нетология

Облачные провайдеры и синтаксис Терраформ

Сергей
Андрюнин

Сергей Андрюнин

DevOps-инженер

RTLabs

Сергей Андрюнин

План занятия

1. Amazon Web Service
2. Amazon EC2
3. Синтаксис терраформ
4. Структура проекта
5. Итоги
6. Домашнее задание

AWS (Amazon Web Services)

AWS

- Достаточно популярное решение.
- Очень большое количество сервисов.
- В первый год использования есть бесплатный тариф: https://aws.amazon.com/free/.

Нужно ли создавать аккаунт?

Все примеры из ближайших занятий могут быть разобраны без aws аккаунта.

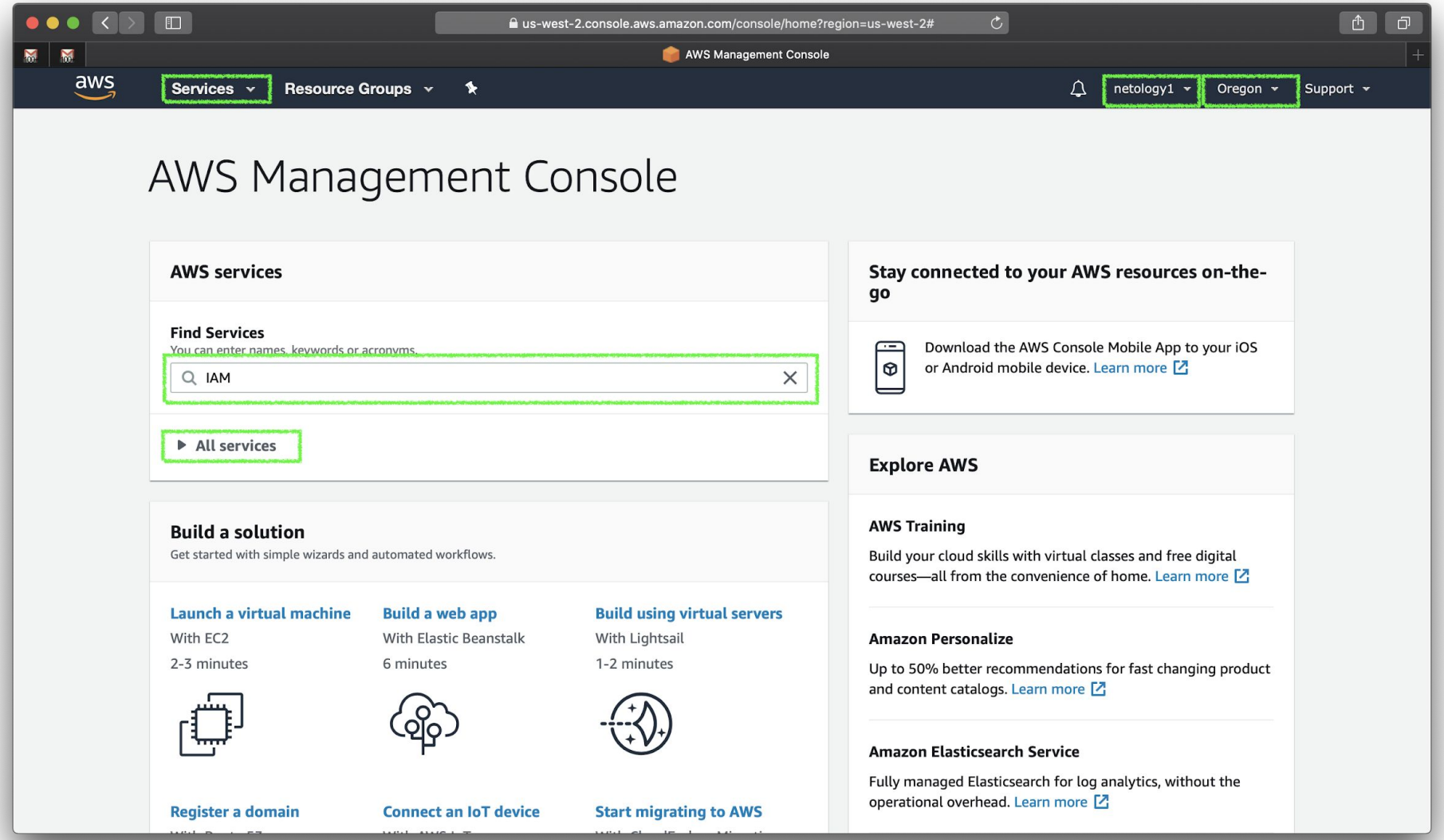
План действий

- Рассмотрим работу с ресурсами aws 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” аккаунта на email типа “yourname**+netology**@gmail.com”

Элементы управления



Регионы и зоны доступности

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

Регионы и зоны доступности

Далее будем использовать регион **us-west-2**.

Установка aws-cli

- При помощи менеджера пакетов apt, brew, ...
- Скачать исходники https://aws.amazon.com/cli/.

