5% Акция действует до 30 июня

Infrastructure as a code

Знакомство с Terraform

otus.ru

Проверить, идет ли запись

Хорошо видно && слышно?

Ставим “+”, если все хорошо
“-”, если есть проблемы

Тема открытого урока

Знакомство с Terraform

Алексей Журавлев

Руководитель группы разработчиков корпоративного ПО в области финтеха

Опыт работы: ИДСиТУ СО РАН, Финансово-кредитные организации, IT-компании

Эл. почта: *Zhuravlev.A.E@gmail.com*
Телеграм: *@Aleksey_Jhuravlev*

Правила вебинара

Активно
участвуем

Задаем вопрос
в чат

Вопросы вижу в чате,
могу ответить не сразу

Маршрут вебинара

1. Знакомство, об Отус
2. Введение в Terraform
3. Основные концепции и принципы работы Terraform
4. Практическая работа
5. Процесс обучения в Отус
6. О курсе, программа обучения
7. Рефлексия

Расскажите о себе

- Ваш опыт работы в IT?
- С какой основной целью вы записались на занятие?

Об ОТУС

О компании

Сфера

ОТУС специализируется на обучении в IT.
Наша фишка — продвинутые программы для специалистов с опытом и быстрый запуск курсов по новым технологиям.

Образовательная лицензия

У OTUS есть образовательная лицензия, поэтому вы сможете получить удостоверение о повышении квалификации или диплом о профессиональной переподготовке, а также сделать налоговый вычет.

Направления курсов

Обучение специалистов разных грейдов: junior, middle, senior, lead

- Программирование
- Инфраструктура
- Тестирование
- Аналитика

- Data Science
- Управление
- GameDev
- Информационная безопасность

Мы в цифрах

170+

курсов для junior, middle, senior специалистов

600+

преподавателей делятся знаниями и реальными кейсами

7

лет со дня основания компании

38 000+

выпускников уже прошли обучение

500 000+

ИТ-специалистов в сообществе, читают материалы, учатся и общаются на наших площадках

Знакомство с Terraform

Цели вебинара

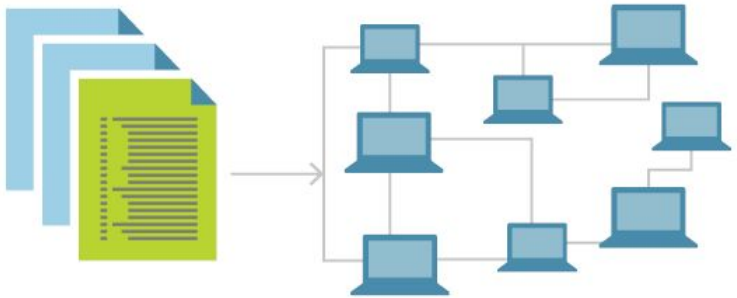
1. Познакомиться с основными концепциями Terraform, узнать, как он работает.
2. Попробовать создать на практике свою первую инфраструктуру в облаке с помощью Terraform и увидеть, насколько просто и удобно управлять ею с помощью кода.

Введение в Terraform

краткое напоминание:

Инфраструктура как код (англ. **Infrastructure as a code, IaC**):

Это подход для управления и описания IT-инфраструктуры через ***отдельно*** хранимые конфигурационные файлы

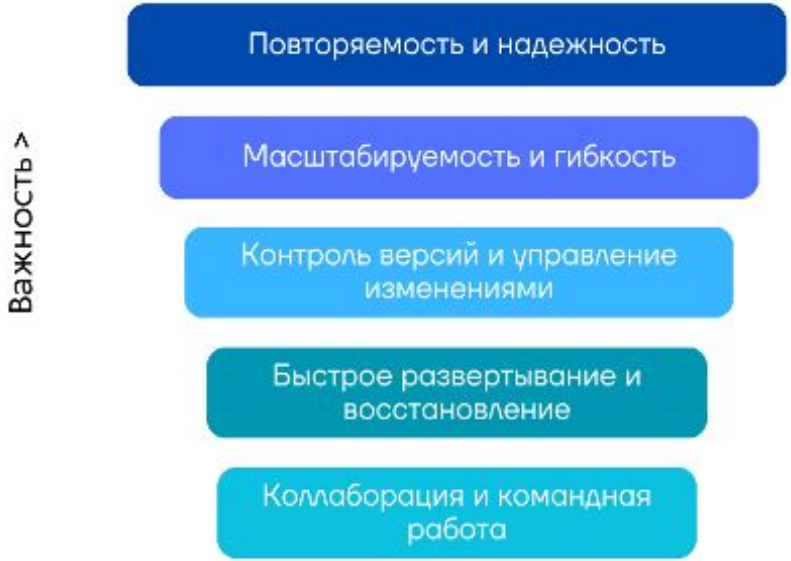


Подход Infrastructure as a ~~Кот~~Code

включает:

