы сможете создавать ВМ с разным наполнением.
   Переменные Terraform хранятся в файлах с расширением `.tfvars`. Создайте файл `my-variables.tfvars` и укажите в нём идентификатор своего образа Packer (узнайте идентификатор с помощью команды `yc compute image list`):

   Скопировать код

```
image-id = "<идентификатор_образа>"
```

5. В файле спецификации `my-config.tf` объявите эту переменную (ключевое слово `variable`). Тогда в секции, где описываются настройки ВМ, вы сможете обратиться к переменной как `var.image-id`:

   Скопировать код

```
...
variable "image-id" {
  type = string
}

resource "yandex_compute_instance" "vm-1" {
  name = "from-terraform-vm"
  platform_id = "standard-v1"
  zone = "ru-central1-a"

  resources {
    cores  = 2
    memory = 2
  }

  boot_disk {
    initialize_params {
      image_id = var.image-id
    }
  }
  network_interface {
    subnet_id = yandex_vpc_subnet.subnet-1.id
    nat       = true
  }

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

 ...
```

6. Скорректируйте описание для сети и подсети.
   Для сети достаточно указать имя:

   Скопировать код

```
 resource "yandex_vpc_network" "network-1" {
```

```
    name = "from-terraform-network"
  }
```

7. Для подсети укажите зону доступности и сеть, а также внутренние IP-адреса, уникальные в рамках сети. Используйте адреса из адресного пространства `10.0.0.0/16`.

   Скопировать код

```
 resource "yandex_vpc_subnet" "subnet-1" {
      name           = "from-terraform-subnet"
      zone           = "ru-central1-a"
      network_id     = "${yandex_vpc_network.network-1.id}"
      v4_cidr_blocks = ["10.2.0.0/16"]
  }
```

Проверьте синтаксис спецификации:

Скопировать код

```
variable "image-id" {
  type = string
}

resource "yandex_compute_instance" "vm-1" {
  name = "from-terraform-vm"
  platform_id = "standard-v1"
  zone = "ru-central1-a"

  resources {
    cores  = 2
    memory = 2
  }

  boot_disk {
    initialize_params {
      image_id = var.image-id
    }
  }

  network_interface {
    subnet_id = yandex_vpc_subnet.subnet-1.id
    nat       = true
  }

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

```
resource "yandex_vpc_network" "network-1" {
  name = "from-terraform-network"
}

resource "yandex_vpc_subnet" "subnet-1" {
  name           = "from-terraform-subnet"
  zone           = "ru-central1-a"
  network_id     = "${yandex_vpc_network.network-1.id}"
  v4_cidr_blocks = ["10.2.0.0/16"]
}

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
}
```

4. Теперь попробуйте применить спецификацию. Перейдите в папку с файлом спецификации и выполните инициализацию.

   Скопировать код

```
terraform init
```

5. Если всё сделано верно, Terraform покажет сообщение:

```
...
 Terraform has been successfully initialized!
 ...
```

Важно: выполняйте команды Terraform в папке, где находится файл спецификации!!!

5. Проверьте спецификацию с помощью команды `terraform plan`.
   Terraform использует все файлы `.tf` из папки, в которой запущена команда. Поэтому название файла спецификации `my-config.tf` указывать не нужно: его Terraform подхватит и так.
   Если файл с переменными называется стандартно (`terraform.tfvars`), его тоже можно не указывать при запуске команды. А если название файла нестандартное, то его нужно указывать:

   Скопировать код

```
terraform plan -var-file=my-variables.tfvars
```

6. Terraform выведет план: объекты, которые будут созданы, и т. п.:

```
...
 Terraform will perform the following actions:
 ...
```

7. На самом деле необязательно помещать переменные в файл, их можно просто указывать при запуске команды. Поскольку у вас только одна переменная, это было бы несложно:

   Скопировать код

```
terraform plan -var="image-id=<идентификатор_образа>"
```

6. Создайте в облаке инфраструктуру по описанной вами спецификации. Выполните команду:

   Скопировать код

```
terraform apply -var-file=my-variables.tfvars
```

7. Terraform запросит подтверждение:

```
...
 Do you want to perform these actions?
      Terraform will perform the actions described above.
      Only 'yes' will be accepted to approve.

