НЕТОЛОГИЯ
групп

# AMAZON WEB SERVICES
# ДЛЯ МАШИННОГО ОБУЧЕНИЯ

Глеб Ивашкевич, независимый разработчик

# Глеб Ивашкевич

независимый разработчик

почта

сообщения

# УЗНАЕМ

- что такое Amazon Web Services;
- как его применять для задач машинного обучения;
- как запускать и настраивать серверы на AWS;
- как пользоваться готовыми сервисами AWS для машинного обучения.

ПЛАН ЗАНЯТИЯ

1. Как устроен AWS;
2. Небольшие задачи: работаем с одним сервером;
3. Сложные задачи: работаем с AWS CLI;
4. Готовые сервисы AWS для машинного обучения.

# Машинное обучение

- **данные**:
  - ➜ хранение
  - ➜ быстрый доступ

# Машинное обучение

- **данные**:
  - ➔ хранение
  - ➔ быстрый доступ
- **алгоритмы**:
  - ➔ ресурсоемкие
  - ➔ специальное железо

# Машинное обучение

- **данные**:
  - ➔ хранение
  - ➔ быстрый доступ
- **алгоритмы**:
  - ➔ ресурсоемкие
  - ➔ специальное железо

нам поможет

**облачная инфраструктура**

# Облачная инфраструктура

- любые вычислительные ресурсы доступны;
- нет upfront затрат;
- легко масштабируется;
- гибкость.

# Amazon Web Services

- облачная инфраструктура Amazon;
- запущена в 2002, перезапущена в 2006;
- managed;
- набор сервисов для (почти) любых задач;
- pay-as-you-go.

# Amazon Web Services

- вычисления:

  **EC2**(Elastic Compute Cloud), **Lambda**

- хранение и БД:

  **S3**(Simple Storage Service), **EFS**(Elastic File System), **RDS**(Relational Database Service)

- машинное обучение:

  **Amazon Rekognition**, **Amazon Comprehend**

- многое другое

History

Console Home

EC2

Cost Explorer

Billing

VPC

Machine Learning

Find a service by name or feature (for example, EC2, S3 or VM, storage).

Group A-Z

## Compute
EC2
Lightsail
Elastic Container Service
Lambda
Batch
Elastic Beanstalk

## Storage
S3
EFS
Glacier
Storage Gateway

## Database
Relational Database Service
DynamoDB
ElastiCache
Amazon Redshift

## Migration
AWS Migration Hub
Application Discovery Service
Database Migration Service
Server Migration Service
Snowball

## Developer Tools
CodeStar
CodeCommit
CodeBuild
CodeDeploy
CodePipeline
Cloud9
X-Ray

## Management Tools
CloudWatch
AWS Auto Scaling
CloudFormation
CloudTrail
Config
OpsWorks
Service Catalog
Systems Manager
Trusted Advisor
Managed Services

## Media Services
Elastic Transcoder
Kinesis Video Streams
MediaConvert
MediaLive
MediaPackage

## Machine Learning
Amazon SageMaker
Amazon Comprehend
AWS DeepLens
Amazon Lex
Machine Learning
Amazon Polly
Rekognition
Amazon Transcribe
Amazon Translate