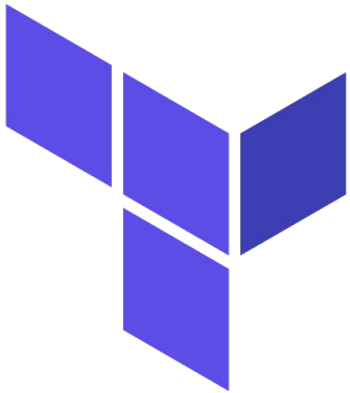
- Применение инструментов для IaC
- Описание инфраструктуры только в виде кода
- Использование инфраструктурного репозитория
- Применение практик из разработки ПО

Преимущества Инфраструктуры как кода



Примеры инструментов IaC:

- Terraform (множество провайдеров)
- Pulumi (для различной облачной инфраструктуры)
- AWS CloudFormation (только AWS)
- Heat (только OpenStack)
- Google Cloud Deployment Manager (только GCP)
- SCM (Ansible, Chef, Puppet, SaltStack)



HashiCorp

Terraform

- ПО для **управления** внешними **ресурсами** в рамках модели **IaC,** лицензия **BSL**
- Создано и поддерживается компанией **HashiCorp**
- Пользователи определяют и предоставляют IT-инфраструктуру с помощью **декларативного** языка конфигурации **HCL**

Почему Terraform?

- Выразительный
- Мощный
- Универсальный
- Распространенный
- Богатое окружение
- …

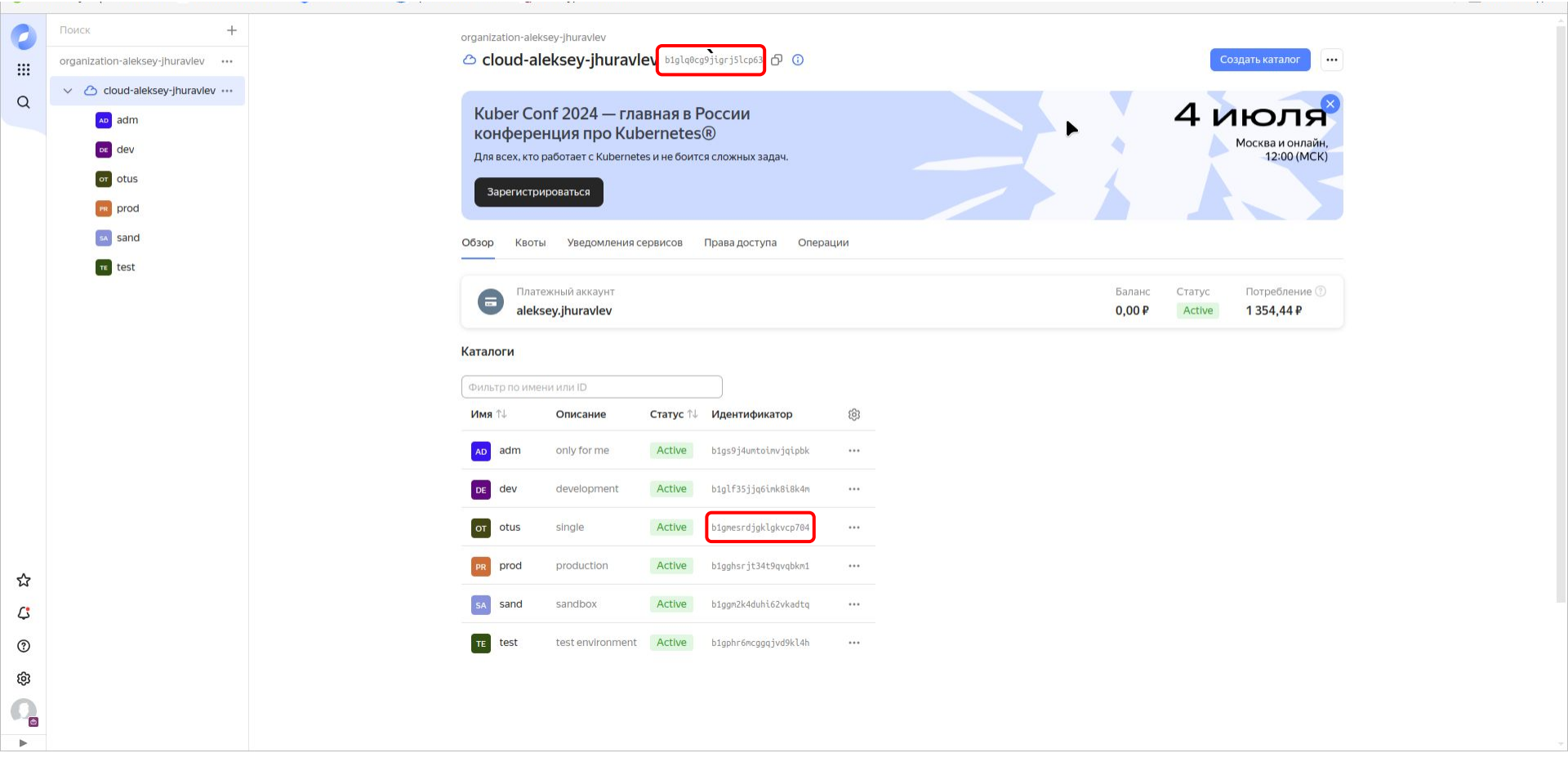
Создание ВМ с помощью terraform в AWS:

```
provider "aws" {
 region = "us-east-1"
}

resource "aws_instance" "example" {
 ami           = "ami-0c55b159cbfafe1f0"
 instance_type = "t2.micro"

 tags = {
   Name = "example-instance"
 }
}
```

