Основы работы с Terraform

Yandex Cloud

Елисей Ильин
DevOps-инженер в Itransition

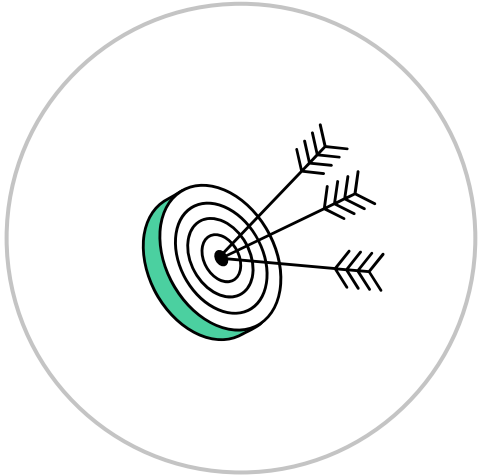
Елисей Ильин

О спикере:

- DevOps-инженер
- Опыт работы в IT 6 лет

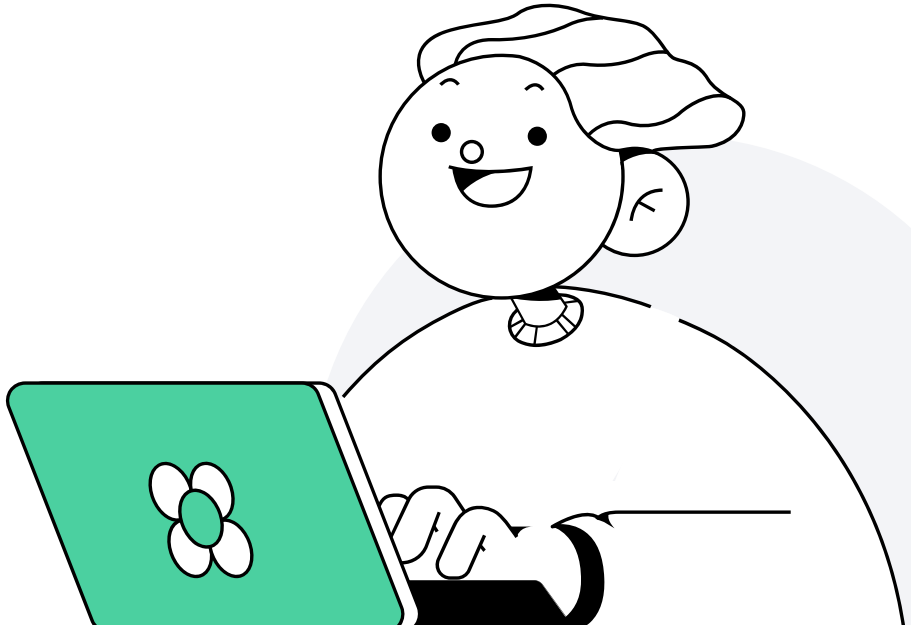
Цели занятия

- Узнать про облачные решения
- Познакомиться с устройством Yandex Cloud
- Создать ресурсы в Yandex Cloud с помощью Terraform
- Рассмотреть типы переменных в Terraform



План занятия

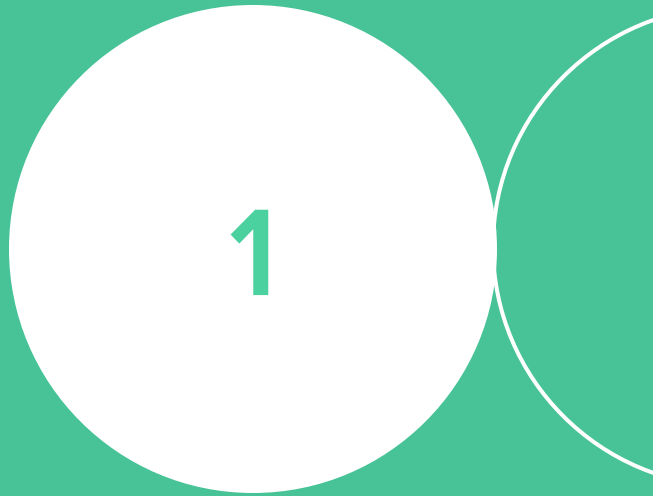
1. Облачные вычисления
2. Основы работы с Yandex Cloud
3. Переменные в Terraform
4. Итоги занятия
5. Домашнее задание