## Analytics
Athena
EMR
CloudSearch
Elasticsearch Service
Kinesis
QuickSight
Data Pipeline
AWS Glue

## Security, Identity & Compliance
IAM
Cognito
GuardDuty
Inspector
Amazon Macie

## AR & VR
Amazon Sumerian

## Application Integration
Step Functions
Amazon MQ
Simple Notification Service
Simple Queue Service
SWF

## Customer Engagement
Amazon Connect
Pinpoint
Simple Email Service

## Business Productivity
Alexa for Business
Amazon Chime
WorkDocs
WorkMail

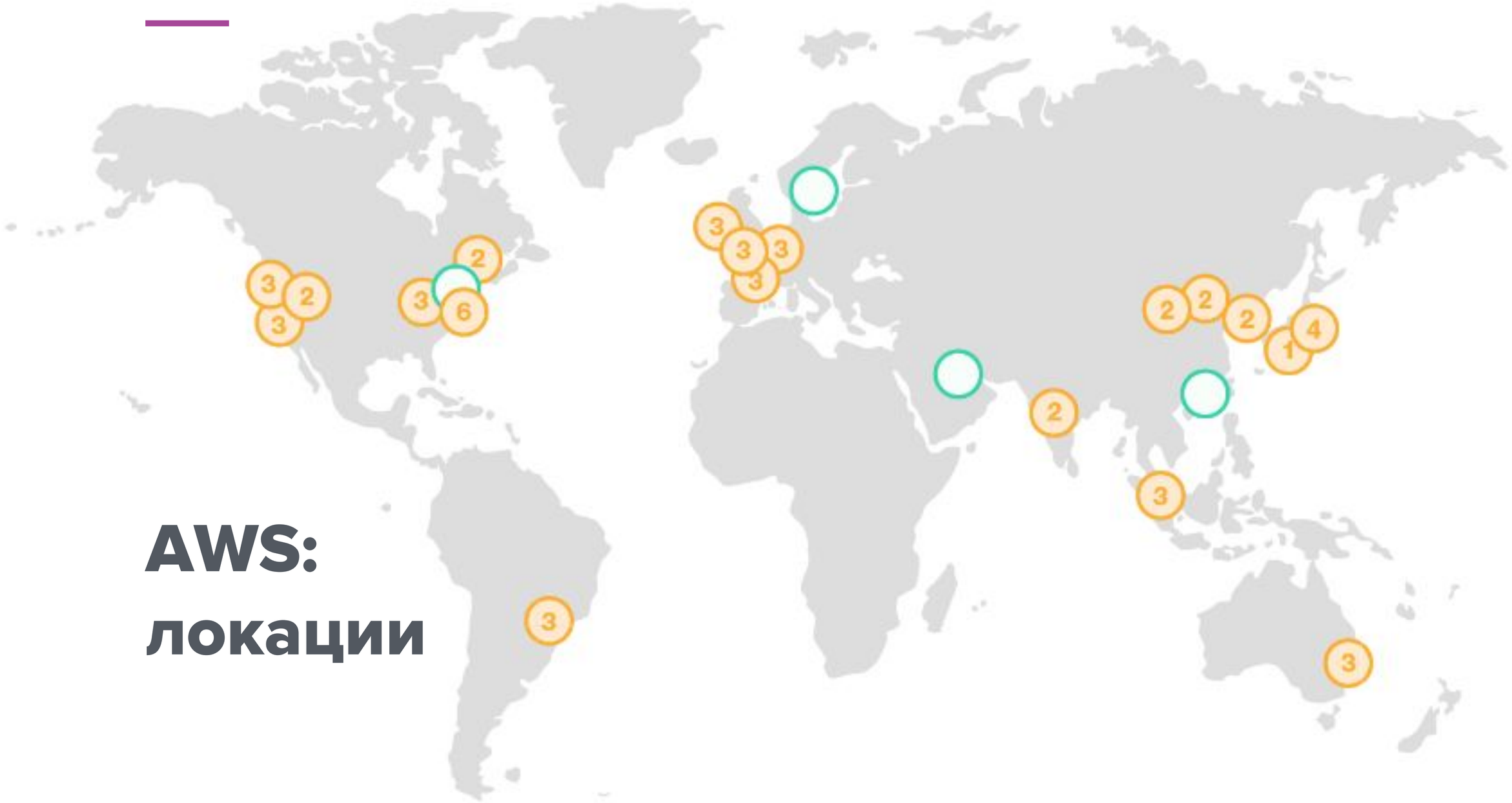
## Desktop & App Streaming
WorkSpaces
AppStream 2.0

# AWS используют

- крупный бизнес: **Siemens**, **General Electric**
- сервисы: **Airbnb**, **Netflix**, **Spotify**
- правительства: **NASA, ESA**

# для

- интернет-сервисов
- HPC(High Performance Computing) и анализа данных
- резервного копирования и др.




# AWS: локации

## production


## R&D

# что будем изучать мы

Часть 1

# **Начнем с простого:** один сервер в AWS

# Что сделаем?

- запустим сервер (**EC2 instance**);
- настроим доступ (**security groups**);
- создадим блочное хранилище (**EBS**);
- установим и настроим ПО (**`Jupyter`**);
- создадим образ виртуальной машины (**AMI**);
- создадим ключи доступа для AWS CLI и `boto`.