Исходные данные в консоле. Часть 1:

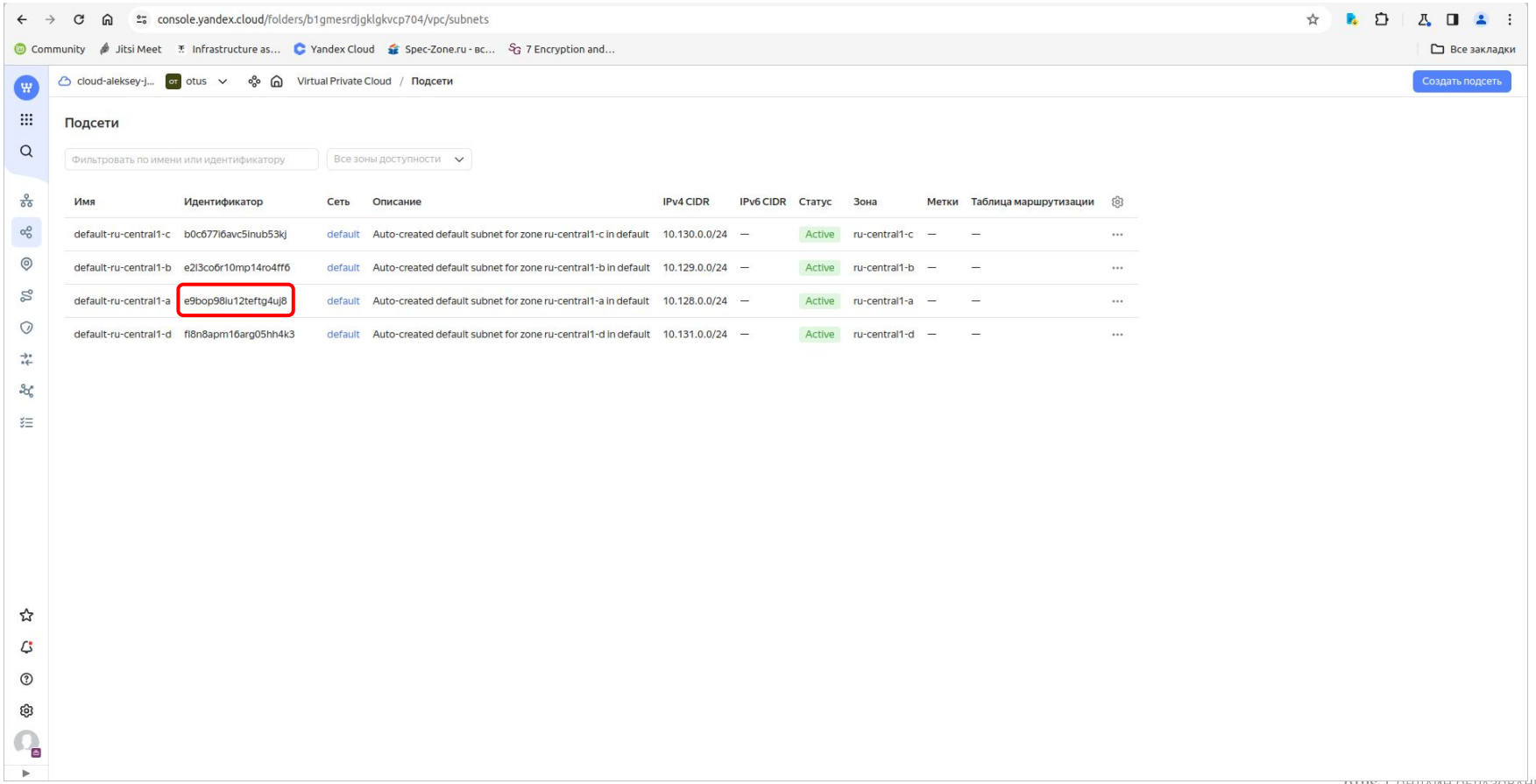


Исх. данные в конфигурационном файле:

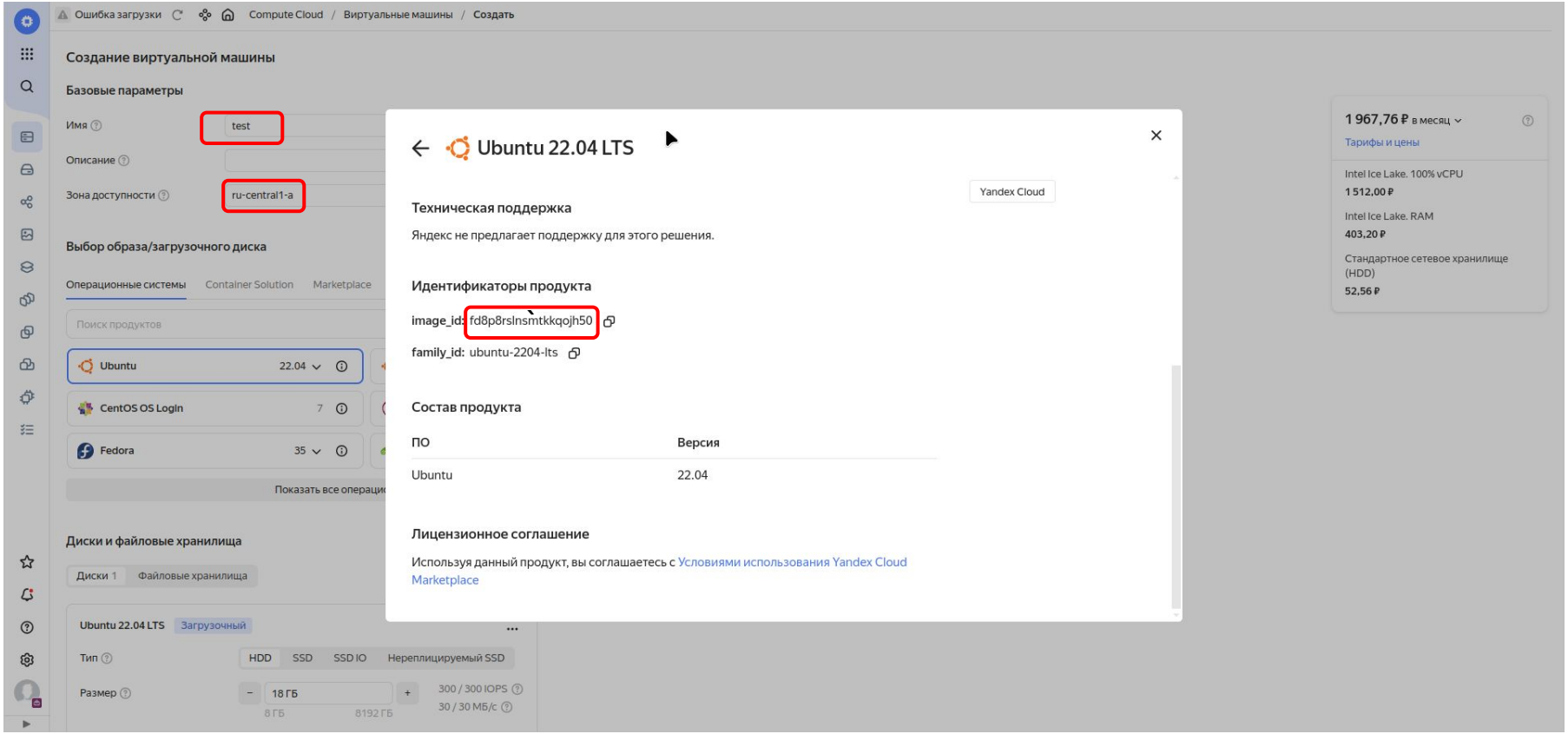
```
provider "yandex" {
  cloud_id  = "b1glq0cg9jigrj5lcp63"
  folder_id = "b1gmesrdjgklgkvcp704"
  zone      = "ru-central1-a"
}

terraform {
  required_providers {
    yandex = {
      source = "yandex-cloud/yandex"
    }
  }
  required_version = ">= 0.13"
}
```

