8.4

Создаем стенд

Текущий стенд состоит из 1 узла: `node-1.sXXXXXX` (XXXXXX - ваш номер студента).
Работа стенда 6 часов. 2 попытки запуска стенда.

Что уже установлено на стенде?

- node-1: ansible2.9, python2.7\3.6, molecule, buildah, docker sdk, docker

Какой IP-адрес у них?

1 и 4 октет сети у всех одинаковый -**172.XX.XXX.6** (node-1). 2 и 3 узнать через команду `ip -a`. Например, это **20.247**

```
ip a | grep eth0

inet 172.20.247.6/24 brd 172.20.247.255 scope global eth0
```

1. Над текстом нажмите кнопку «Создать стенд». Каждый стенд создан под определенную тему курса. Сейчас вы находитесь в теме **8.Облака, Ansible и все все все**, этот стенд не подойдет под практические занятия из других пунктов курса.

Запуск обычно идёт до 10 минут, в редких случаях до 30 минут.

2. После создания стенда авторизуйтесь по SSH на adminbox с адресом **sbox.slurm.io** с помощью логина и пароля, находящихся в настройках профиля или над текстом по кнопке «**Доступы»**.

3. Потом подключаетесь к первой ноде — контролнода. (XXXXXX - ваш номер студента)

```
ssh node-1.sXXXXXX
```

И повышаете себя до root пользователя.

```
sudo -i
```

Можете приступать к выполнению практических заданий. Желаем успешно выполнить их!

Buildah

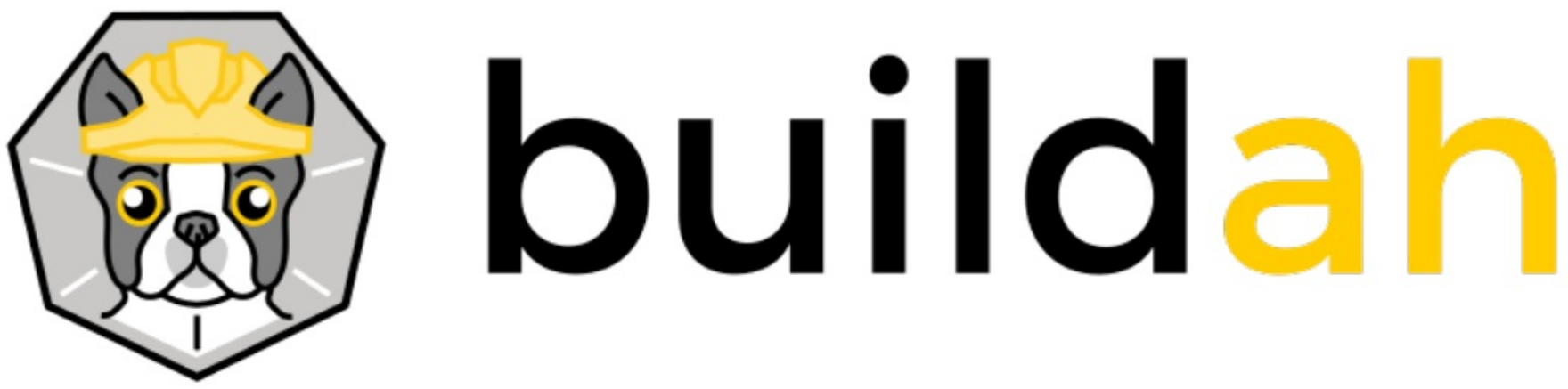
В предыдущих лекциях мы рассмотрели работу с Docker в Ansible. Одной из частей запуска программ внутри Docker-контейнеров является построение так

называемого image -> образа с необходимым набором библиотек который и будет являться окружением для нашего проекта.

Стандартным способом для настройки Docker image является написание Dockerfile, содержащего в себе команды, подготавливающие окружение, а затем сборки этого контейнера с помощью самой системы Docker. Это отличный, простой и быстрый в освоении подход. НО!

Он требует полной установки Docker SDK, запуска демонов и привязывает вас к экосистеме самого Docker. А хочется какой-нибудь легковесной штуки, которая просто соберет имейдж и отправит его дальше.

Одним из таких инструментов является Buildah.



Buildah позволяет делать докер контейнеры с помощью скриптов, не используя докерфайлы.

Выглядит это как то так

```
#!/usr/bin/env bash -x

ctr1=$(buildah from "${1:-fedora}")

## Get all updates and install our minimal httpd server
buildah run "$ctr1" -- dnf update -y
buildah run "$ctr1" -- dnf install -y lighttpd

## Include some buildtime annotations
buildah config --annotation "com.example.build.host=$(uname -n)" "$ctr1"

## Run our server and expose the port
buildah config --cmd "/usr/sbin/lighttpd -D -f /etc/lighttpd/lighttpd.conf" "$ctr1"
buildah config --port 80 "$ctr1"

## Commit this container to an image name
buildah commit "$ctr1" "${2:-$USER/lighttpd}"
EOF
```

Как мы можем видеть, это последовательный консольный запуск команд в bash скрипте. Я думаю, здесь уже можно увидеть проблему buildah — она весьма низкоуровневая, позволяя работать только скриптами, не используя никаких высокоуровневых и удобных решений.

Таких, как, например, Ansible.

Почему Ansible?

Потому что Ansible обладает четкой структурой, огромным количеством модулей, системой дебага и заточенностью именно под настройку софта и сборку чего угодно. Поэтому использовать его совместно с Buildah — великолепный путь построения своих контейнеров.

Внимание! *Если ваша инфраструктура уже запущена на докерфайлах, скорее всего, вам просто не нужно это все. В этом есть смысл, только если вы видите, что докерфайлы вам не подходят, либо хотите унифицировать весь пайплайн, используя Ansible.*

Для этого мы будем использовать ansible-bender.

Это утилита, написанная для автоматизации работы Ansible с Buildah, которая базово позволяет вам вообще не разбираться что такое Buildah и как он работает, а просто собирать image с помощью обычного плейбука. Соответственно точно так же можно использовать все ваши наработки, использовать роли и все что вы хотите.

Давайте установим ansible-bender и посмотрим как он работает на примере. Нам нужен будет Ansible на Python > 3.6, более ранние версии не поддерживаются.

Установим ansible-bender через pip

```
sudo pip3 install ansible-bender
```

Установим buildah

```
sudo yum -y install buildah
```

Установим podman - утилиту для управления самими имейджами, buildah только умеет билдить, а хранить и отображать, это уже к подману.

```
sudo yum -y install podman
```

Возможные проблемы.

На CentOS 7 ансибл из yum ставится под питон 2, а bender не работает с ним, ставьте ansible через pip3, тогда все будет хорошо.

И мы готовы строить наши образы.

Для построения нашего image, я дам простой пример

```
---
- name: Demonstration of ansible-bender functionality
  hosts: localhost
  vars:
    ansible_bender: # все переменные для контейнеров мы помещаем в переменную
ansible_bender
      base_image: python:3-alpine # укажем базовый docker образ
      working_container:
        volumes:
          - '{{ playbook_dir }}:/src' # на рабочий контейнер мы смонтируем
директорию, используя переменную playbook_dir
      target_image:
        name: a-very-nice-image  # определим имя полученного контейнера
        working_dir: /src #определим рабочую директорую для запуска скриптов
        labels:
          built-by: 'root' # метаданные
        environment:
          FILE_TO_PROCESS: README.md # env файлы
  tasks: # эти таски запустятся ВНУТРИ контейнера и он потом будет запечен как docker
image
  - name: Run a sample command
    command: 'ls -lha /src'
  - name: Stat a file
    stat:
      path: "{{ lookup('env','FILE_TO_PROCESS') }}" # здесь не происходит ничего
полезного, но вы спокойно можете расширить плейбук, что то установить, вызвать роли и
тд. Это все будет запущено внутри контейнера.
```

Давайте запустим этот файл и посмотрим что же у нас получилось.

```
ansible-bender build playbook.yml
```

Увидим вывод плейбука, практически как в Ansible.

```
[root@localhost vagrant]# [root@localhost vagrant]# ansible-bender build playbook.yml

PLAY [Demonstration of ansible-bender functionality] ***************************************

TASK [Gathering Facts] *********************************************************************
ok: [a-very-nice-image-20210812-124540593517-cont]

TASK [Run a sample command] ****************************************************************
changed: [a-very-nice-image-20210812-124540593517-cont]

TASK [Stat a file] *************************************************************************
ok: [a-very-nice-image-20210812-124540593517-cont]

PLAY RECAP *********************************************************************************
a-very-nice-image-20210812-124540593517-cont : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

Getting image source signatures
Copying blob sha256:bc276c40b172b1c5467277d36db5308a203a48262d5f278766cf083947d05466
Copying blob sha256:011860d140a221ba966d393f31239d64a4f0972b63506fbf87b1b5c0c1d1f7a0
Copying blob sha256:f4f3ab477a61220290b89eb825d830c34b80c2da5cc379eb716038934c6ea8ac
Copying blob sha256:102020cbb86c41b9f0520a3f9aab7c91a3bbbb496186e0e0f5147ca36bb5d26d
Copying blob sha256:f732974f2c436a68093b652eee351f2907257112d7cbaffc4440156dc0433466
Copying blob sha256:a1d7ddb44bc77c63635ebb0b51d996cb6e0f6fb35462d03158132334bf266d98
Copying config sha256:a76d2f3065f73252fb9ddf78b938f882a5558565bfcd47bf5a08f4d076d8f252
Writing manifest to image destination
Storing signatures
a76d2f3065f73252fb9ddf78b938f882a5558565bfcd47bf5a08f4d076d8f252
Image 'a-very-nice-image' was built successfully \o/
```

Помимо этого bender помогает нам увидеть списки контейнеров, с их статусами, что очень удобно для контроля билдов. Команда `ansible-bender list-builds`

```
  BUILD ID  IMAGE NAME                                          STATUS    DATE                        BUILD TIME
----------  -------------------------------------------------  --------  --------------------------  ------------
         1  a-very-nice-image-20210812-124414022892-failed     failed    2021-08-12 12:44:14.021090  5 seconds
         2  a-very-nice-image                                  done      2021-08-12 12:45:58.946788  13 seconds
```

Ну а команда `ansible-bender get-logs` выведет нам все логи сборки.

