

Облачные провайдеры и синтаксис Terraform



Андрей
Борю



Андрей Борю

Principal DevOps Engineer, Snapcart



План занятия

1. [Облачные провайдеры](#)
2. [Amazon EC2](#)
3. [Синтаксис Terraform](#)
4. [Структура проекта](#)
5. [Итоги](#)
6. [Домашнее задание](#)



Облачные провайдеры

AWS

- Популярное решение на зарубежном рынке
- Очень большое количество сервисов
- В первый год использования есть бесплатный тариф:
<https://aws.amazon.com/free/>.

Yandex.Cloud

- Популярное решение в русскоязычном сегменте
- Документация на русском языке
- Достаточное количество сервисов

План действий

- Рассмотрим работу с ресурсами облачных провайдеров через командную строку и веб-консоль
- Поймем, в чем сложности такого подхода
- Разберемся, как Terraform упростит нам жизнь

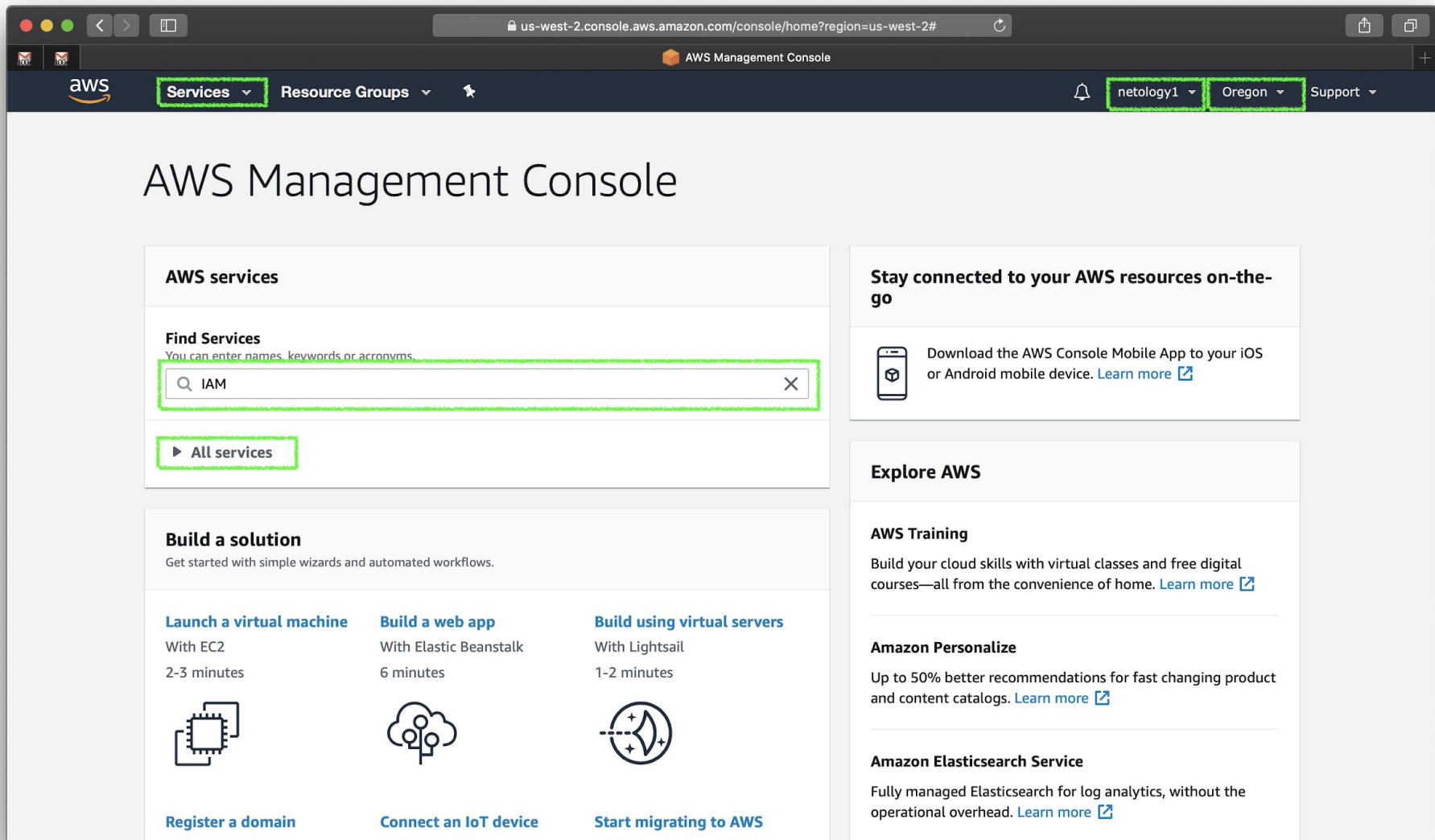
Регистрация в AWS

- Кредитная карта нужна только для регистрации
- Пользуемся бесплатным тарифом, который доступен год после регистрации
- Можно зарегистрировать отдельный «учебный» аккаунт на email типа «yourname+**netology**@gmail.com»

Регистрация в Yandex.Cloud

- Есть бесплатный пробный период
- Далее каждому студенту будут выданы промокоды

Элементы управления AWS



Элементы управления Яндекс.Сloud

The screenshot displays the Yandex Cloud console interface. The top navigation bar includes the Yandex Cloud logo, a search bar for cloud resources, and user account information (default netology-demo-01). The left sidebar lists various managed services under the heading 'Консоль управления'. The main content area features a 'Yandex Scale' announcement banner, a 'Ваши ресурсы' (Your Resources) section with a table of active resources, and a right sidebar with a 'Блог' (Blog) and 'Документация' (Documentation) section.

Консоль управления

- Compute Cloud
- Managed Service for PostgreSQL
- Managed Service for MongoDB
- Managed Service for ClickHouse
- Managed Service for Redis