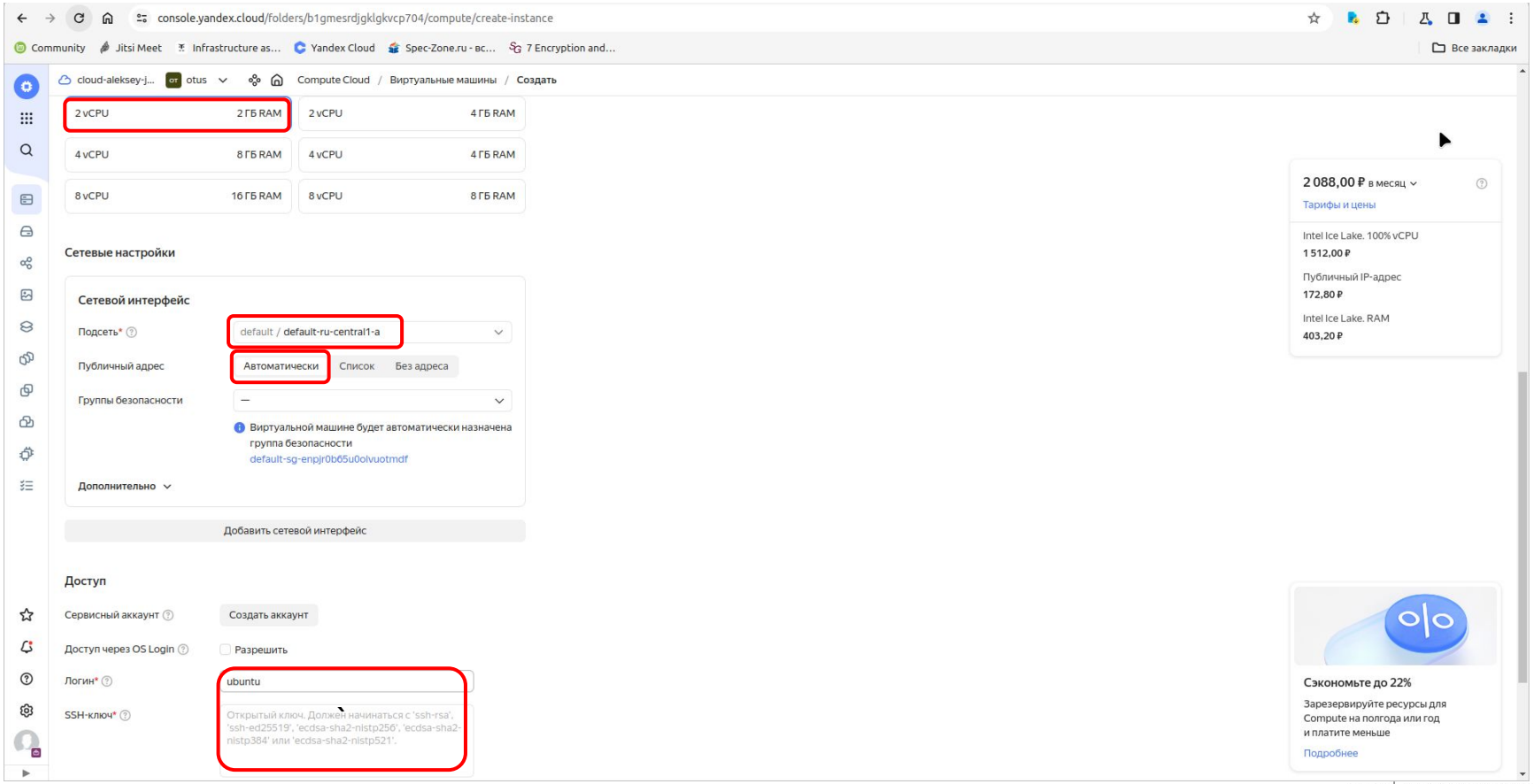
Исходные данные в консоле. Часть 2:



Создание ВМ в консоле. Часть 1:



Создание ВМ в консоле. Часть 2:



Создание ВМ в конфигурационном файле:

```
resource "yandex_compute_instance" "test" {
  name = "test"

  resources {
    cores  = 2
    memory = 2
  }

  boot_disk {
    initialize_params {
      image_id = "fd8p8rslnsmtkkqojh50"
    }
  }

  network_interface {
    subnet_id = "e9bop98iu12teftg4uj8"
    nat       = true
  }

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

}
```

Выводы:

1. Консольный способ позволяет на скорую руку создавать инфраструкту, в соответствующих формах облачной консоли и заполняя поля или выбирая значения из списка предлагаемых. Лучше подходит для незначительной временной инфраструктуры, не требующей развития и сопровождения.
2. При создании инфраструктуры методом IaC с использованием Terraform фактически ту же самую информацию мы прописываем в конфигурационных файлах. Лучше подходит для масштабной инфраструктуры, требующей развития и сопровождения.

Вопросы?