Таким образом у нас появляется очень мощный и удобный инструмент для сборки модулей на Ansible.

Все, что мы проделали, можно было бы и сделать с помощью команд Docker-модуля, но он требует установки Docker , что на сборочной машине не всегда хороший вариант.

8.8

Перед нами плейбук, создающий инстансы в AWS EC2, подобно тому, что вы видели в лекциях. В реальной жизни даже у опытных разработчиков бывают ошибки, не дающие плейбукам запуститься. Задача — проанализировать ошибки в данном плейбуке и указать на них.

```
---
- name: "Create a EC2 instance"
  hosts: localhost
  gather_facts: false
  tasks:

 - name: "Create an instance"
   amazon.aws.ec2:
     key_name: ansiblekey
     region: us-east-2a
     instance_type: t2.micro
     image: ami-0ba62214afa52bec7
     wait: yes
     group: launch-wizard-1
     count: 2
     vpc_subnet_id: subnet-5da52d36
     instance_tags:
       group: web

- name: Add new instance to host group
  add_host:
    hostname: "{{ item.public_ip }}"
    groupname: webs
  loop: "{{ ec2.instances }}"

- name: Wait for SSH to come up
  delegate_to: "{{ item.public_dns_name }}"
  wait_for_connection:
    delay: 60
    timeout: 320
```

```
    loop: "{{ ec2.instances }}"

- name: Terminate instances
  hosts: localhost
  gather_facts: false

 tasks:
 - name: Terminate instances that were previously launched
    amazon.aws.ec2:
      state: 'absent'
      region: us-east-2
      instance_ids: '{{ ec2.instance_ids }}'
      aws_access_key: AKIAXCQMX4YHR7SEWMPB
      aws_secret_key: $$$$$$$$$$$$$$$$$$$$$$$$$$
```

8.9

Ansible в облаках

Если у вас ещё нет аккаунта в AWS или не знаете, как настроить доступ к API AWS, то мы расскажем об этом в трёх шагах.

В пятой лекции мы решили большую задачу по настройке баз данных. Теперь нам важно научиться не только настраивать окружения сами по себе, но и уметь расширять уже однажды сделанные задачи, поэтому следующая задача будет включать в себя уже выполненное задание с усложнением.

Расширьте плейбуки из пятой задачи с учетом следующих условий:

1.
    1. Постгрес устанавливается в конфигурации master-slave на разные сервера;
    2. Metrics exporter устанавливается на каждый конкретно взятый сервер, на мастера и слейвы;
    3. Сервера необходимо динамически создавать в облаке AWS (машины EC2);
    4. Prometheus хостится на отдельной машине.

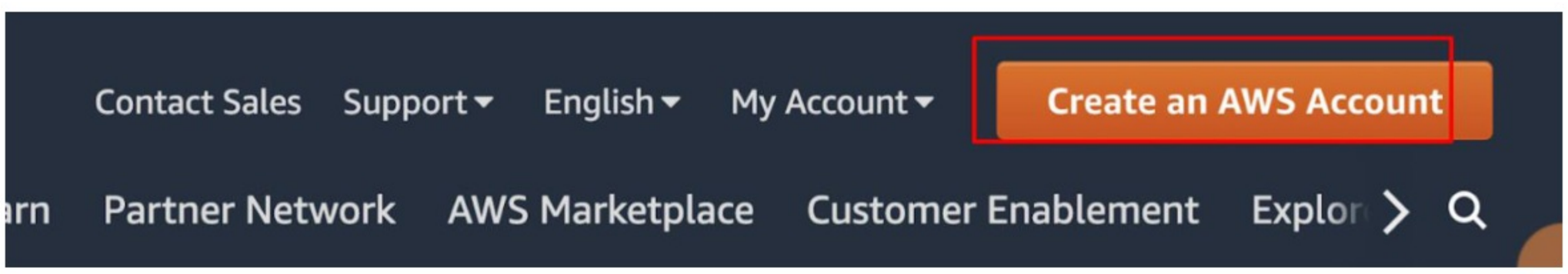
Вы можете создать ключи для машин заранее, а можете воспользоваться созданием ключей прямо из Ansible с помощью модуля

https://docs.ansible.com/ansible/latest/collections/amazon/aws/ec2_key_module.html

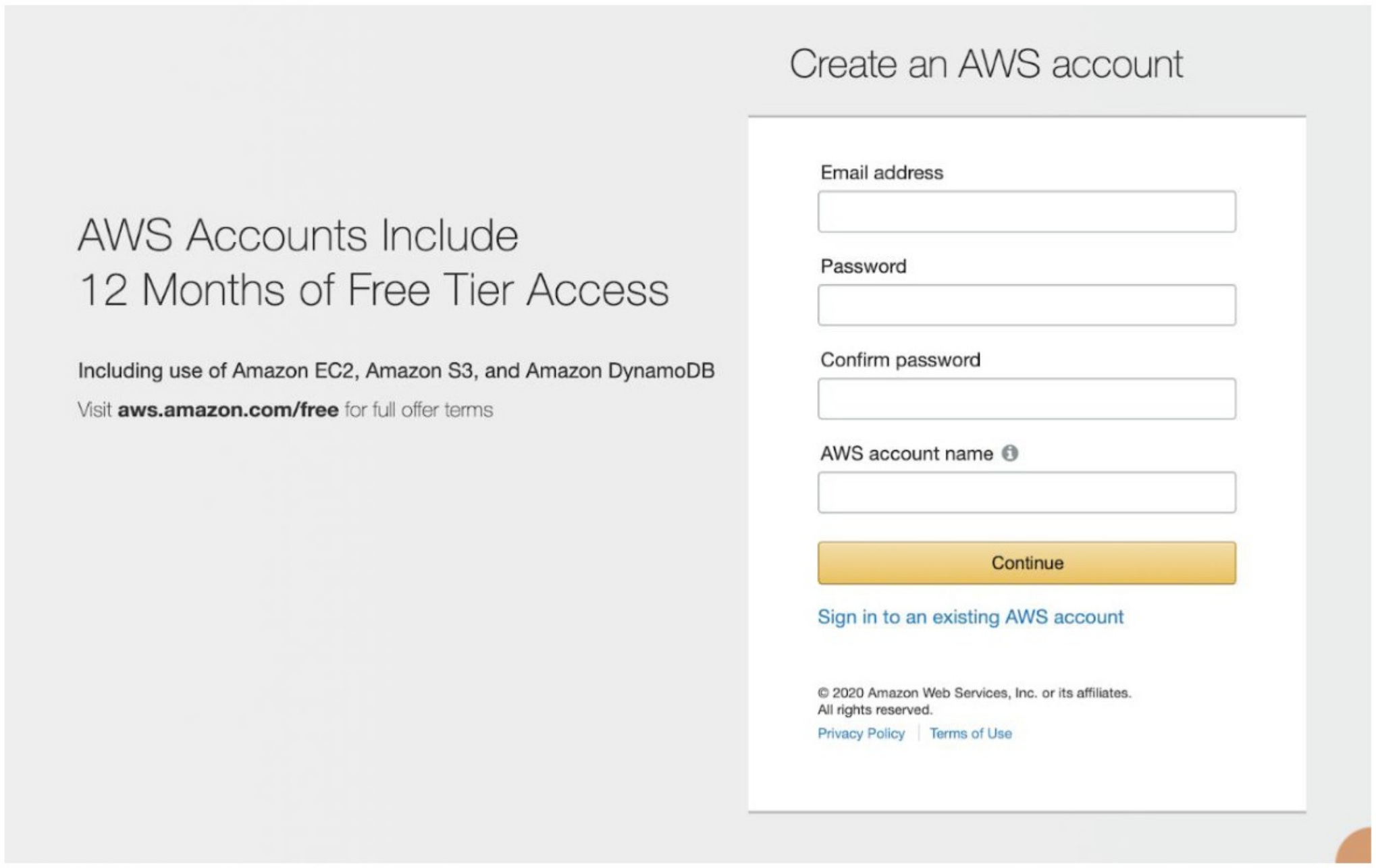
Модуль вернет строковое значение ключа, которое можно положить в переменную с помощью register и сохранить в файле.

Создание аккаунта

Первое, что нам понадобится — учётная запись AWS. Для начала регистрации перейдите на страницу aws.amazon.com и нажмите на кнопку «Create an AWS account».