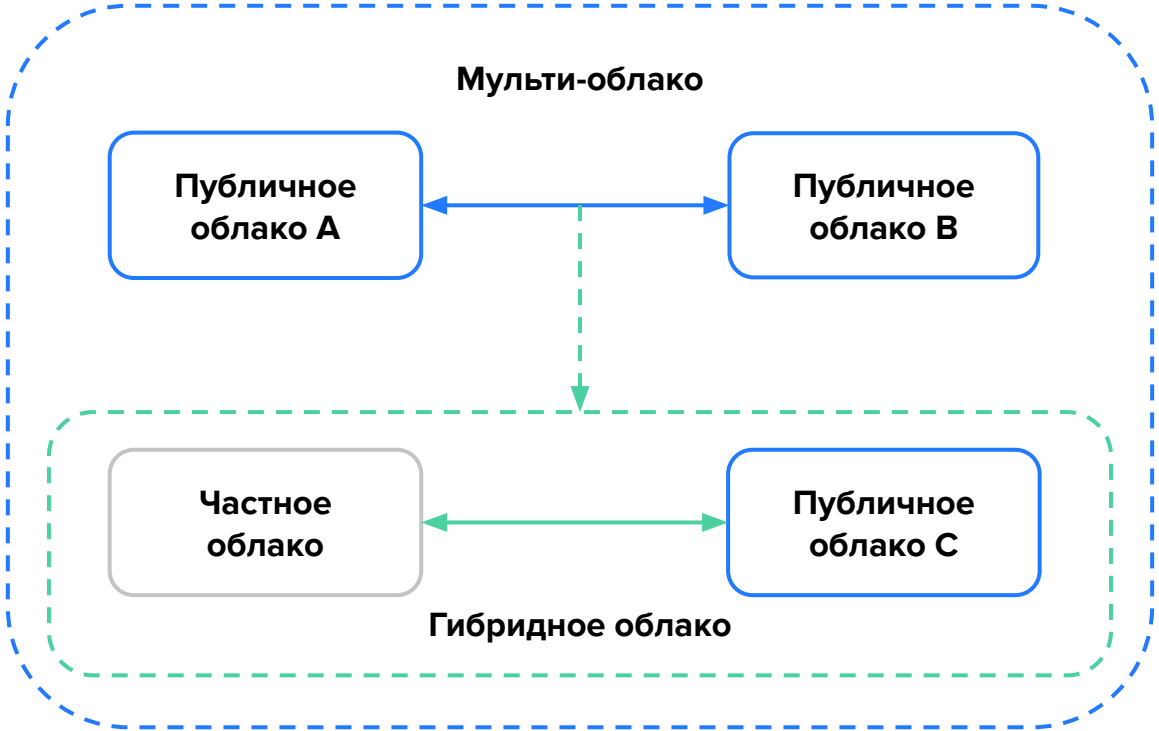
*Нажми на нужный раздел для перехода

Облачные вычисления

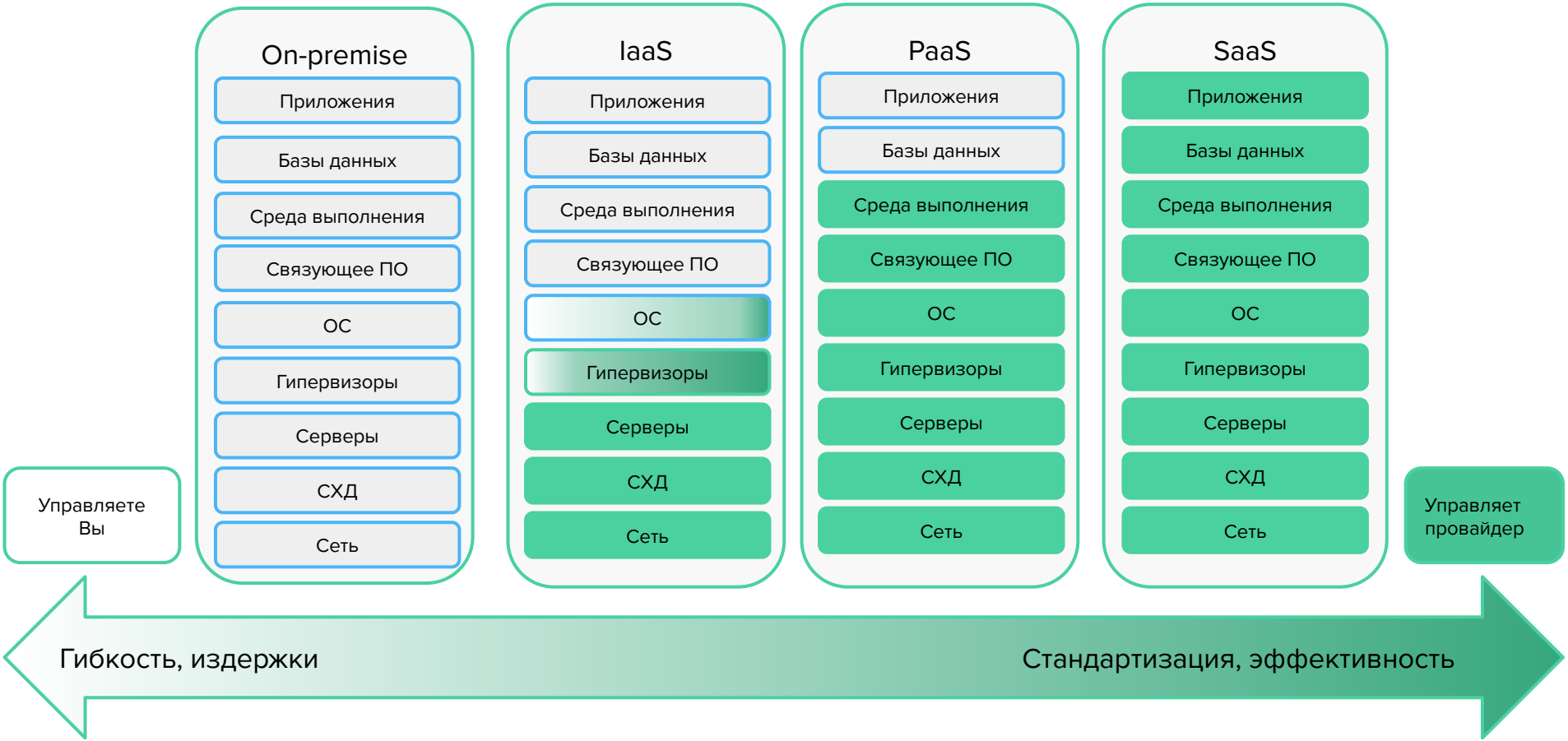
Cloud Computing



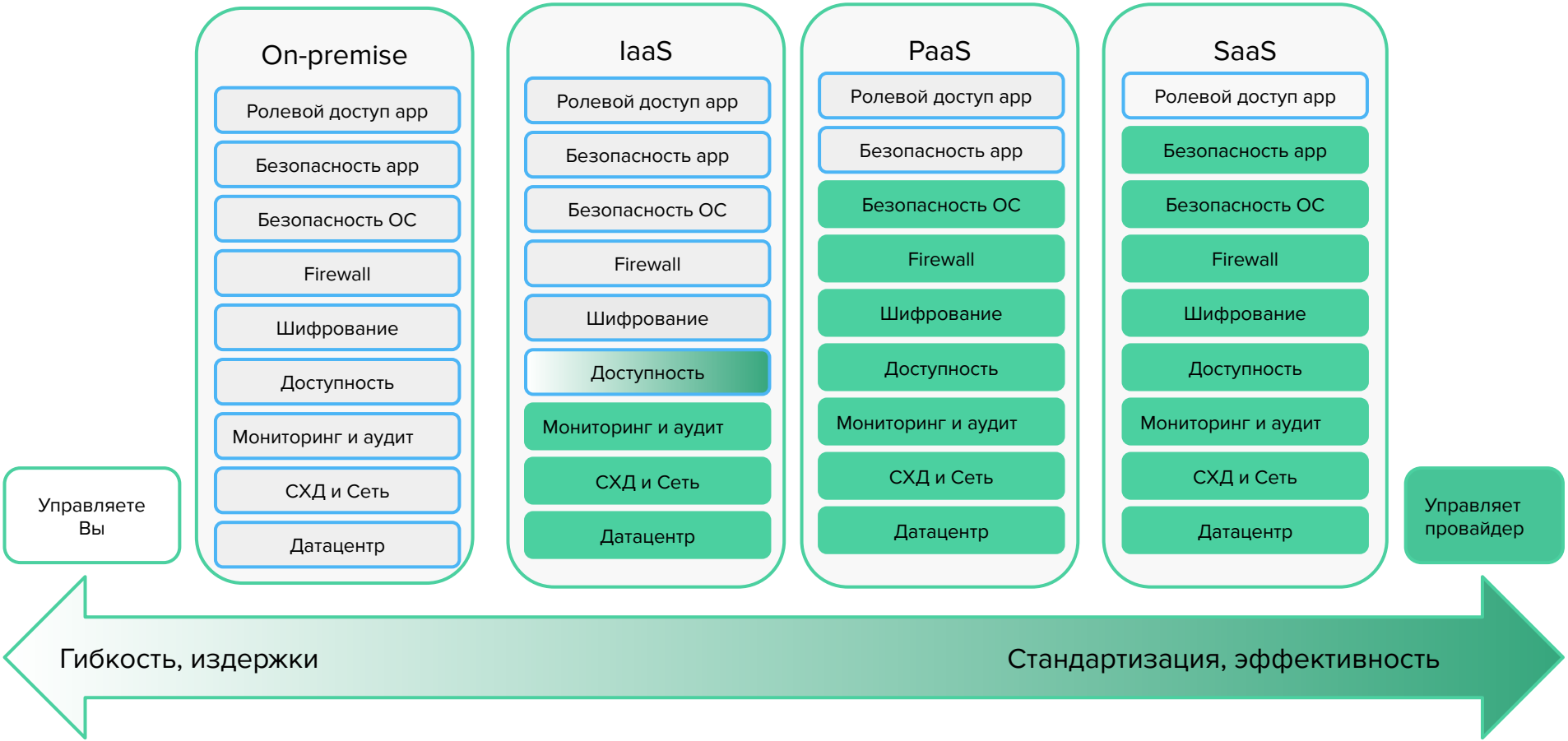
Типы облаков



Типы облачных услуг, границы управляемости



Распределение ответственности за безопасность



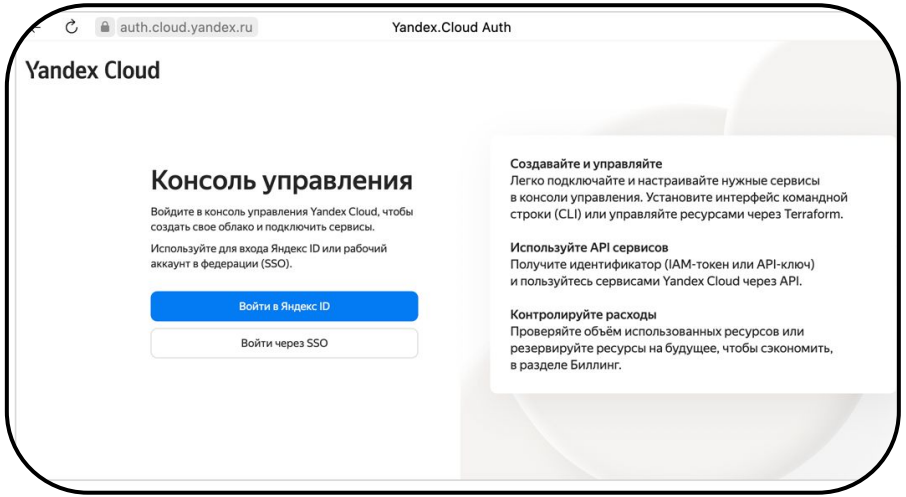
Топ облачных провайдеров РФ

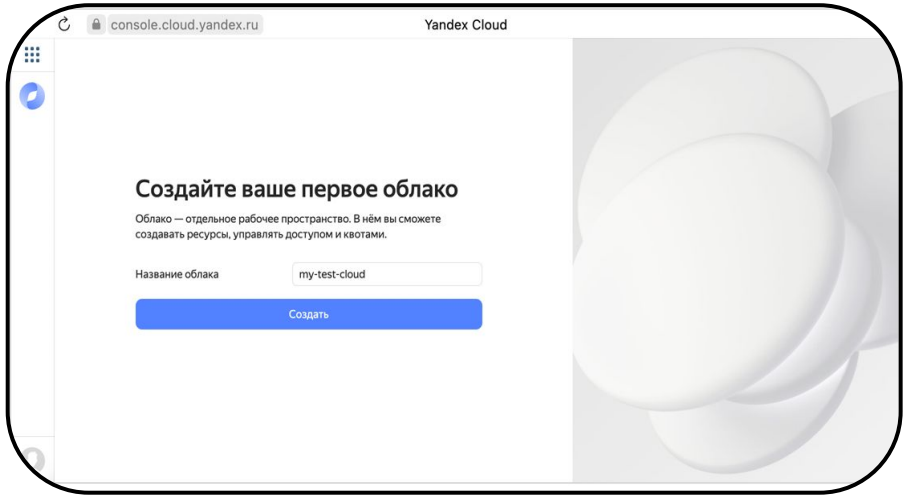
Основы работы с Yandex Cloud

2

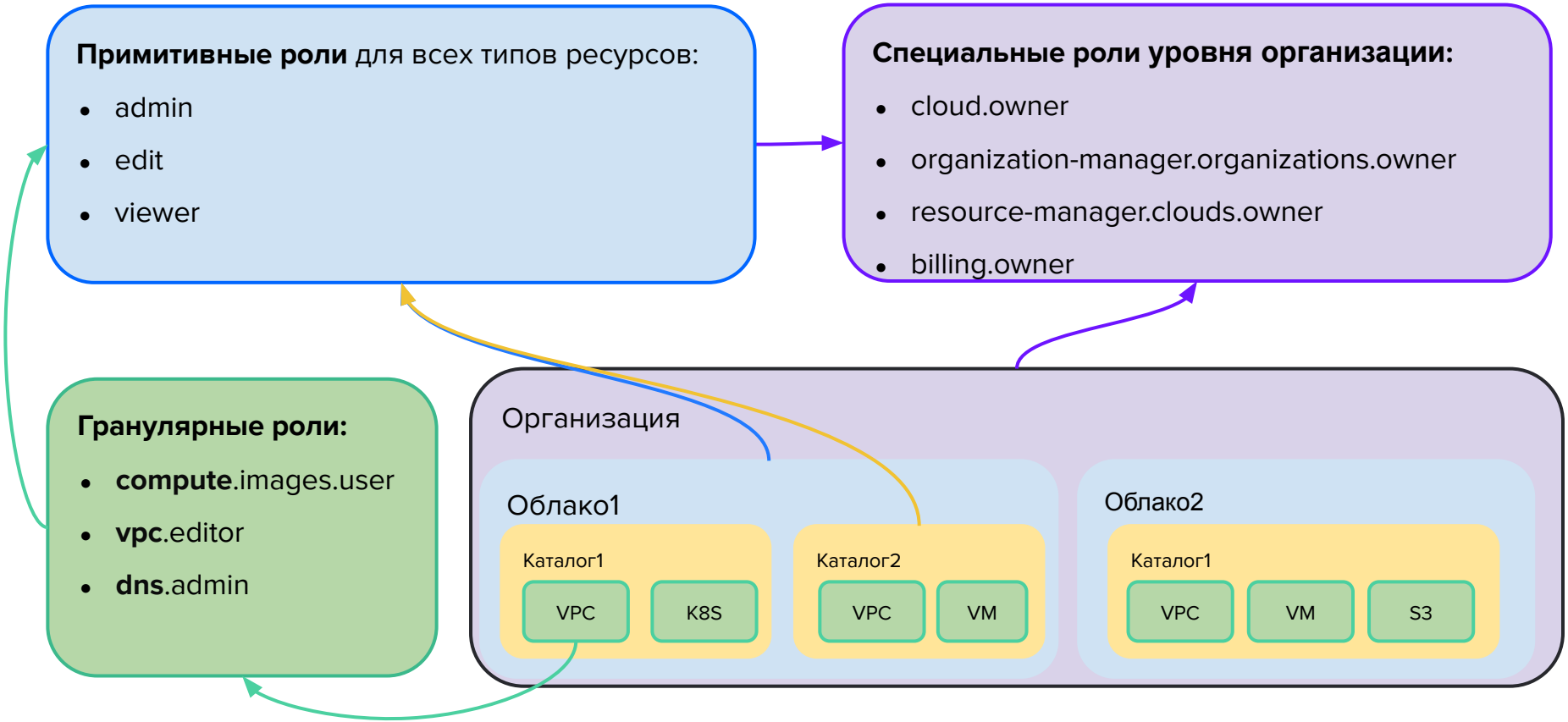
Регистрация аккаунта

Вам будет назначена роль владельца — **cloud.owner**

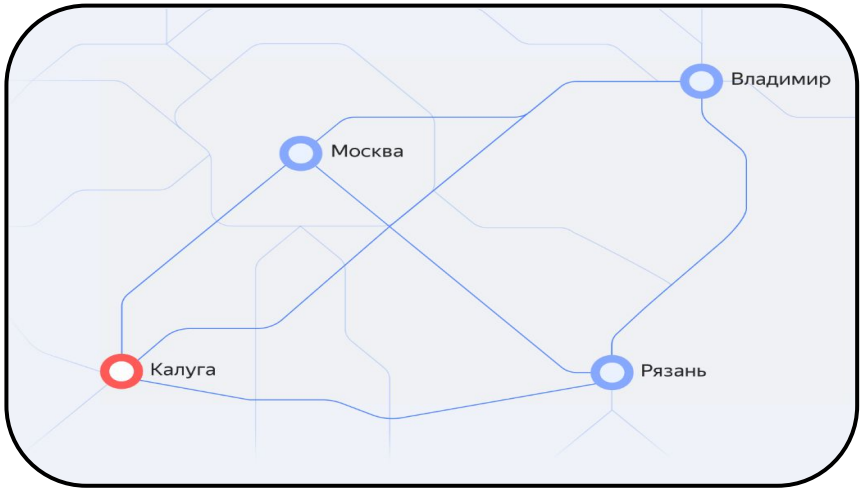




Иерархия ролей и ресурсов



Зоны доступности

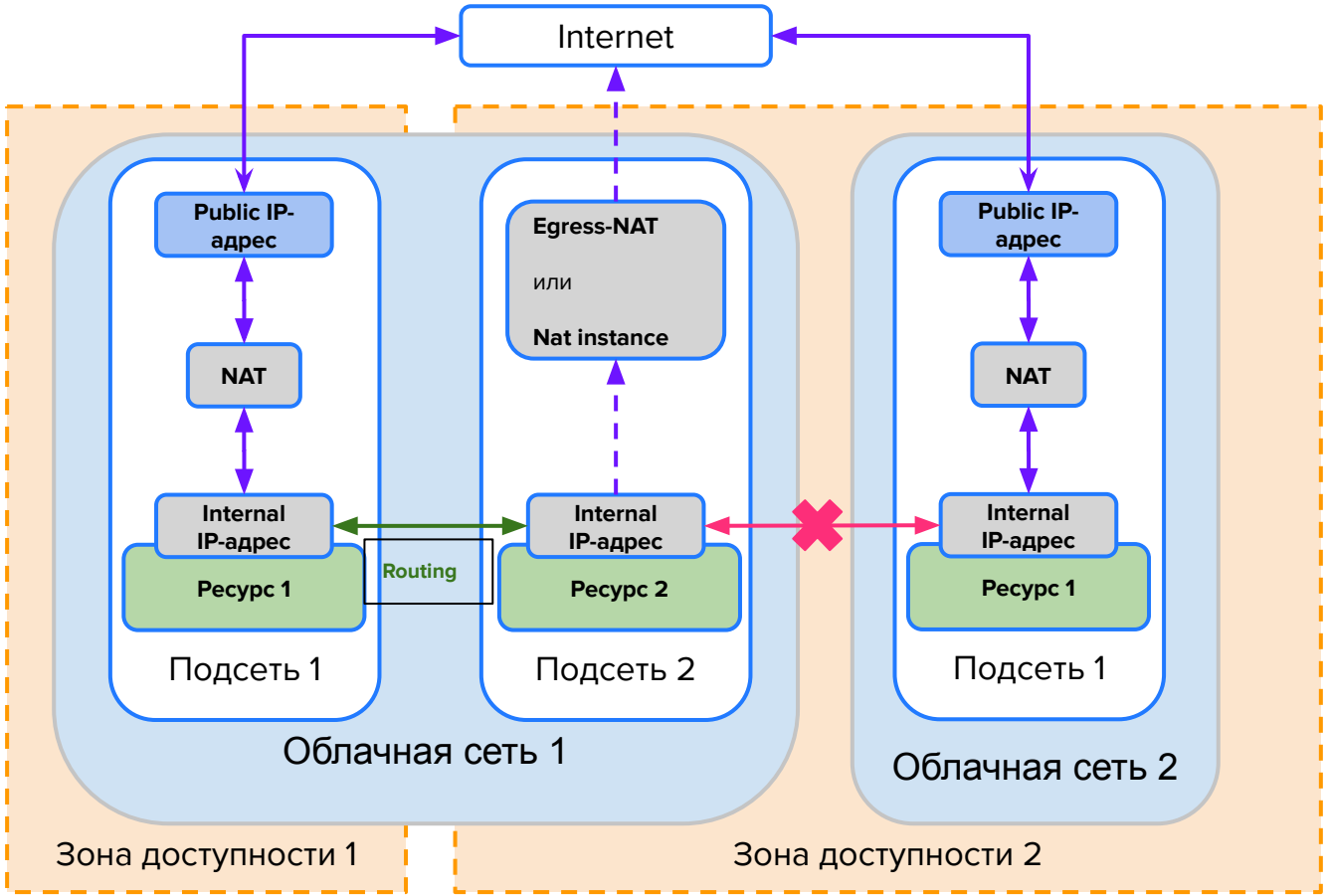


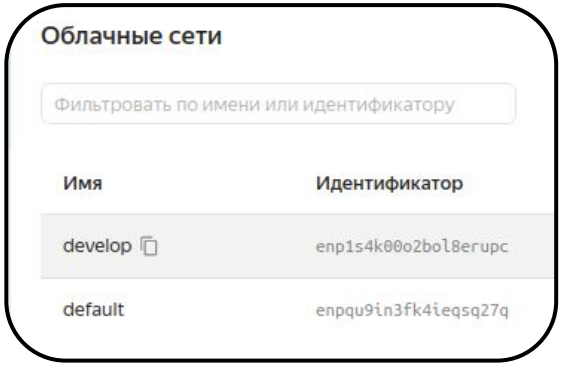
Регион	Зоны	Географическая привязка

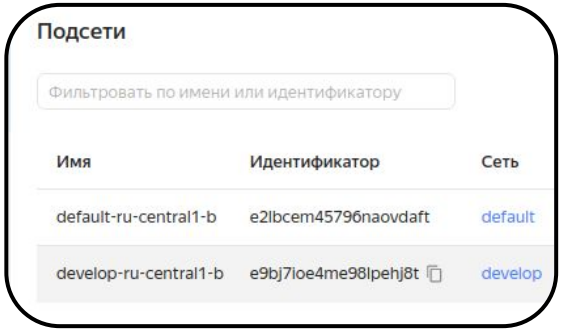
`ru-central1`	`ru-central1-a`
	`ru-central1-b`
	`ru-central1-c`
		Владимирская, Московская и Рязанская области.

Источник