Ставим “+” или пишем
вопросы если есть

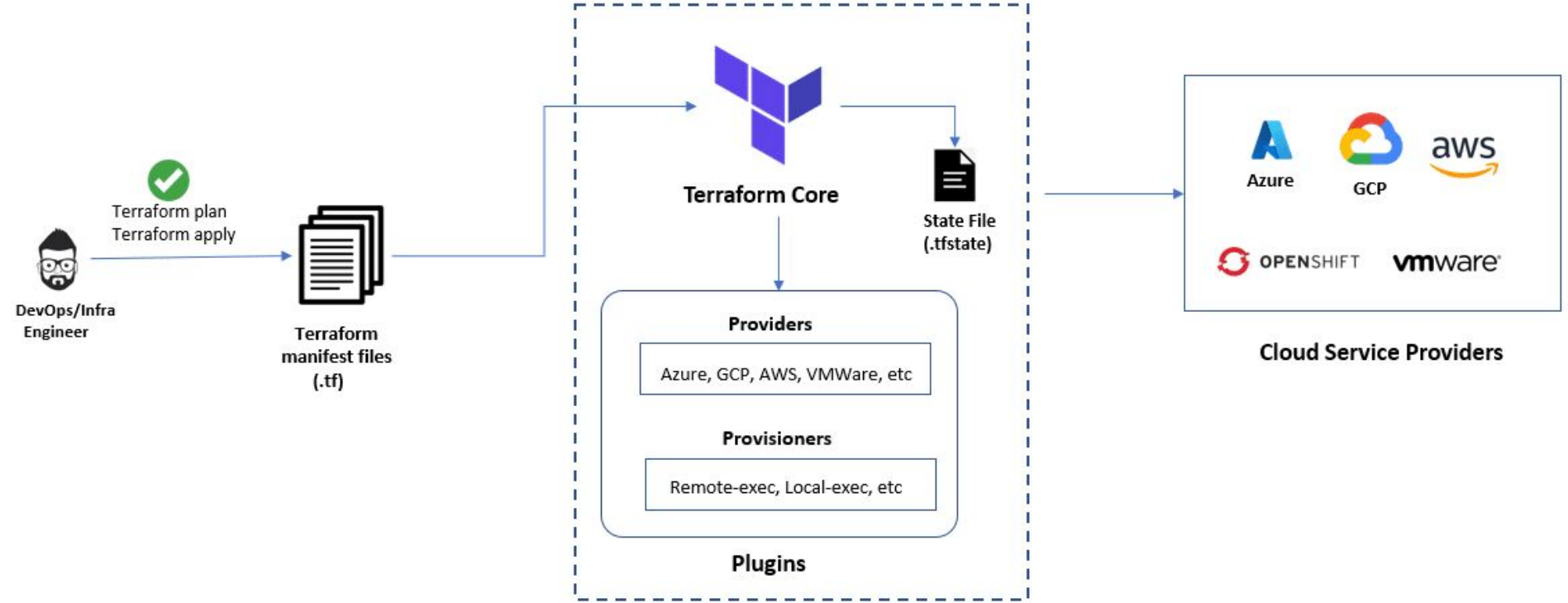
Ставим “–”,
если вопросов нет

Основные концепции и принципы работы Terraform

Основные понятия

- **Провайдер**
- **Описание инфраструктуры** (хранится в конфигурационных **.tf**-файлах)
- Понятие **State**

Terraform Architecture



Язык HCL:

- JSON-based синтаксис
- Блоки
- Аргументы
- Переменные
- Функции
- Выражения
- Комментарии

```
resource "aws_vpc" "main" {
  cidr_block = var.base_cidr_block
}

<BLOCK TYPE> "<BLOCK LABEL>" "<BLOCK LABEL>" {
  # Block body
  <IDENTIFIER> = <EXPRESSION> # Argument
}
```

Формат блоков:

```
provider "Имя_провайдера" {
  Ключ = Значение
}
```

```
resource "Тип_ресурса" "Имя ресурса" {
  Ключ = Значение
}
```

```
variable "Имя_переменной" {
  type    = "Тип_переменной"
  default = "Значение"
}
```

Переменные и типы:

Переменные

- Входные аргументы `variable {}`
- Выходные значения `output {}`
- Локальные переменные `local {}`

Типы

- Простые (primitive): **string**, **number**, **bool**
- Сложные (complex): `list(string)`, `set()`, `map()`, `object()`, `tuple()`

Встроенные функции:

- Numeric
- String
- Collection
- Encoding
- Filesystem
- Date and Time
- IP Network
- Type Conversion

Ссылки и использование переменных:

- `<RESOURCE TYPE>.<NAME>`
- `var.<NAME>`
- `local.<NAME>`
- `self.<NAME>`
- `var.names[0]` или `var.names.0`
- `"instance ${google_compute_instance.test.name} created"`

Пример описания и использования variable:

```
variable "image_id" {
  type        = string
  description = "The id of the machine image (AMI) to use for the server."

  validation {
    condition     = length(var.image_id) > 4 && substr(var.image_id, 0, 4) == "ami-"
    error_message = "The image_id value must be a valid AMI id, starting with \"ami-\"."
  }
}
```

Пример описания выходной переменной:

```
output "instance_ip_addr" {
  value       = aws_instance.server.private_ip
  description = "The private IP address of the main server instance."
}
```

Пример описания локальных переменных:

```
locals {
  service_name = "forum"
  owner        = "Community Team"
}
```

Пример описания провайдера:

```
terraform {
  required_providers {
    mycloud = {
      source  = "mycorp/mycloud"
      version = "~> 1.0"
    }
  }
}

provider "mycloud" {
  # ...
}
```

Пример ресурса с local-exec provisioner:

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

  provisioner "local-exec" {
    command = "echo The IP-address of our web server is ${self.private_ip}"
  }
}
```

Еще пример ресурса с local-exec provisioner:

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

  provisioner "local-exec" {
    command = "echo The server's IP address is ${self.private_ip}"
  }
}
```

Пример ресурса с connection и remote-exec Provisioner:

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

  # Establishes connection to be used by all
  # generic remote provisioners (i.e. file/remote-exec)
  connection {
    type     = "ssh"
    user     = "root"
    password = var.root_password
    host     = self.public_ip
  }

  provisioner "remote-exec" {
    inline = [
      "puppet apply",
      "consul join ${aws_instance.web.private_ip}",
    ]
  }
}
```

Пример provisioner типа file:

```
# Copies the file as the Administrator user using WinRM
provisioner "file" {
  source      = "conf/myapp.conf"
  destination = "C:/App/myapp.conf"

  connection {
    type     = "winrm"
    user     = "Administrator"
    password = "${var.admin_password}"
    host     = "${var.host}"
  }
}
```

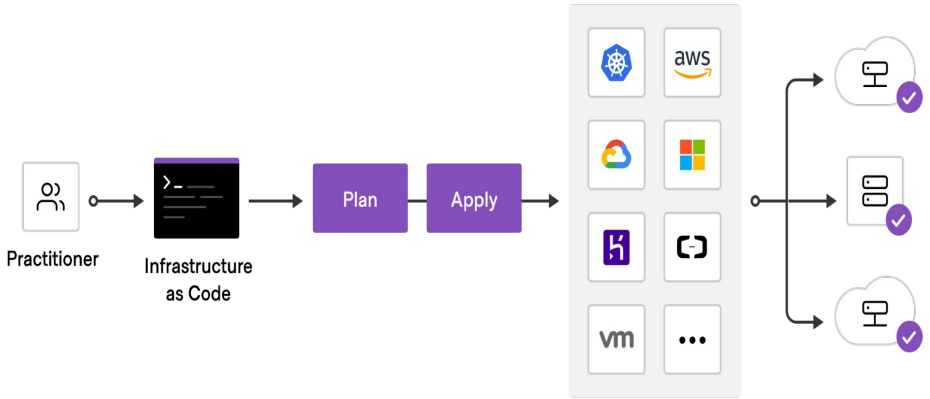
Каталог проекта:

- .terraform/
- main.tf
- variables.tf
- outputs.tf
- ….tf
- terraform.tfstate
- terraform.tfstate.backup
- terraform.tfvars

Основные команды:

terraform -h

- terraform init
- terraform validate
- terraform plan
- terraform apply
- terraform destroy
- terraform show
- terraform output



Другие возможности Terraform:

- Data sources
- Повторяющиеся описания
- Мета-аргументы
- Модули
- Troubleshooting, отладка
- Окружение от других производителей
- …

Выводы:

1. Terraform достаточно универсален для облачной инфраструктуры, но могут проблемы с bare-metal
2. Terraform проявляет лучшие свои качества, такие как выразительность и декларативность при работе с нижними уровнями инфраструктуры (например с виртуальными ресурсами), но не всегда с верхними уровнями (например с настройкой ПО).
3. С учетом п.2 Terraform хорошо сочетается с другими SCM, например Ansible.

Вопросы?

Ставим “+” или пишем
вопросы если есть

Ставим “–”,
если вопросов нет

Практическая работа: создание инфраструктуры в облаке с помощью Terraform

Стартовые условия:

- Эккаунт в YC
- Пустой тестовый каталог в YC
- Установленный Terraform и настроенный на эккаунт YC и провайдер YC

LIVE

И так, с помощью terraform мы опробовали:

- Запускать простейшую конфигурацию “hello world”.
- Запускать локальный и удаленный provisioner
- Создавать сеть, подсеть и виртуальные машины в них.
- Создавать 2 различных веб-сервера с балансировщиком.

Список материалов для изучения

1. Infrastructure as code: обзор опенсорсных инструментов