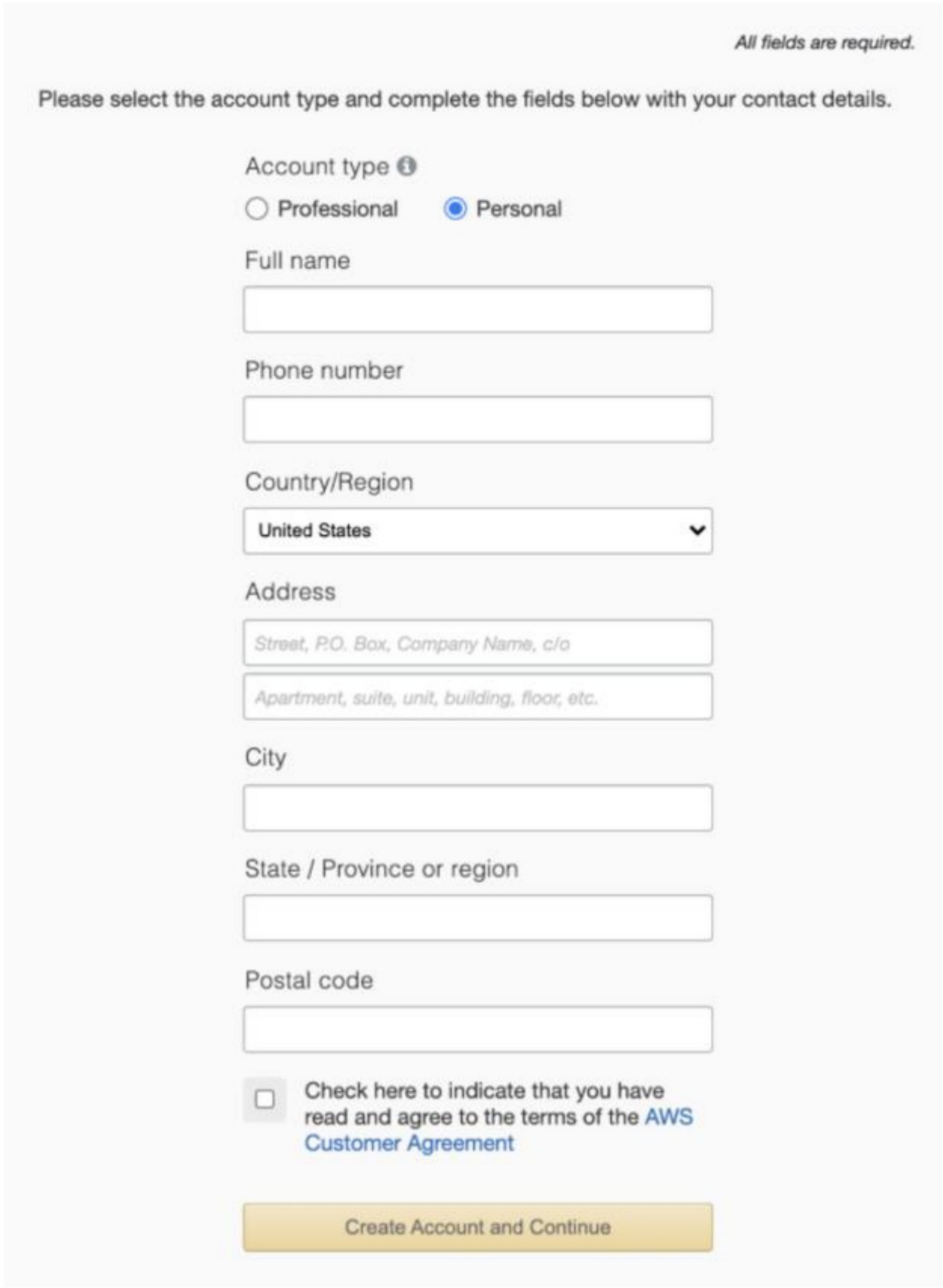
На этом этапе необходимо ввести e-mail, придумать пароль и название аккаунта. Пароль должен быть не короче 8 символов (включать специальные символы, цифры и символы разных регистров). Название аккаунта может быть любым, его можно будет поменять позднее.



На следующем экране необходимо выбрать тип аккаунта и заполнить остальные данные. Тип аккаунта может быть professional или personal. Если вы создаёте свой личный аккаунт, выбирайте personal — никакой практической разницы не будет, зато в personal форма регистрации короче :)

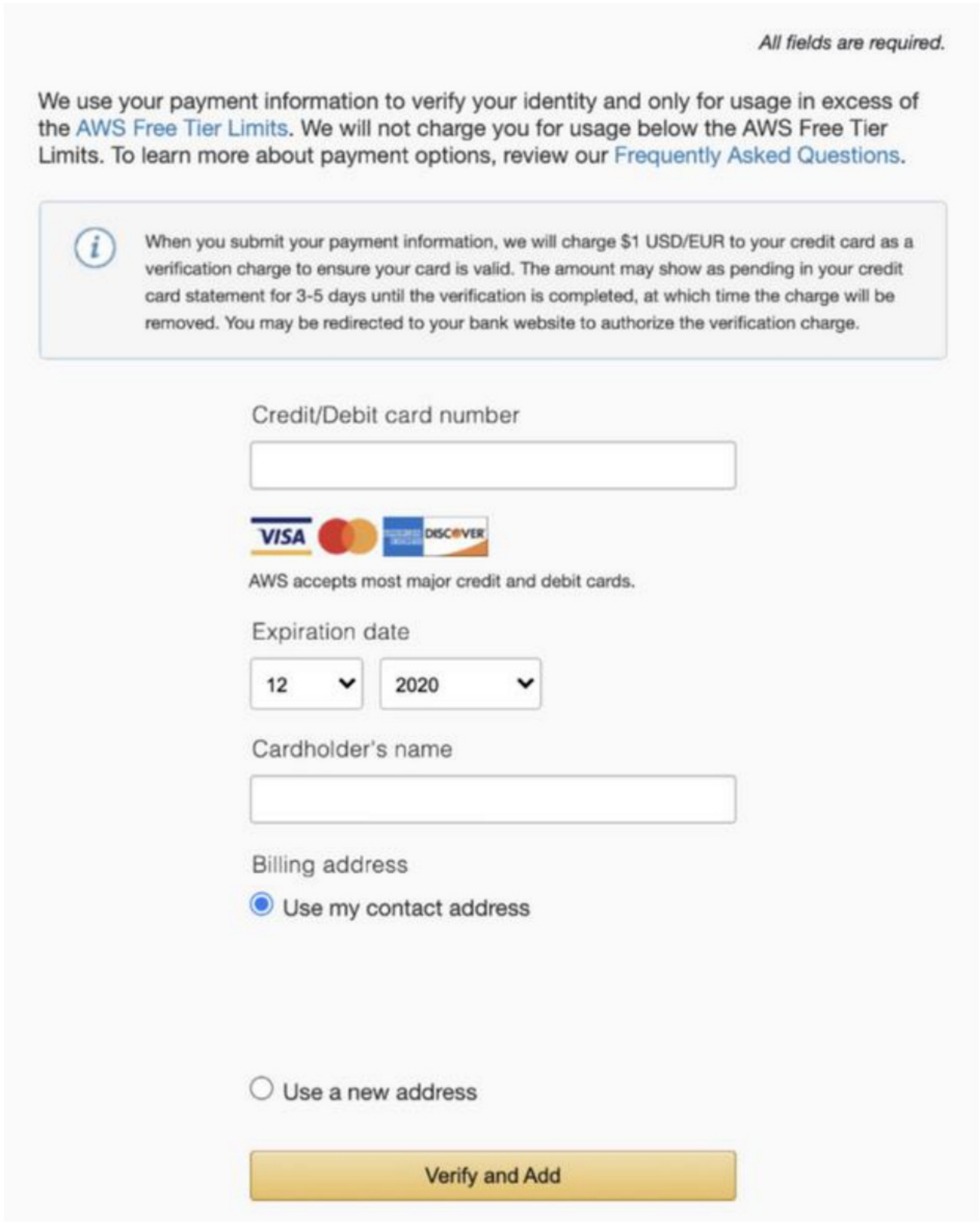
Здесь необходимо указать своё полное имя (рекомендуем указывать реальное, в противном случае подтвердить права доступа при утрате пароля или устройства MFA будет гораздо сложнее). Телефонный номер с кодом страны тоже должен быть верным, без него завершить регистрацию не удастся!

Страна, почтовый адрес и индекс тоже являются полями, обязательными для заполнения. Необходимо принять условия пользовательского соглашения.



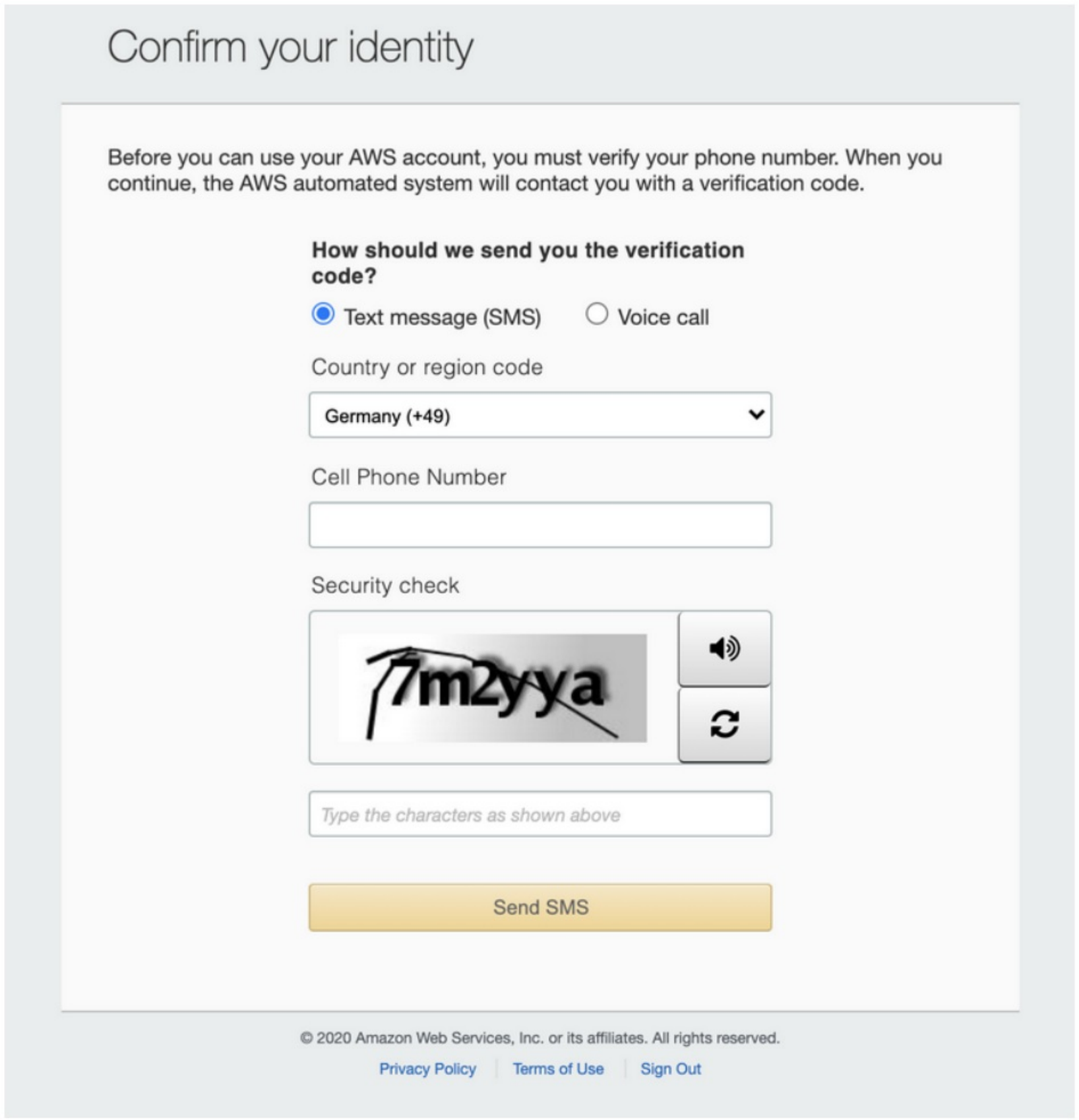
На следующем экране надо ввести данные банковской карты: принимаются карты от Visa, Mastercard, American Express и Discover. Обратите внимание, карта должна быть действующая — AWS снимет с неё один евро или доллар для проверки введённых данных. Деньги, разумеется, вернут, но это может занять несколько дней.