Virtual Private Cloud (VPC)







Настройка yandex provider для авторизации в YC

1. Установите yc-tools
2. Получите OAuth-токен (срок жизни 1 год, используется для первичной настройки, можно перевыпустить)
3. Получите идентификатор облака
4. Получите идентификатор каталога
5. Создайте профиль для yc-tools
6. Используйте команду **yc iam create-token** для получения iam-токена (срок жизни 12 часов).

Аутентификация для terraform provider на примере yandex

Используйте полученные данные в блоке **provider {..}**

```
provider "yandex" {
#Небезопасный способ, только для изучения!
   token     = "<iam-token или OAuth-токен>"
   cloud_id  = "<идентификатор_облака>"
   folder_id = "<идентификатор_каталога>"
   zone      = "ru-central1-a"
 }
```

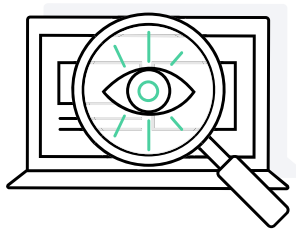
«Хардкодим» первые облачные ресурсы

```
#main.tf #добавить авторизацию!!
terraform {
  required_providers {
    yandex = { source = "yandex-cloud/yandex"
  }}
  required_version = ">= 0.13"
}

#создаем облачную сеть
resource "yandex_vpc_network" "develop" {
  name = "develop"
}

#создаем подсеть
resource "yandex_vpc_subnet" "example" {
  name           = "develop-ru-central1-a"
  zone           = "ru-central1-a"
  network_id     = yandex_vpc_network.example.id
  v4_cidr_blocks = ["10.0.1.0/24"]
}

#считываем данные об образе ОС
data "yandex_compute_image" "ubuntu-2004-lts" {
  family = "ubuntu-2004-lts"
}
.........
```

```
.........
#создаем ресурс ВМ
resource "yandex_compute_instance" "example" {
  name        = "netology-develop-platform-web"
  platform_id = "standard-v1"

  resources {
    cores  = 2, memory = 1, core_fraction = 5
  }

  boot_disk {
    initialize_params {
      image_id =
data.yandex_compute_image.ubuntu-2004-lts.image_id
      type = "network-hdd"
      size = 5
    }
  }

  scheduling_policy { preemptible = true }

  network_interface { subnet_id =
yandex_vpc_subnet.example.id }