2. Immutable-инфраструктура и ее преимущества / Хабр
3. Terraform: новый подход к Infrastructure as code
4. Начинаем работать с Terraform
5. Yandex Cloud. Практические руководства. Управление инфраструктурой. Начало работы с Terraform
6. Terraform | HashiCorp Developer
7. Docs overview | yandex-cloud/yandex | Terraform | Terraform Registry
8. Обзор OpenTofu 1.7.0: установка, миграция с Terraform, ключевые особенности / Хабр

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Процесс обучения в Отус и знакомство с курсом

Процесс обучения

Обучение проходит онлайн по вечерам

В чате можно уточнять моменты, которые были непонятны на уроке

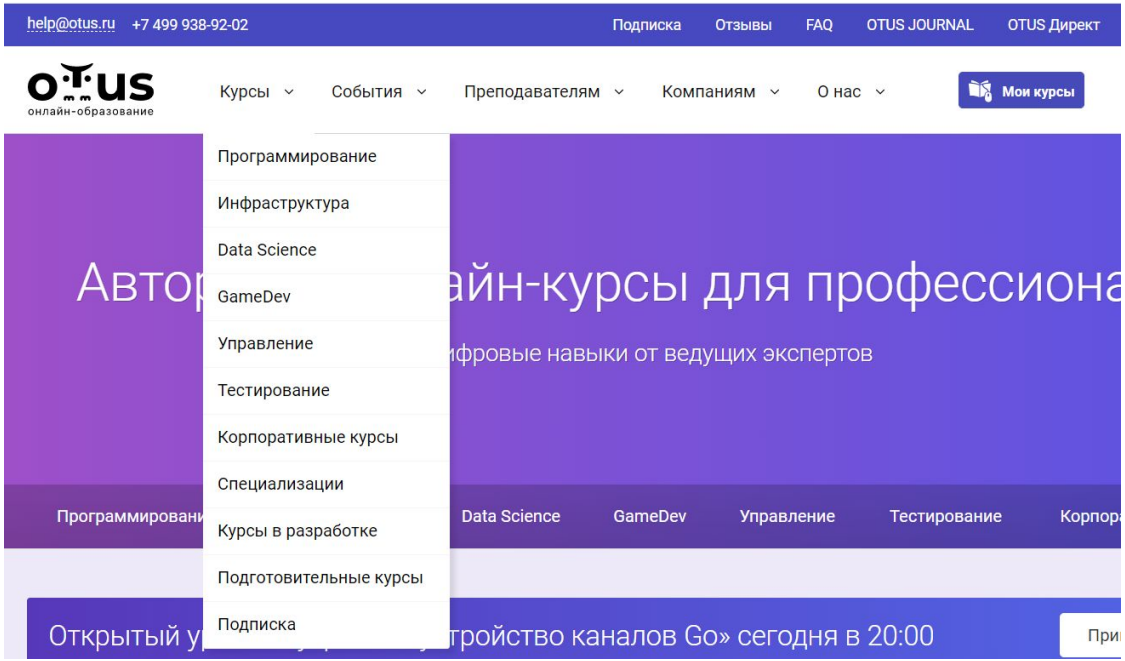
Все записи занятий и материалы сохраняются в личном кабинете навсегда

Время на обучение: от 4 ак. часов на занятия и 4-8 часов на домашнюю работу в неделю

По каждому домашнему заданию преподаватель даёт развернутый фидбек

Программа обучения на курсах обновляется каждый запуск

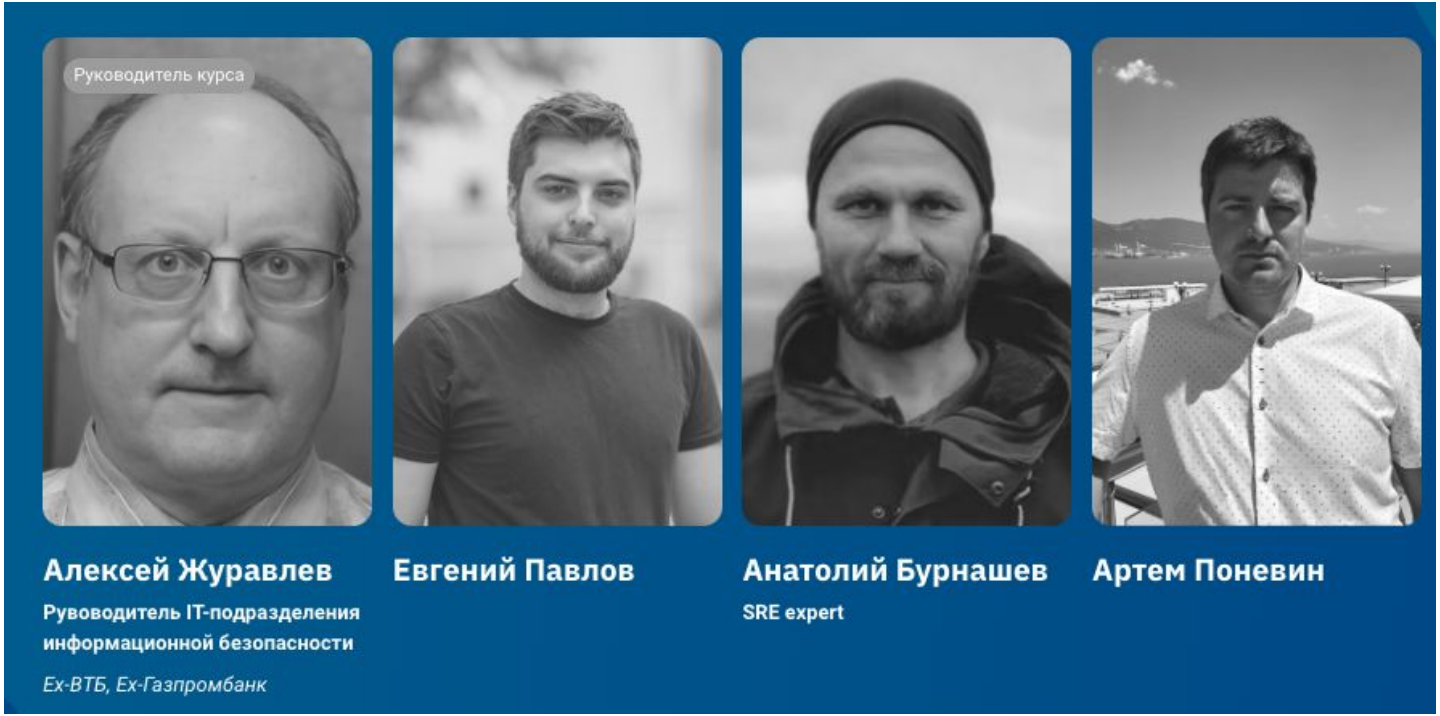
Познакомимся с курсом



Программа курса

- Введение в Infrastructure as Code
- Инструменты SCM
- Пайплайн для инфраструктуры как код
- Everything as Code
- Проектная работа

Преподаватели курса



Рефлексия

Цели вебинара

Проверка достижения целей

1. Познакомились с основными концепциями Terraform, узнали как он работает.
2. Попробовали создать на практике свою первую инфраструктуру в облаке с помощью Terraform и увидели, насколько просто и удобно управлять ею с помощью кода.

Рефлексия

С какими впечатлениями уходите с вебинара?

Как будете применять на практике то,
что узнали на вебинаре?

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

О курсе

Infrastructure as a code

старт обучения: *26.06.2024*

Заполните, пожалуйста,
опрос о занятии

Важно! Пройти опрос могут только залогиненные пользователи платформы OTUS

Спасибо за внимание!