Если вы создаёте аккаунт впервые, то попадаете под условия программы AWS Free Tier и большинство услуг в первый год будут бесплатны. Только убедитесь, что вы используете ресурсы с пометкой «free tier». На этом мы подробно остановимся в следующей части.

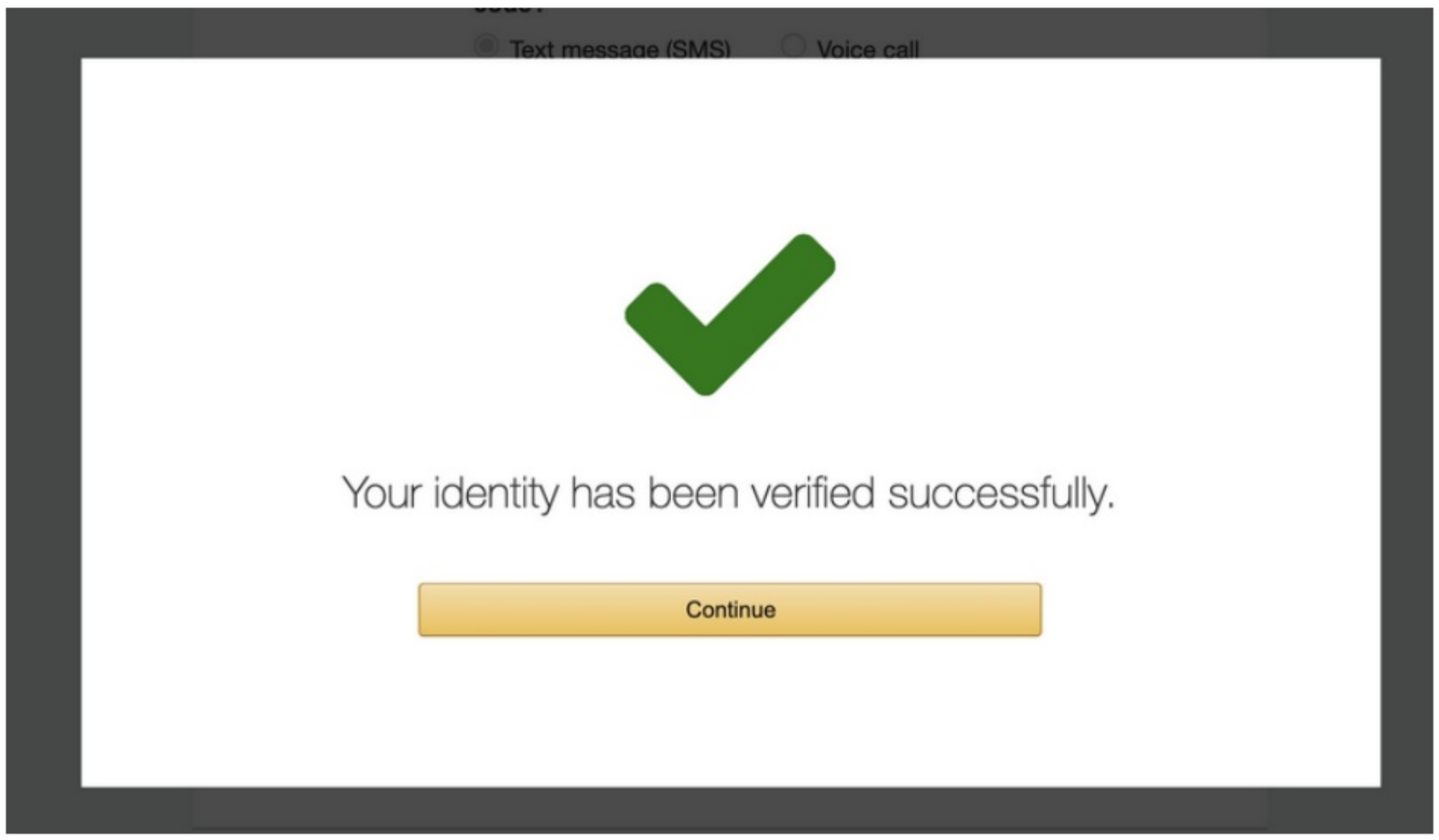


На последнем шаге надо подтвердить ваш номер телефона. Самый простой путь: выбирайте Text Message, выберите страну, введите номер телефона и капчу.

394783

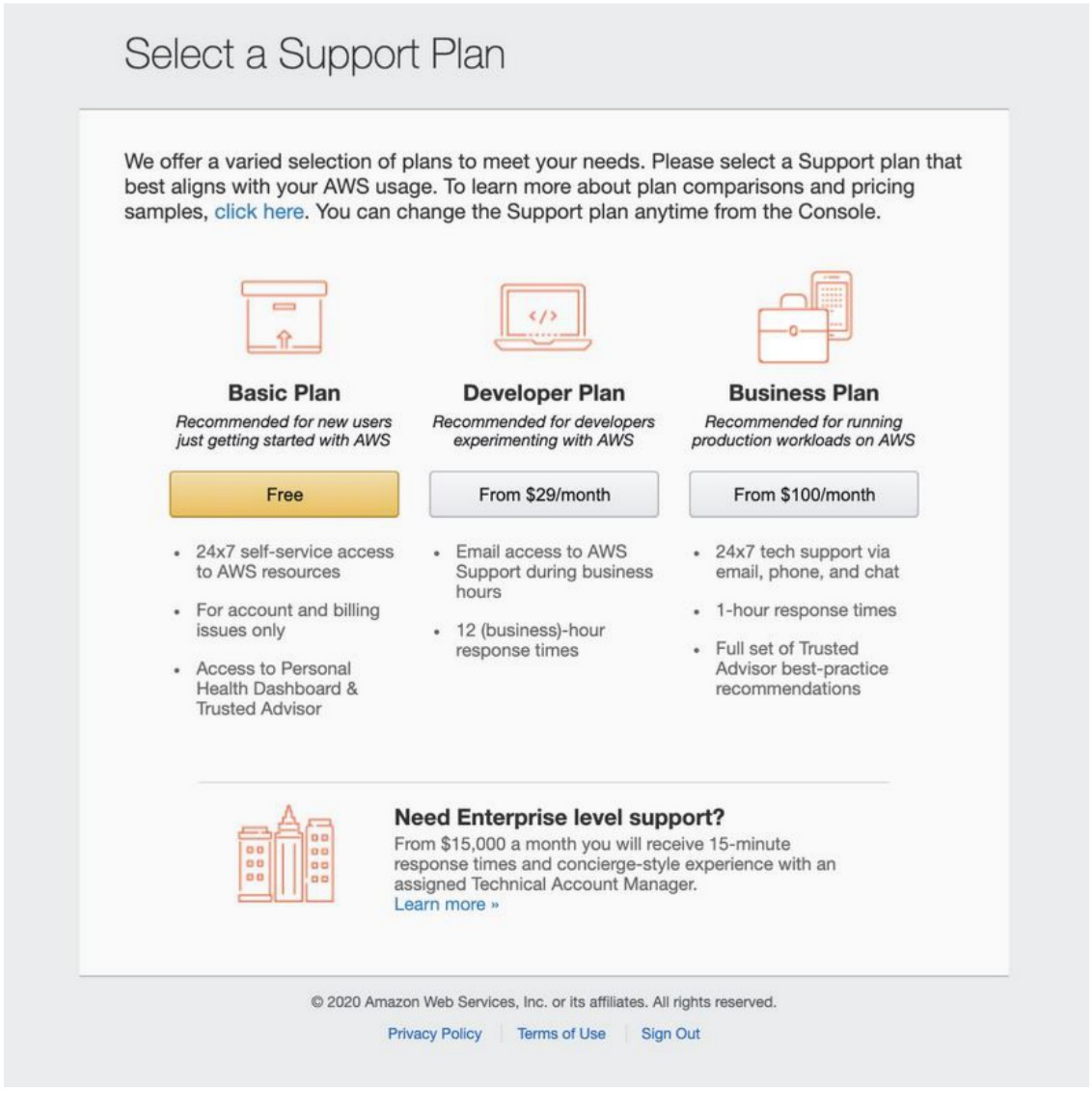


Обычно, SMS приходит сразу же, и уже через несколько секунд вы увидите заветную зелёную галочку!



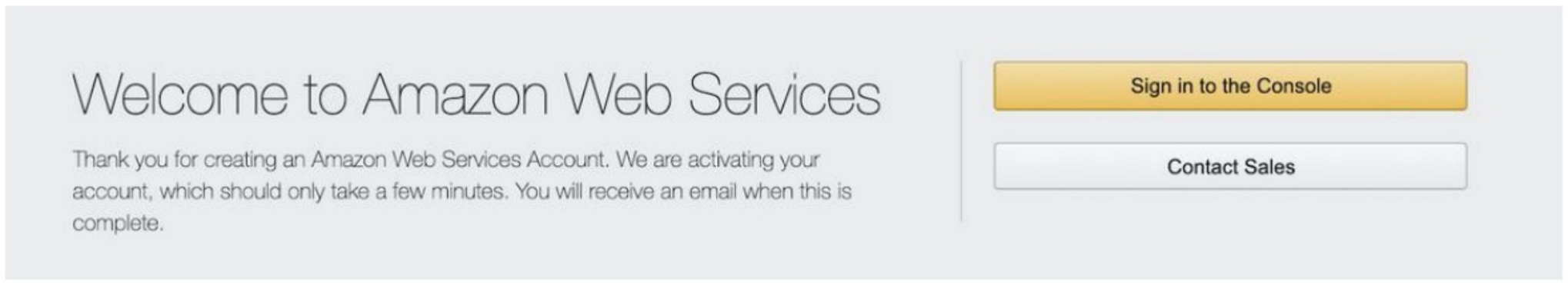
В последнем шаге необходимо выбрать план поддержки: в бесплатном basic plan вам предложат искать помощь на форумах, а платные developer plan, business plan и

enterprise plan гарантируют ответ от технической поддержки AWS. В большинстве случаев, basic plan будет хорошим выбором для начинающего разработчика облачных решений - если вы дружите с Stack Overflow и Google, конечно!