}
```

Демонстрация работы

Создание ресурсов в YC с помощью Terraform без использования input variables(хардкод)



Переменные в Terraform

3

Input variables (входящие переменные) позволяют параметризировать значения аргументов при выполнении кода

Блок input variables

Объявляются блоком кода **variable "<уникальное_имя>" {..}**, содержащим:

- type – тип переменной (опционально)
- description – назначение (опционально)
- default – значение по-умолчанию (опционально)
- иные опциональные (validation, sensitive, nullable)

```
variable "image_name" {
  type          =  "string"
  description   =  "ubuntu release name"
  default       =  "ubuntu-2004-lts"
}
```

Обращение к input variables

Использование **input variable** в коде проекта: **var.<имя_переменной>**

```
data "yandex_compute_image" "ubuntu-2004-lts" {
  family = var.image_name
}
```

Переменные окружения TF_VAR_

```
Формат: export TF_VAR_image_name = "ubuntu-2004-lts"

Безопасная аутентификация для yandex provider:

export TF_VAR_yc_token     = $(yc iam create-token)
export TF_VAR_yc_cloud_id  = $(yc config get cloud-id)
export TF_VAR_yc_folder_id = $(yc config get folder-id)

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

Переменные окружения можно использовать для изменения некоторых параметров работы Terraform. Например, для детализации логов работы: **export TF_LOG=trace**

TFVARS Files

Terraform загружает переменные из tfvars-файлов модуля:

- terraform.tfvars – файл по-умолчанию
- *.auto.tfvars – именованные файлы

Любые другие tfvars файлы игнорируются.

Чтобы **безопасно** записать аутентификационные данные, например, в файл personal.auto.tfvars – **добавьте его в .gitignore**.

TFVARS Files

Вы можете загрузить дополнительные файлы переменных, указав путь с помощью одного или нескольких флагов **-var-file.**

Таким образом можно подгружать переменные для разных окружений