Amazon VPC (Virtual Private Cloud)

Это логически изолированный раздел облака AWS, в котором можно запускать ресурс AWS в собственной виртуальной сети. Можно полностью контролировать среду виртуальной сети, включая выбор собственного диапазона IP-адресов, создавать подсети, а также настраивать таблицы маршрутизаций и сетевые шлюзы.

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.

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 Elastic Compute Cloud (Amazon EC2)

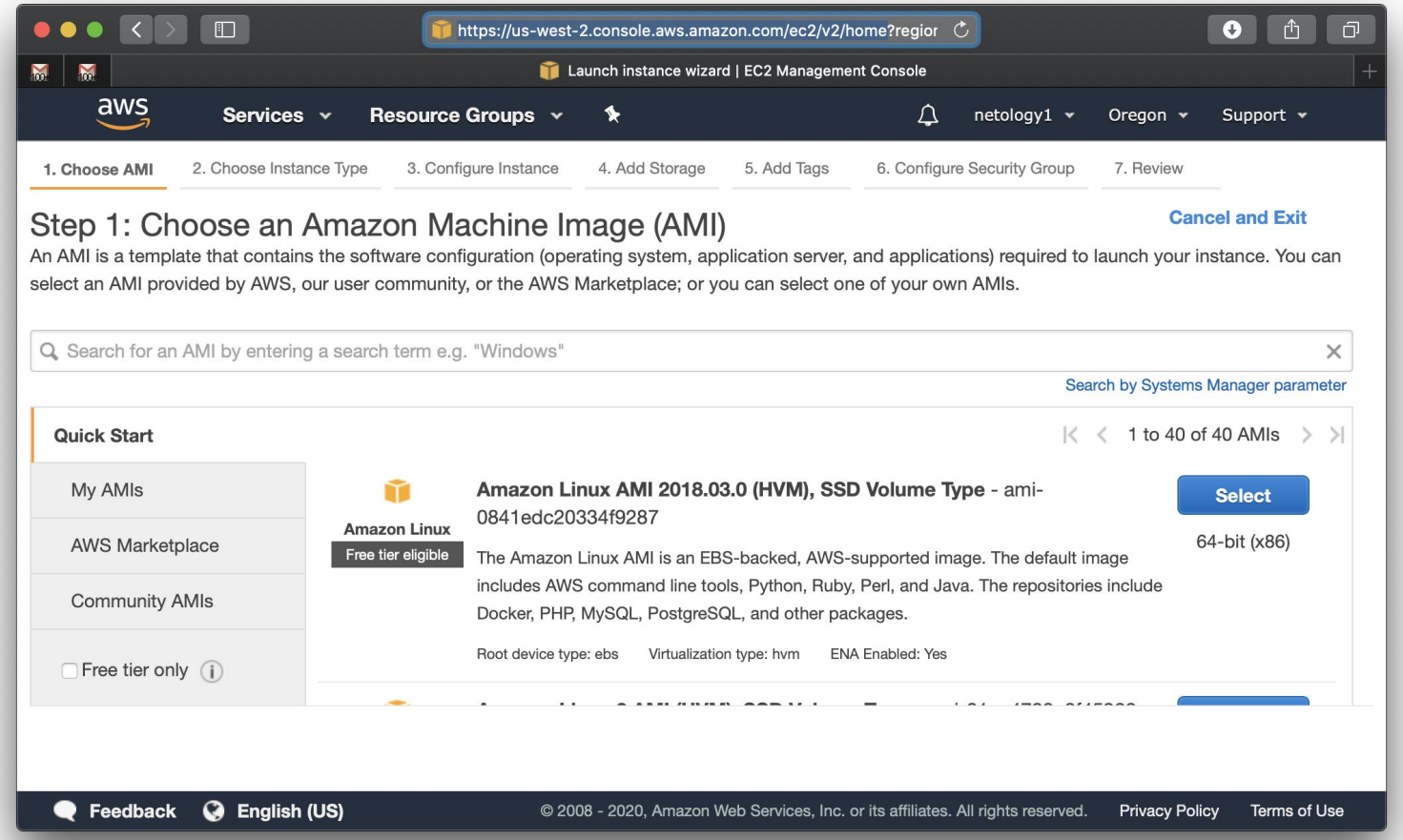
Это веб-сервис, предоставляющий безопасные масштабируемые вычислительные ресурсы в облаке.

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
- ....

Терраформ

Терраформ - это просто API-клиент.

Терраформ-провайдер знает все эти команды и умеет приводить состояние ресурсов к указанному в своих конфигурационных файлах.

Они могут работать с любым клиентом: cli, http, их комбинациями и другими.

Терраформ 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

Терраформ поддерживает несколько видов комментариев:

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

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

- Познакомились с облачным провайдером AWS.
- Познакомились с базовым синтаксисом терраформа.
- Узнали как создавать виртуальный инстанс ec2:
  - через веб интерфейс,
  - через cli консоль,
  - при помощи терраформ.

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**.

нетология

Задавайте вопросы и пишите отзыв о лекции!

Сергей Андрюнин

Сергей Андрюнин