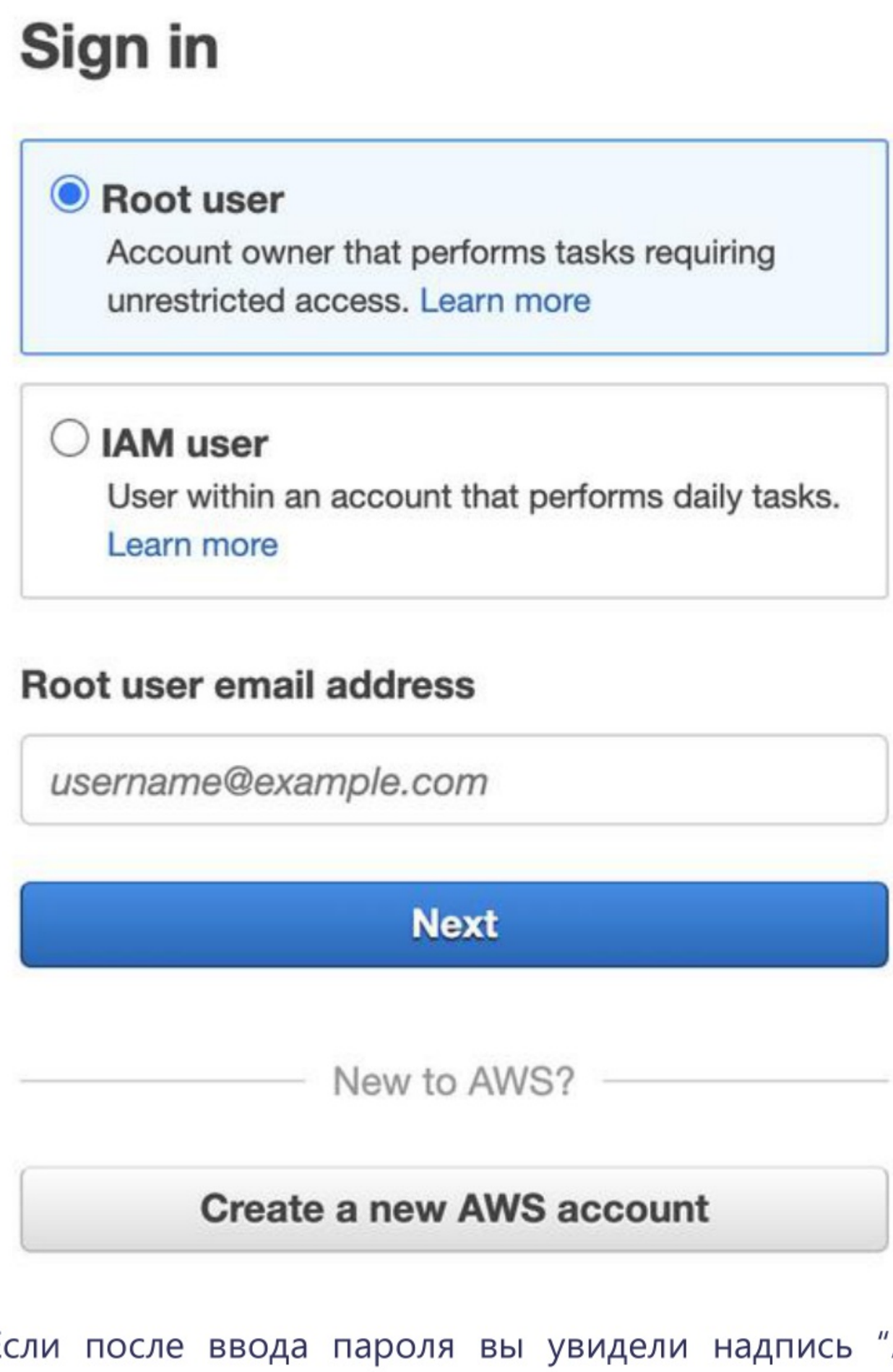


После выбора плана поддержки, необходимо зайти в AWS Console – так называется веб-интерфейс Amazon Web Services, который позволяет управлять всеми сервисами этой платформы.

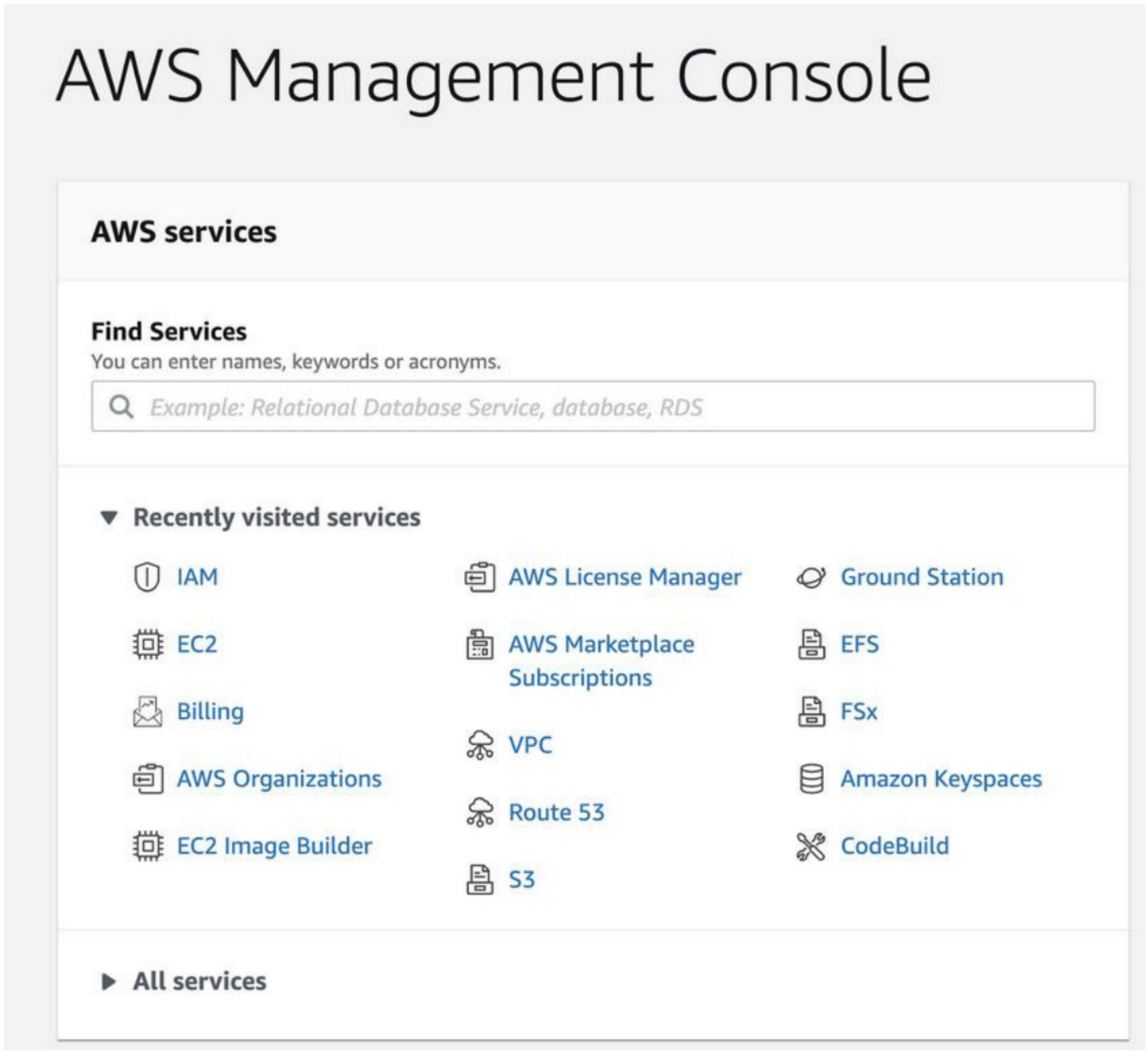
Кликните на “Sign in to the Console”.



При первом входе в аккаунт, нужно выбрать Root User — это администраторский доступ к управлению вашим облаком. Скоро мы создадим IAM пользователя, чтобы работать с платформой более безопасным способом, но пока что потребуется зайти как root. Укажите e-mail, использованный при регистрации.



Если после ввода пароля вы увидели надпись “AWS Management Console” — поздравляем, вы успешно создали аккаунт и готовы продолжать наше путешествие!