```
terraform apply -var-file=./develop/env.tfvars


terraform apply -var-file=./prod/env.tfvars  -var-file=/prod/additional.tfvars
```

Порядок загрузки значений переменных

1. Считывание default значения из блока **variable**
2. Считывание из Environment переменных `export TF_VAR_image_name="ubuntu-2004-lts"`
3. Считывание переменных из файла terraform.tfvars (если существует)
4. Считывание переменных из автоматически загружаемых файлов tfvars
5. Считывание с командной строки `terraform apply -var="image_name=ubuntu-2004-lts"`

Порядок загрузки значений переменных

Если переменная не определена — terraform запросит её в командной строке.

Если одна и та же переменная определена разными методами, действует последнее в порядке загрузки значение. Исключение для переменных типа map: их значения объединяются

Организация кода проекта

Для удобства принято группировать блоки по типам и назначению в отдельные файлы ***.tf.** Как именно — решает команда. Но технически весь код можно сложить в один файл.

outputs.tf	Блоки output
locals.tf	Локальные переменные
providers.tf	Блок провайдеров
main.tf	Root модуль
variables.tf	Описание **общих** переменных
vpc.tf I **vm.tf ..**	Именованные файлы для ресурсов и/или переменных
db.auto.tfvars **personal.auto.tfvars**	Именованные файлы для загрузки переменных. **Если они содержат личные токены доступа — добавьте их в .gitignore**

Базовые типы переменных

string	Последовательность символов **в кавычках**. Пример: `"hello"`
number	Целые числа и десятичные дроби, **без кавычек**. Пример: `10`
bool	Логическое значение, **без кавычек**. Пример: `true or false`
null	Несуществующее (незаданное) значение, **без кавычек**
any	Допускается любой тип данных, значение по-умолчанию

Комплексные типы переменных. Collections

1 `list(<VAR_TYPE>)`

Список — **индексированный** набор значений **одинакового** типа.

Пример:

```
list(string)=["abc", "123"]
```

Обращение к индексу списка:

```
<тип переменной>.<имя переменной>[индекс]
```

Пример:

```
var.list[0]
```

Комплексные типы переменных. Collections

2 **`map(<VAR_TYPE>)`**

Набор уникальных ключей = «строковое значение».

Пример:

```
dict = { first = "A", second = "B" }
```

Обращение к элементу словаря:

```
<тип переменной>.<имя переменной>.<ключ>
```

Пример:

```
var.dict.first или var.dict["first"]
```

Комплексные типы переменных. Collections

3 **`set(<VAR_TYPE>)`**

Не индексированный набор **уникальных** значений **одинакового** типа.

Примеры:

```
set(string) = ["abc", "123"]      set(number) = [1, 2, 3]
```

```
set(list) = [ ["abc", "123"], [1, 2, 3] ]
```

Операция преобразования **list -> set** , удаляет дубликаты.

Комплексные типы переменных. Structural

4 **`tuple([<VAR_TYPE>, …])`**

Неизменяемый, индексированный список значений любого типа, соответствующих структуре переменной.

Пример:

```
tuple([string, number, bool])

default = ["abc", 123, false]
```

Комплексные типы переменных. Structural

5

```
object({<ATTR_NAME>=<VAR_TYPE>, … }
```

Неизменяемый набор уникальных «ключ = любый тип данных», соответствующих структуре переменной.

Пример:

```
object({ name = string, age = number, married = bool})