# Для каких задач?

- индивидуальное использование;
- небольшие проекты по машинному обучению (размер данных до 1Тб);
- соревнования на Kaggle.

# **Запускаем EC2 сервер** (в консоли AWS)

- создаем ssh-ключ (NETWORK AND SECURITY → Key Pairs → Create Key Pair)
- выбираем образ (INSTANCES → Instances → Launch Instance)
- настраиваем запуск (INSTANCES → Instances → Launch Instance)
- подключаемся по ssh (INSTANCES → Instances → Launch Instance)

# **Настраиваем EC2 сервер** (по ssh)

- устанавливаем пакеты Python;
- запускаем Jupyter;
- проверяем;
- создаем конфигурацию Jupyter:
  - ➜ доступ по https с паролем
  - ➜ настройка Jupyter

# Настраиваем Jupyter

- создаем пароль;
- создаем файл конфигурации;
- базовые параметры;
- доступ по https;
- директория с ноутбуками;
- *добавляем systemd сервис для Jupyter*.

# Создаем EBS диск

- в консоли AWS создаем диск на 100Gb;
- подключаем диск к нашему инстансу;
- создаем файловую систему;
- монтируем диск;
- создаем запись в /etc/fstab;
- создаем образ инстанса.

# Финальный штрих

- создаем ключи доступа;
- понадобятся для `AWS CLI` и `boto`.
- My Security Credentials → Access keys (access key ID and secret access key) → Create New Access Key.

# Что у нас теперь есть

- **образ машины** — легко запускать;
- **диск** — не нужно каждый раз копировать данные;
- **базовые настройки**: security group, ключ ssh и ключи доступа к AWS.

# Что можно было бы добавить

- **установка CUDA** — для `tensorflow`;
- **упростить запуск** с помощью `AWS CLI`.

Часть 2

# Усложняем:

конфигурации посложнее и немного автоматизации

# Что не так с нашим сервером?

- непонятно, как копировать данные на инстанс;
- запускаем вручную;
- а стоит ли даже с https и паролем светить Jupyter всему миру?
- данные на EBS диске доступны только одному инстансу.

# Копируем данные

- проще всего с scp;
- на инстанс:

```
$ scp -i <ssh_key_pem_file> <local_file>\
ubuntu@<instance_public_ip>:<location>
```

- с инстанса:

```
$ scp -i <ssh_key_pem_file> ubuntu@<instance_public_ip>:<location>\
<local_file>
```

# Запускаем с AWS CLI

- нужно установить `awscli`:

```
$ sudo pip install awscli
```

- запускаем:

```
$ aws configure
$ aws ec2 run-instances\
    --count 1 --image-id <ami-id>\
    --instance-type t2.micro\
    --key-name <key_name>\
    --security-group-ids <security_group_id>\
    --subnet-id <subnet_id>
```

# Запускаем Jupyter еще безопасней

- port forwarding для ssh:

```
$ ssh -i <ssh_key_pem_file> -N -L
<local_port>:localhost:<remote_port> ubuntu@<instance_public_ip>
```

- можно убрать пароль и https.

# **Объектное хранилище S3** (Simple Storage Service)

- масштабируемое;
- надежное;
- поддерживает шифрование;
- интегрировано с другими сервисами AWS;
- объекты хранятся в **бакетах (buckets)**;
- возможно версионирование.