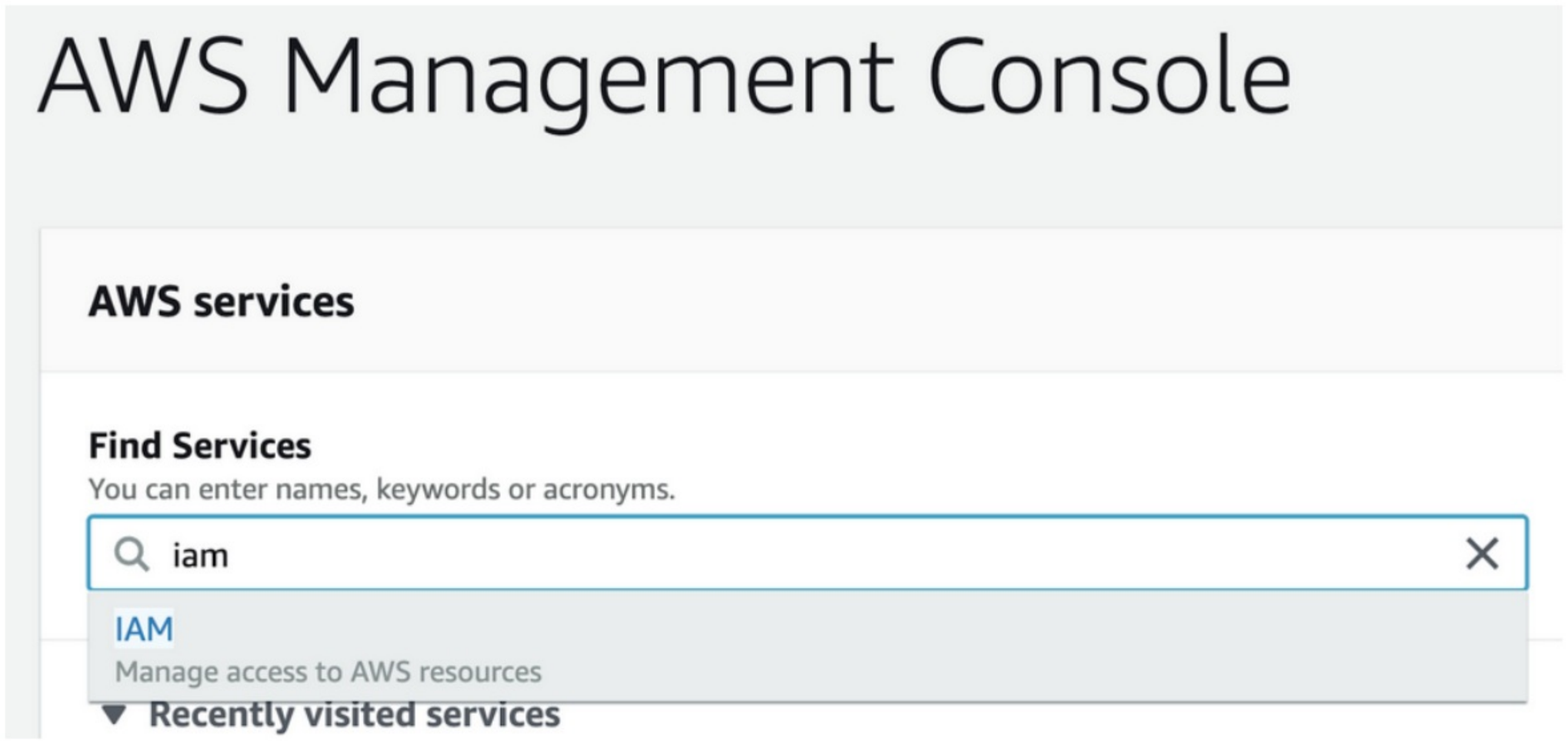


Как и в линуксе, учётная запись root имеет полный безусловный доступ к системе, поэтому очень важно держать её защищённой. В следующем коротком видео мы обезопасим учётную запись при помощи многофакторной авторизации.

Создание учётной записи IAM

Для работы с вашим облаком лучше всего создать специальную учётную запись в сервисе Identity and Access Management. Давайте этим займёмся!

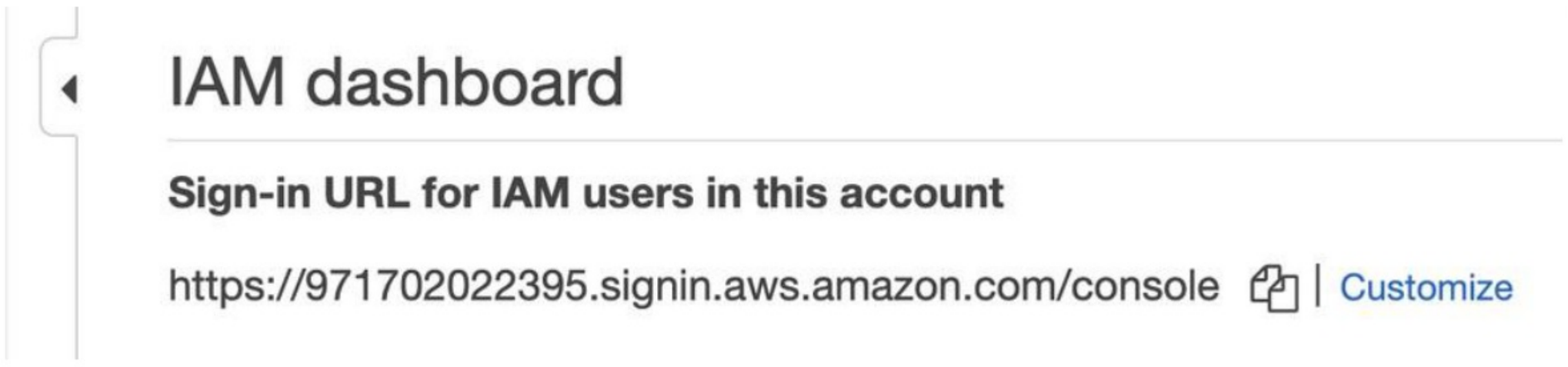
В первую очередь, выберите в консоли AWS необходимый сервис: IAM. Проще всего просто ввести “iam” в строке поиска:



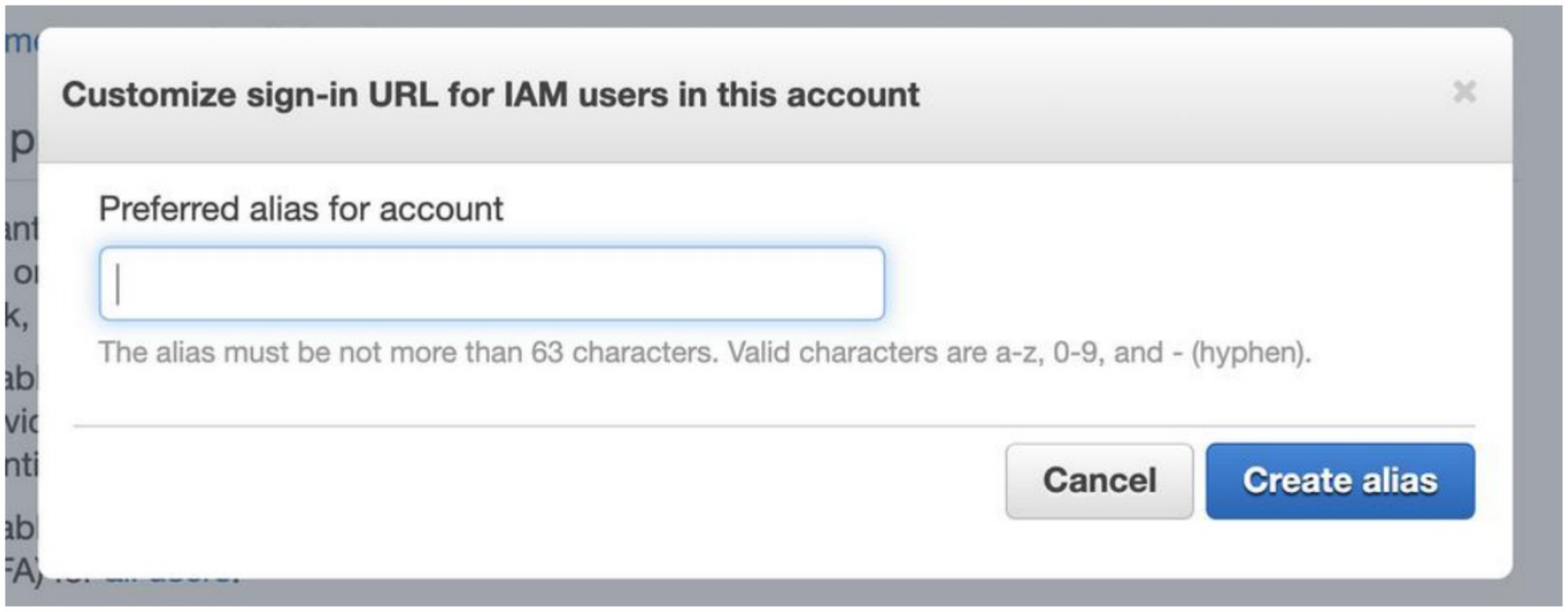
Сервис IAM позволяет не только управлять пользователями, группами, политиками и ролями, но и использовать внешних поставщиков аутентификации, таких как SAML или OpenID Connect, а так же анализировать выданные доступы и создавать отчёты.

Это очень гибкая и мощная система, но сейчас нас интересует самый простой случай — необходимо создать учетную запись пользователя, которая предоставила бы нам значительные полномочия, но при этом была ограничена в правах.

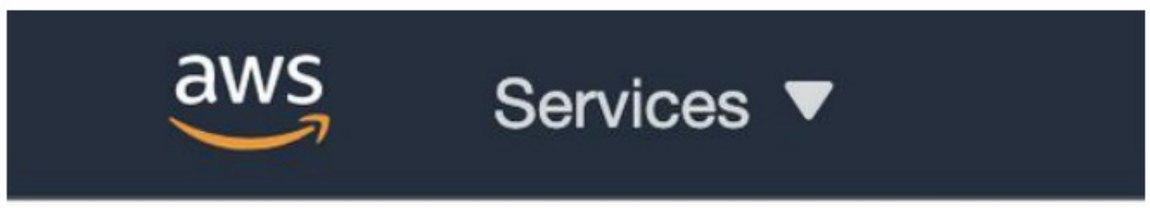
В первую очередь, скопируйте и сохраните ссылку на страницу входа вашей записи — она пригодится, когда вы будете входить в систему как IAM пользователь, ведь тогда потребуется не только логин и пароль, но и идентификатор аккаунта.