- Managed Service for MySQL
- Managed Service for Kafka
- Managed Service for Elasticsearch
- Managed Service for SQL Server
- Managed Service for Greenplum
- Managed Service for Kubernetes
- Managed Service for GitLab
- Data Proc
- Yandex Database

Yandex Scale

Все главные новости Yandex.Cloud в одном месте. Смотрите запись Yandex Scale.

[Смотреть](#)

Ваши ресурсы

[Перейти в текущий каталог](#)

Статус	Баланс	Осталось	
Пробный период	1000 ₽ / 3000 ₽	9 дней	Перейти на платную версию
	cloud-andrey-borue	b1g8dutu8l67ue04cn6f	
	default	b1g8kam2npjj1e7ndoeo	
	netology-demo-01 netology-demo-01	b1glpdkdn62l0ifequa2	
	default	b1gibf75jkk8aj4tdmi0	

Блог

- Новые возможности Биллинга, доступные всем пользователям Yandex.Cloud
24 сентября ❤️ 2
- Новые решения в области безопасности Yandex.Cloud
24 сентября ❤️ 4
- Больше возможностей для визуализации и анализа данных: главные апдейты Yandex DataLens
24 сентября ❤️ 3

Документация

- [Иерархия ресурсов в Yandex.Cloud](#)
- [Добавить пользователя](#)
- [Зоны доступности](#)
- [Создать виртуальную машину](#)
- [Интерфейс командной строки](#)

Мобильное приложение

Управляйте ресурсами, следите

Регионы и зоны доступности AWS

AWS охватывает 77 зон доступности в 24 географических регионах по всему миру.



Установка cli клиентов

- При помощи менеджера пакетов apt, brew, ...
- AWS: Скачать исходники <https://aws.amazon.com/cli/>
- Yandex: <https://cloud.yandex.ru/docs/cli/quickstart>



VPC (Virtual Private Cloud)

Это логически изолированный раздел облака, в котором можно запускать ресурсы в самостоятельно заданной виртуальной сети. Таким образом можно полностью контролировать среду виртуальной сети, в том числе выбирать собственный диапазон IP-адресов, создавать подсети, а также настраивать таблицы маршрутизации и сетевые шлюзы.

AWS Identity and Access Management (IAM)

IAM – это место где происходит управление учетными записями пользователей и их правами.