      Enter a value:
```

8. В ответ введите `yes`.

   Когда команда будет выполнена, вы увидите сообщение:

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

  Outputs:

  external_ip_address_vm_1 = "84.201.133.49"
  internal_ip_address_vm_1 = "10.2.0.24"
```

9. В консоли управления убедитесь, что ВМ создана. Откройте в браузере страницу с указанным IP-адресом и проверьте, доступна ли ВМ.

10. Terraform хранит описание инфраструктуры в стейт-файлах. Посмотрите, как выглядит стейт-файл сейчас:

Скопировать код

```
terraform state list
```

11. Вы увидите список объектов:

Скопировать код

```
yandex_compute_instance.vm-1
yandex_vpc_network.network-1
yandex_vpc_subnet.subnet-1
```

12. Теперь добавьте в файл спецификации блок, описывающий создание кластера БД PostgreSQL.

Скопировать код

```
resource "yandex_mdb_postgresql_cluster" "postgres-1" {
  name        = "postgres-1"
  environment = "PRESTABLE"
  network_id  = yandex_vpc_network.network-1.id

  config {
    version = 12
    resources {
      resource_preset_id = "s2.micro"
      disk_type_id       = "network-ssd"
      disk_size          = 16
    }
    postgresql_config = {
      max_connections                   = 395
      enable_parallel_hash              = true
      vacuum_cleanup_index_scale_factor = 0.2
      autovacuum_vacuum_scale_factor    = 0.34
      default_transaction_isolation     =
"TRANSACTION_ISOLATION_READ_COMMITTED"
      shared_preload_libraries          =
"SHARED_PRELOAD_LIBRARIES_AUTO_EXPLAIN,SHARED_PRELOAD_LIBRARIES_PG_HINT
_PLAN"
    }
  }

  database {
```

```
    name  = "postgres-1"
    owner = "my-name"
  }

  user {
    name       = "my-name"
    password   = "Test1234"
    conn_limit = 50
    permission {
      database_name = "postgres-1"
    }
    settings = {
      default_transaction_isolation = "read committed"
      log_min_duration_statement    = 5000
    }
  }

  host {
    zone      = "ru-central1-a"
    subnet_id = yandex_vpc_subnet.subnet-1.id
  }
}
```

Сохраните файл спецификации.

13. Теперь примените обновлённую спецификацию. В папке с файлом спецификации выполните команду `terraform plan`:

Скопировать код

```
terraform plan -var-file=my-variables.tfvars
```

Если появляются сообщения об ошибках — исправьте ошибки и снова выполните команду.

Обновите инфраструктуру в соответствии с дополненной спецификацией командой `terraform apply`:

Скопировать код

```
terraform apply -var-file=my-variables.tfvars
```

Поскольку спецификация теперь включает создание БД, команда может выполняться довольно долго (около 10 минут).

В консоли управления откройте раздел **Managed Service for PostgreSQL** и убедитесь, что кластер `postgres-1` создан и имеет статус `Alive`.

Проверьте, как изменился стейт-файл:

Скопировать код

```
terraform state list
```

В списке появился новый объект:

```
yandex_compute_instance.vm-1
yandex_mdb_postgresql_cluster.postgres-1
yandex_vpc_network.network-1
yandex_vpc_subnet.subnet-1
```

Удалите инфраструктуру:

Скопировать код

```
terraform destroy -var-file=my-variables.tfvars
```

В конце вы увидите сообщение о выполнении команды:

```
...

Destroy complete! Resources: 4 destroyed.
```

В консоли управления убедитесь, что объекты удалены.