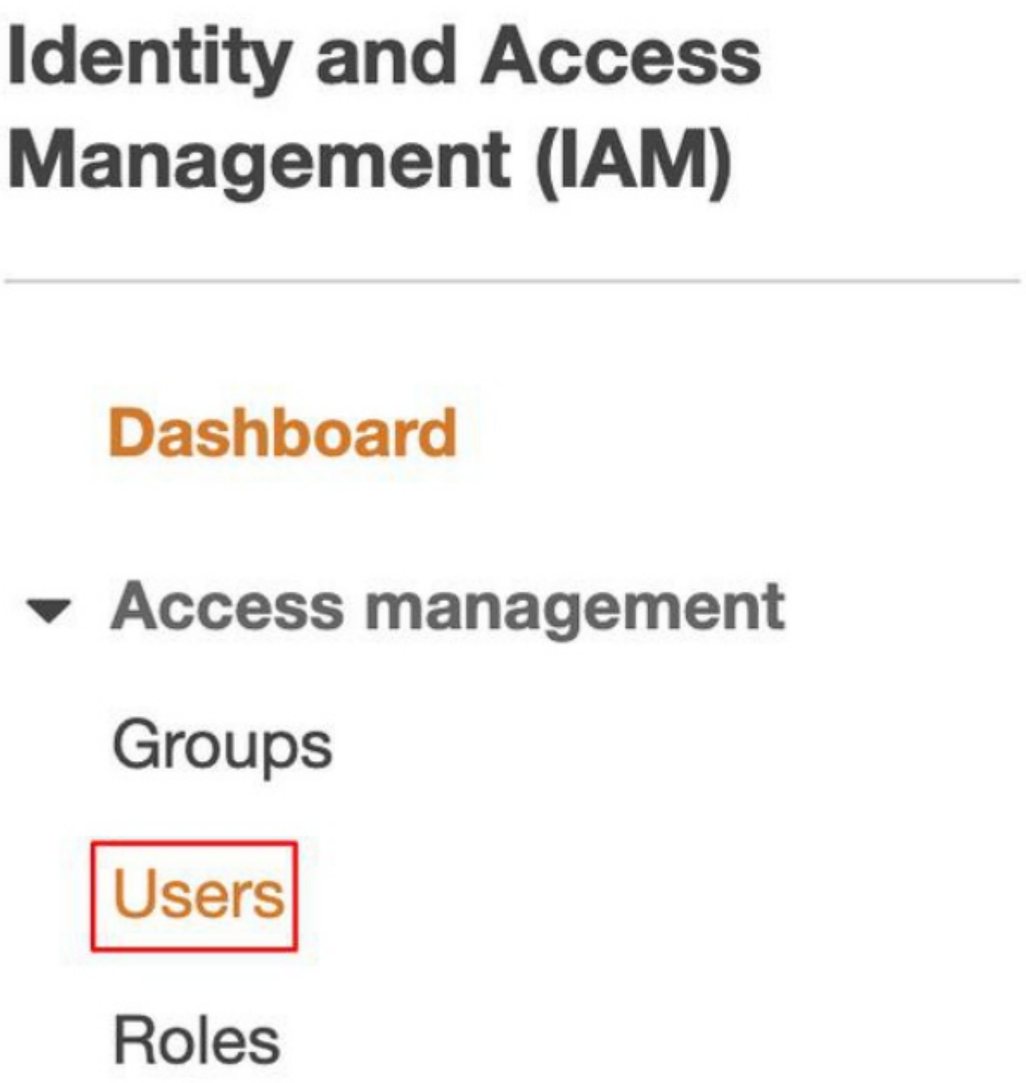


Ваша ссылка будет немного отличаться из-за другого номера аккаунта. Обратите внимание, что эту ссылку можно изменить, нажав на “Customize” — это позволит ввести собственный идентификатор (алиас) аккаунта, если такой ещё свободен, конечно.

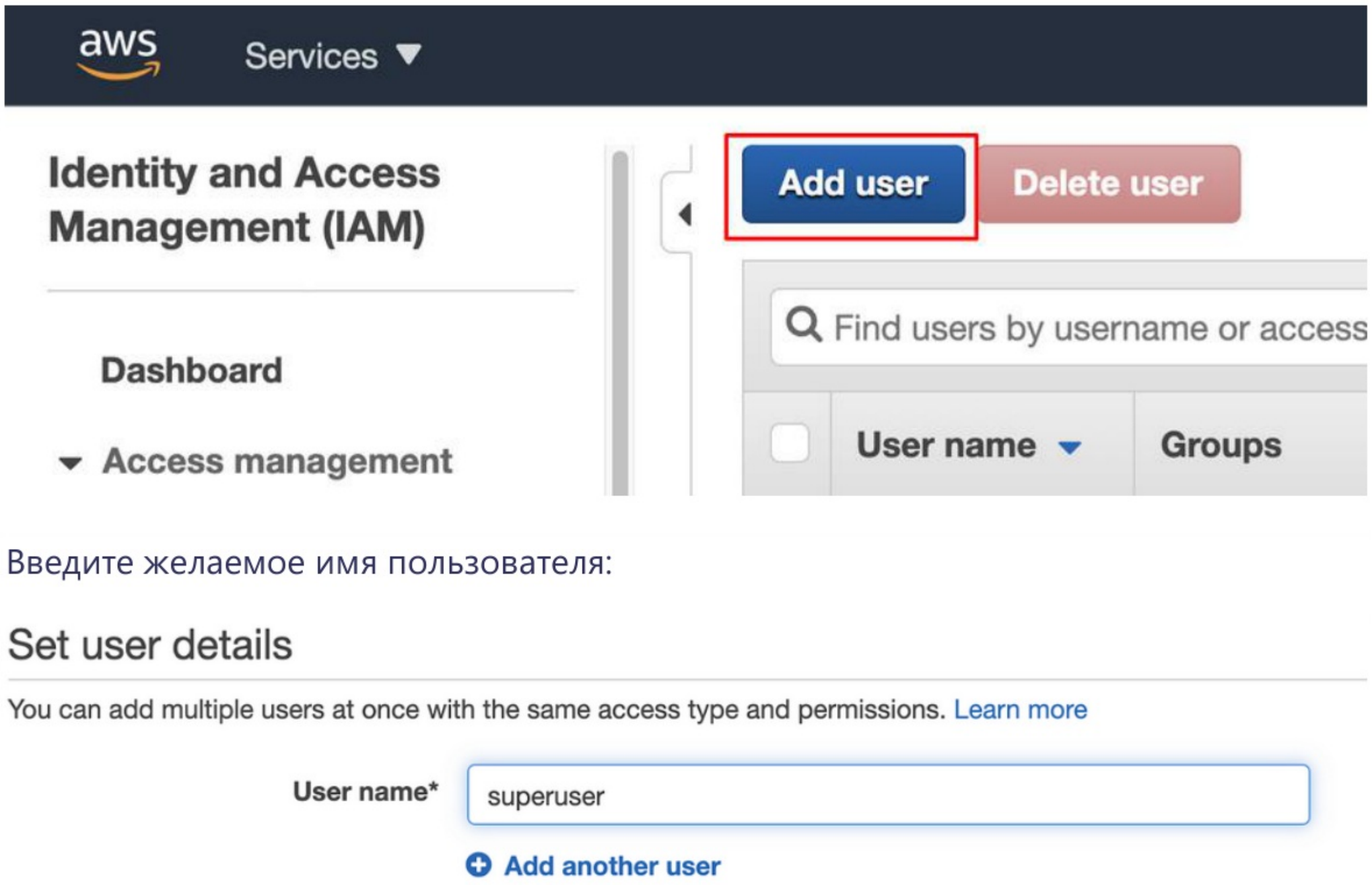


Теперь перейдите на вкладку Users:



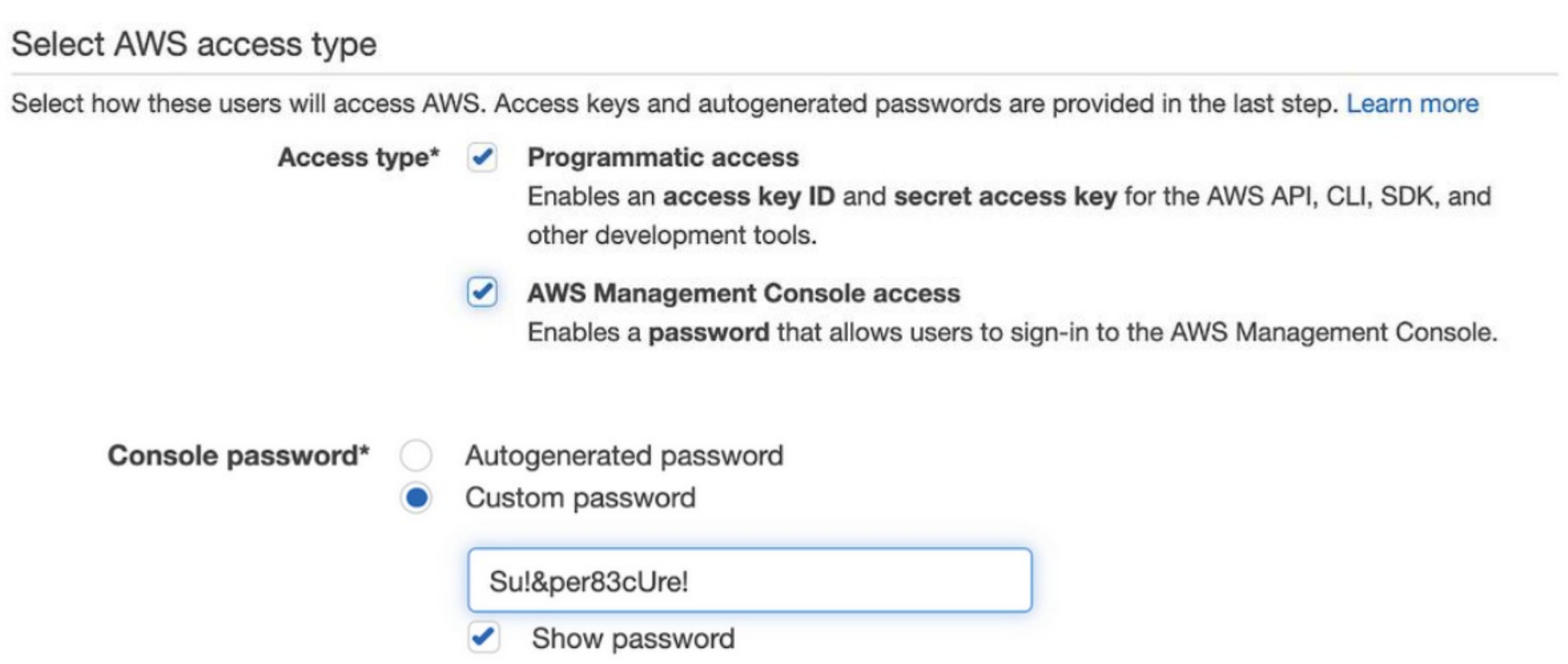


И нажмите на кнопку “Add User”.