- Создаем отдельного пользователя для дальнейшей работы
- Нужно получить:
 - идентификатор ключа доступа: Access Key ID,
 - секретный ключ доступа: Secret Access Key



Yandex.Cloud IAM для Terraform

Инструкция для получения токена:

<https://cloud.yandex.ru/docs/iam/operations/iam-token/create>

Политика (policy) IAM

Политика IAM – это документ в формате JSON, который определяет, что пользователю позволено, а что — нет.

Назначим нашему пользователю:

- AmazonEC2FullAccess
- AmazonS3FullAccess
- AmazonDynamoDBFullAccess
- AmazonRDSFullAccess
- CloudWatchFullAccess
- IAMFullAccess

Регистрируем этого пользователя локально

Чтобы консольный клиент AWS и Terraform получили доступ к нашему аккаунту создаем переменные окружения:

```
$ export AWS_ACCESS_KEY_ID=(your access key id)
$ export AWS_SECRET_ACCESS_KEY=(your secret access key)
```



Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2)

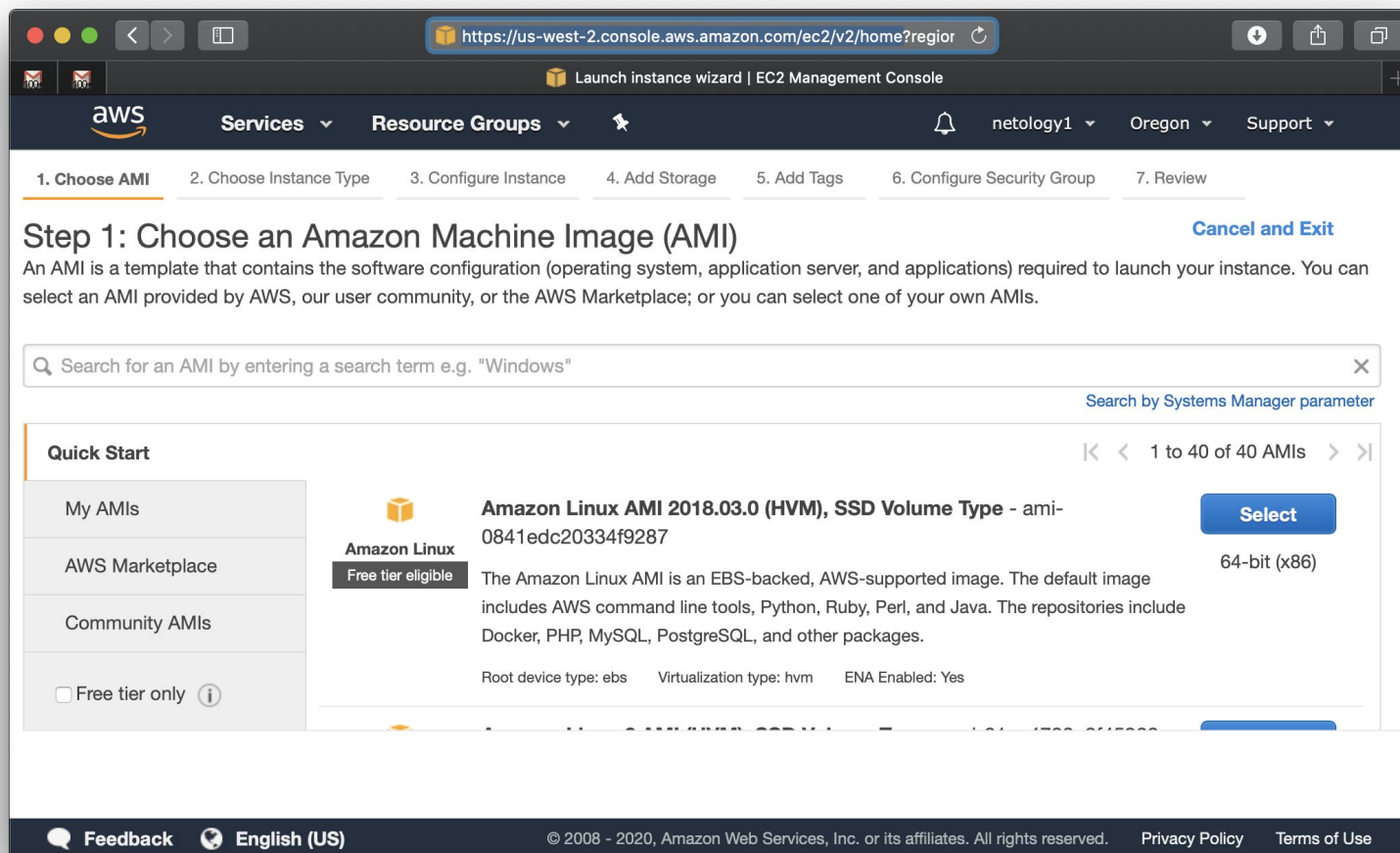
Это веб-сервис, предоставляющий безопасные масштабируемые вычислительные ресурсы в облаке.