# Объектное хранилище S3 (Simple Storage Service)

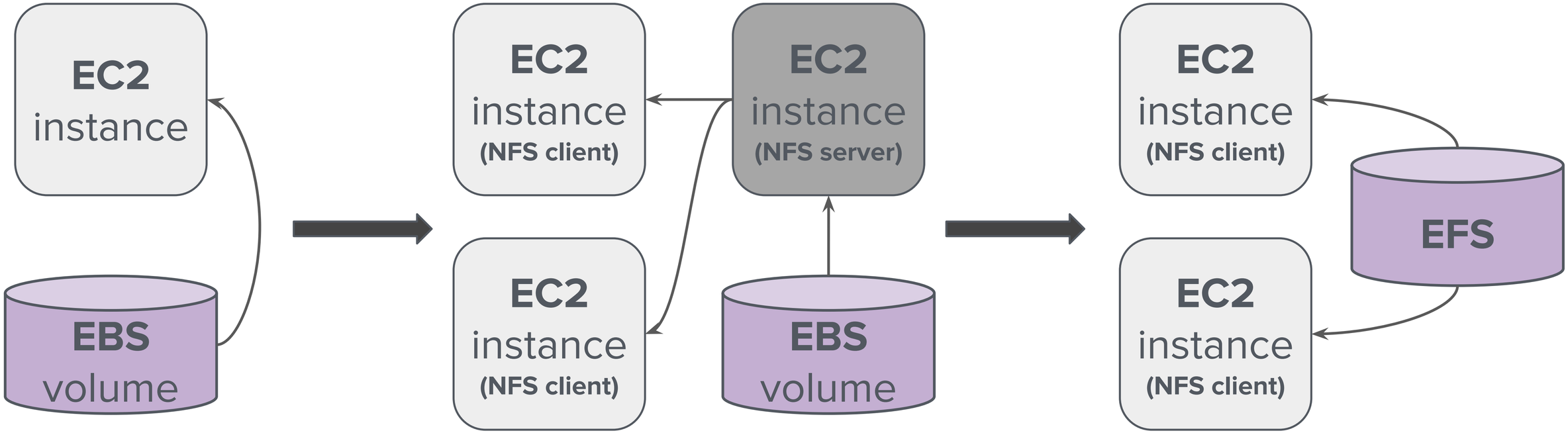
- создадим бакет с помощью консоли AWS;
- и с помощью AWS CLI:

```
$ aws s3 mb s3://<bucket_name>
$ aws s3 cp <filename> s3://<bucket_name>
$ aws s3 ls s3://<bucket_name>
```

# Другие сервисы для хранения данных

- **RDS** — Relational Database Service;
- **EFS** — Elastic File System (расширяемая NFS);
- **Glacier** — для редко используемых объектов;
- другие: **Aurora**, **RedShift**, etc

# Хранение данных: NFS vs EFS




**один сервер**

**много серверов + NFS**

**много серверов + EFS**

# Что у нас теперь есть

- пользуемся AWS CLI;
- знаем, что есть разные возможности для хранения данных;
- знаем, как работать с S3.

# Что можно было бы добавить

- **развернуть** `tensorflow` и `tensorboard`;
- использовать **`boto`** вместо AWS CLI (удобнее автоматизировать).

Часть 3

# Упрощаем:

готовые сервисы для машинного обучения

# Делать ли все вручную?

- многие реальные задачи можно решить, комбинируя более простые решения;
- многие задачи уже решены, незачем изобретать велосипед: face recognition, text recognition, etc;
- трудозатраты отличаются не в разы, а на порядки.

# Сервисы AWS для ML

- **Rekognition**: анализ изображений и видео;
- **Comprehend**: анализ текста;
- **Transcribe**: распознавание речи;
- **SageMaker** и другие: упрощают работу data scientist’a.

# AWS Rekognition

- доступен по простому API;
- powered by deep learning ©;
- умеет: распознавать объекты, людей, текст и др.;
- может работать с изображениями и видео.

# AWS Rekognition

- может работать с напрямую с изображениями или с изображениями, хранящимися на S3;
- попробуем:

```
$ aws s3 cp <filename.jpg> s3://<bucket_name>
$ aws rekognition detect-faces\
--image "S3Object={Bucket=<bucket_name>,Name=<image_name>}"
```

- **попробовать самостоятельно:** то же самое с boto, визуализировать распознанные landmarks.

# AWS Comprehend

- распознавание тональности, языка, entities;
- попробуем:

```
$ aws comprehend detect-entities --text "Some funny English text
about AWS, deep learning and tensorflow." --language-code en
```

# Что у нас теперь есть

- знаем, какие есть готовые сервисы;
- не будем мастерить велосипед *(ладно, будем, но только для самообразования!😼)*.

# Что дальше?

- исследовать;
- учиться строить инфраструктуру для своих задач из готовых блоков;
- стараться минимизировать затраты на решение задач, которые уже решены.

# Что почитать?

- **документация** - наш друг;
- **Amazon Web Services in Action** by Michael Wittig and Andreas Wittig;
- **экспериментировать**: например, создайте чат-бота, который может распознавать лица или текст, с EC2, S3, Lambda и Recognition (think big, start small).

НЕТОЛОГИЯ
групп

# Спасибо за внимание!

ГЛЕБ ИВАШКЕВИЧ