default = { name = "John Connor", age = 15, married = false}
```

Возможны еще более сложные, многократно вложенные комбинации. Но в этом случае часто используют тип **any.**

map(object(list(string))) или map(**any**)

Любой объект при создании «возвращает» в **State** значения своей конфигурации.
С помощью блока **output** можно записать эти значения или их часть в **output-переменную** для последующего использования

Блок output variables

Объявляются блоком **output "" {..}**, содержащим:

- **value** = значение переменной, может быть string, list, или map

После применения конфигурации, terraform отобразит в консоли значения всех объявленных **outputs в Root Module**.

С помощью команды **terraform output** можно отобразить в консоли значения объявленных outputs из State

Пример:

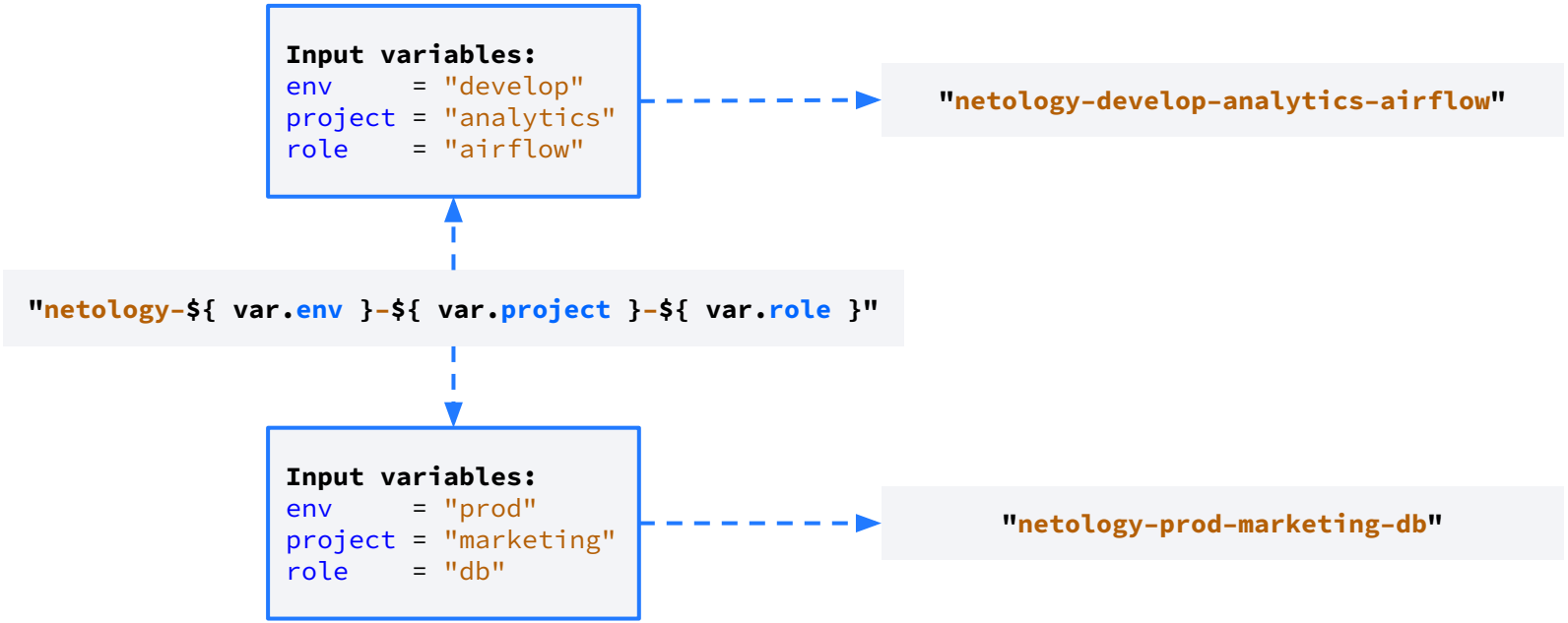
```
output "vm_external_ip_address" {
  value =
yandex_compute_instance.vm.*.network
_interface.[0].nat_ip_address
  description = "vm external ip"
}
```

В программировании **интерполяция** — это вычисление значения строки путем **подстановки** переменных в шаблон

String Interpolation

С помощью конструкции **"${ ... }"** можно составить шаблон строки с использованием условий, функций и выражений.

Пример универсального шаблона названия виртуальной машины:



Input variables

Input variables не могут принимать в качестве значения вычисляемое выражение. Т.е. не могут содержать в себе переменные, функции и прочие выражения.

Недопустимый пример:

```
variable "bad_example" { default = "${ var.env }-${ var.project }"}
```

Локальные переменные позволяют присвоить переменной вычисляемое выражение для дальнейшего использования в области видимости модуля

Блок locals

Локальные переменные объявляются блоком **locals { .. }** , содержащим **одну** или **несколько** переменных в виде ключ=значение.

Количество блоков locals не ограничено, но чем больше вы их используете — тем сложнее понимать чужой код. **А спустя время и свой собственный**!

Обращение к **local** переменной в модуле:
local.<Имя Переменной>

```
local.my_value
local.default_tags
```

```
locals {
  my_value     = 10
  default_tags = {
    env        = var.env
    project    = var.project
    managed_by = "terraform"
 }
}
```

Интерактивная консоль Terraform

Команда **terraform** console открывает интерактивную оболочку.

Она позволяет тестировать текущую конфигурацию без необходимости выполнения кода проекта.

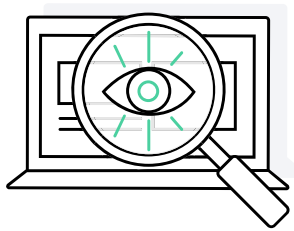
Вы можете:

- интерполировать переменные проекта
- считывать данные из state файла
- использовать функции и выражения

```
Примеры:
> "${abs(5-15)}"
10

> length("qwerty")
6

> split( ",", "abc,123,@#$" )
tolist([  "abc",  "123",  "@#$",])

> element(["1a","2b","3c","4d"], 3)
"4d"
```

Демонстрация работы

- Интерактивная консоль Terraform
- Разбор типов переменных на практике



Итоги занятия

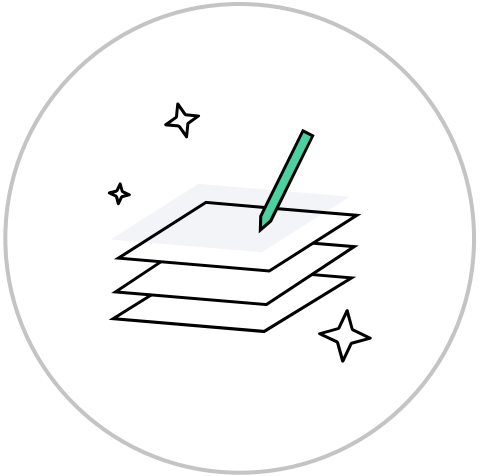
Сегодня мы

1. Рассмотрели виды облачных вычислений
2. Научились создавать ресурсы в Yandex Cloud с помощью Terraform
3. Разобрали типы переменных в Terraform

Домашнее задание

Давайте посмотрим ваше домашнее задание.

1. Вопросы по домашней работе задавайте в чате группы
2. Задачи можно сдавать по частям
3. Зачёт по домашней работе ставят после того, как приняты все задачи



Дополнительные материалы

Документация:

- input variables
- outputs
- local variables

Задавайте вопросы и пишите отзыв о лекции

Елисей Ильин
DevOps-инженер в Itransition