Позволяет выбрать:

- тип и количество ядер процессора,
- объем оперативной памяти,
- хранилища,
- акселераторы,
- и другое.

Создание EC2 через веб интерфейс

<https://us-west-2.console.aws.amazon.com/ec2/v2/home>



Создание EC2 через консоль

<https://awscli.amazonaws.com/v2/documentation/api/latest/reference/opsworks/create-instance.html>

```
aws ec2 create-instance
[--source-dest-check | --no-source-dest-check]
[--attribute <value>] [--block-device-mappings <value>]
[--disable-api-termination | --no-disable-api-termination]
[--dry-run | --no-dry-run] [--ebs-optimized | --no-ebs-optimized]
[--ena-support | --no-ena-support] [--groups <value>]
--instance-id <value> [--instance-initiated-shutdown-behavior <value>]
[--instance-type <value>] [--kernel <value>]
[--ramdisk <value>] [--sriov-net-support <value>] [--user-data <value>]
[--value <value>] [--cli-input-json | --cli-input-yaml]
[--generate-cli-skeleton <value>] [--cli-auto-prompt <value>]
```

Основные параметры EC2

Что нужно знать для создания инстанса:

- тип (процессор, память),
- идентификатор виртуального приватного облака,
- способ автоскейлинга,
- операционная система,
- идентификатор образа (ami),
- ключ доступа по ssh,
- зона доступности,
- идентификатор подсети,
- тип подключенных хранилищ,
- ... и еще десяток параметров.

А теперь нужно изменить инстанс

- Иногда необходимо предварительно остановить инстанс
- Иногда пересоздать
- Хорошо бы понять, что конкретно будет изменено
- Часто надо привести инстанс в исходное состояние после ручных правок

Как это сделать?

- Зайти в веб интерфейс и проверять все параметры?
- Через консоль выполнить:
 - describe,
 - сравнить с целевыми (исходными) значениями,
 - modify.
- Хорошо бы понять, что конкретно будет изменено (типа git diff)
- Часто надо привести инстанс в исходное состояние после ручных правок

Другими словами...

Надо воспользоваться командами:

- `aws ec2 create-key-pair`
- `aws ec2 create-instance`
- `aws ec2 create-tags`
- `aws ec2 create-volume`
- `aws ec2 describe-key-pair`
- `aws ec2 describe-instances`
- `aws ec2 describe-tags`
- `aws ec2 describe-volume`
-



Синтаксис Terraform



Terraform

Terraform – это просто API-клиент.

Terraform-провайдеры «знают» все эти команды и умеют приводить состояние ресурсов к описанному в своих конфигурационных файлах.

Провайдеры могут работать как с любым клиентом:

cli, http, их комбинациями и другими.



Terraform-провайдеры

terraform.io/docs/providers/index.html

В официальном репозитории около 150 штук.

Плюс много неофициальных, и можно достаточно просто создавать собственные.

Блоки

Все конфигурации описываются в виде блоков.

```
resource "aws_vpc" "main" {  
  cidr_block = var.base_cidr_block  
}  
  
тип "идентификатор" "имя" {  
  название_параметра = выражение_значение_параметра  
}
```

Блок провайдеров

registry.terraform.io/providers/hashicorp/aws/latest/docs

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

Блок требований к провайдерам

Блок «terraform» для указаний версий провайдеров и бэкэндов.

```
terraform {  
  required_providers {  
    aws = {  
      source = "hashicorp/aws"  
      version = "~> 3.0"  
    }  
  }  
}
```


Блок ресурсов

registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/instance

Ресурс **aws_instance** – это экземпляр ec2

```
resource "aws_instance" "web" {  
    ami = data.aws_ami.ubuntu.id  
    instance_type = "t3.micro"  
}
```

Блок внешних данных

Для того что бы прочитать данные из внешнего API и использовать для создания других ресурсов.

registry.terraform.io/providers/hashicorp/aws/latest/docs/data-sources/caller_identity

```
data "aws_caller_identity" "current" {}

// data.aws_caller_identity.current.account_id
// data.aws_caller_identity.current.arn
// data.aws_caller_identity.current.user_id
```

Блок переменных

Каждый модуль может зависеть от переменных.

```
variable "image_id" {  
    type = string  
}  
  
resource "aws_instance" "example" {  
    instance_type = "t2.micro"  
    ami           = var.image_id  
}
```

Блок переменных

Структура переменной может быть достаточно сложной.

```
variable "availability_zone_names" {
  type    = list(string)
  default = ["us-west-1a"]
}

variable "docker_ports" {
  type = list(object({
    internal = number
    external = number
    protocol = string
  }))
  default = [
    {
      internal = 8300
      external = 8300
      protocol = "tcp"
    }
  ]
}
```

Типы переменных

Примитивные типы:

- string
- number
- bool

Комбинированные типы:

- list(<TYPE>)
- set(<TYPE>)
- map(<TYPE>)
- object({<ATTR NAME> = <TYPE>, ... })
- tuple([<TYPE>, ...])

Валидация переменных

Особенно важно для повторно используемых модулей.

```
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-\"."  
    }  
}
```

Блок output

Для того чтобы разные модули могли использовать результат работы друг-друга.

```
output "instance_ip_addr" {  
    value          = aws_instance.server.private_ip  
    description = "The private IP address of the main server  
instance."  
  
    depends_on = [  
        # Security group rule должна быть создана перед тем как  
можно будет использовать этот ip адрес, иначе сервис будет  
недоступен  
        aws_security_group_rule.local_access,  
    ]  
}
```

Локальные переменные

Могут быть использованы внутри модуля сколько угодно раз.

```
locals {  
  service_name = "forum"  
  owner        = "Community Team"  
}  
locals {  
  instance_ids = concat(  
    aws_instance.blue.*.id, aws_instance.green.*.id  
  )  
  common_tags = {  
    Service = local.service_name  
    Owner   = local.owner  
  }  
}
```

Комментарии

Terraform поддерживает несколько видов комментариев:

- `#` начинает однострочные комментарии;
- `//` также однострочные комментарии;
- `/*` и `*/` для обозначения многострочных комментариев.



Структура проекта

Структура каталогов

- `/main.tf`
- `/any_file.tf`
- `/modules/`
- `/modules/awesome_module/`
- `/modules/awesome_module/main.tf`
- `/modules/awesome_module/any_other_file.tf`
- `/modules/next_module/`
- `/modules/next_module/main.tf`
- `/modules/next_module/any_other_file.tf`

Структура файлов

- main.tf
- variables.tf
- outputs.tf
- any_other_files.tf



Итоги

Итоги

Сегодня мы:

- Познакомились с облачными провайдерами AWS и Yandex.Cloud\$
- Познакомились с базовым синтаксисом Terraform;
- Узнали, как создавать виртуальный инстанс ec2:
 - через веб интерфейс,
 - через cli консоль,
 - при помощи Terraform.



Домашнее задание

Домашнее задание

Давайте посмотрим ваше [домашнее задание](#).

- Вопросы по домашней работе задавайте **в чате** мессенджера Slack.
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

**Задавайте вопросы и
пишите отзыв о лекции!**

Андрей Борю



[andreyborue](https://t.me/andreyborue)



[andreyborue](https://netology.ru/andreyborue)



[andreyborue](https://t.me/andreyborue)