Введите желаемое имя пользователя:

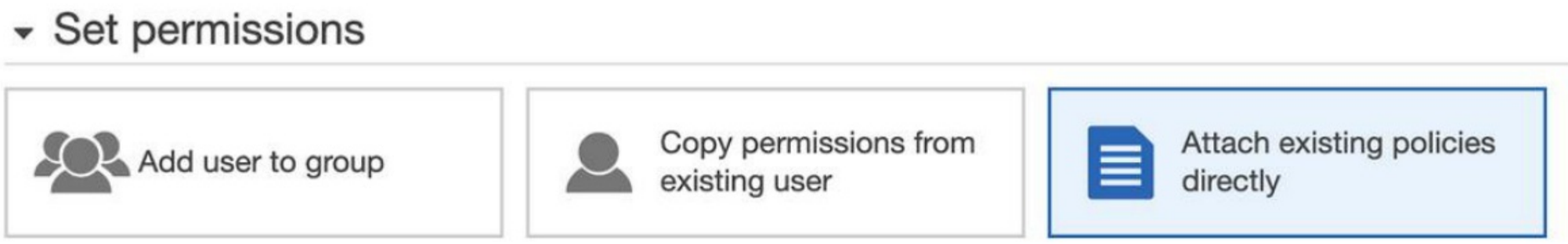
Отметьте необходимый вам тип доступа. Для прохождения курса потребуются оба типа, и “programmatic access” (доступ к API), и “AWS Console access” (доступ к веб-интерфейсу). В дальнейшем неиспользуемый тип доступа будет лучше отключить.



Поскольку мы запросили доступ к AWS Console, необходимо создать пароль: сгенерировать его или ввести собственный. Если вы создаёте учётную запись не для себя, можете потребовать сменить пароль при первом входе. В этом случае

создаваемому пользователю будет автоматически назначена политика безопасности IAMUserChangePassword, чтобы разрешить смену пароля.

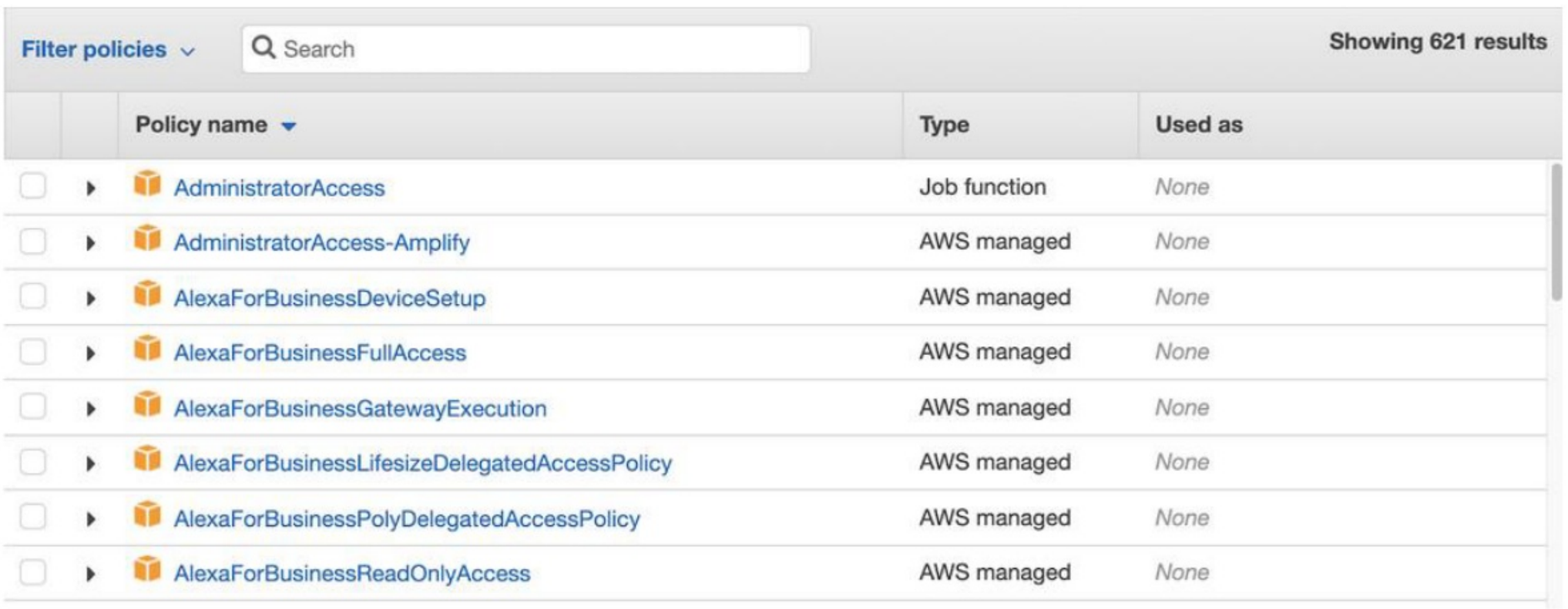
Следующим шагом новому пользователю нужно настроить доступ:



Сделать это можно тремя способами:

- добавить пользователя в группу
- скопировать разрешения существующего пользователя
- напрямую назначить политики безопасности

Поскольку ни групп, ни других пользователей IAM у нас ещё нет (root-аккаунт не считается), придётся назначить политики напрямую, смело выбирайте этот пункт.

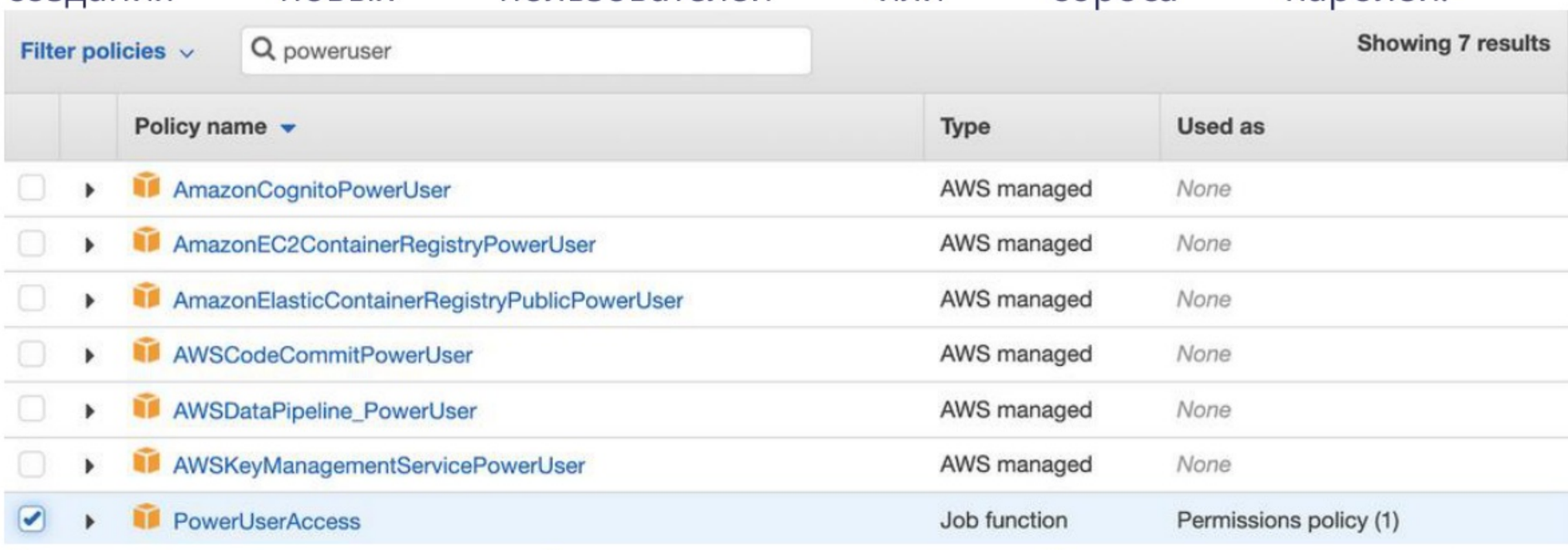


На сегодняшний день AWS IAM предлагает около 640 (количество постоянно меняется) предустановленную политику безопасности, и если этого вам мало, можно создать собственную! В рамках этого задания мы остановимся на уже существующих.

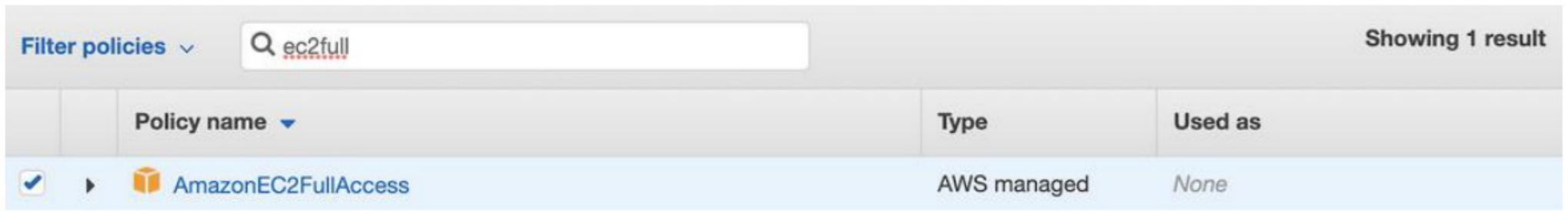
Здесь вы можете пойти двумя путями:

1. Найти и выбрать PowerUserAccess — эта политика даст вам доступ ко всем сервисам AWS, за исключением доступа к управлению аккаунтами. Это достаточно хороший баланс между возможностями учётной записи и безопасностью — вам придётся переключаться в root-аккаунт только для

создания новых пользователей или сброса паролей.



2. Если же вы предпочитаете более строгие стандарты безопасности, можно остановиться на AmazonEC2FullAccess — эта политика предоставит полный доступ к сервису Elastic Cloud Computing, с которым мы начинаем работать уже совсем скоро. Обратите внимание, что AmazonEC2FullAccess будет недостаточно для выполнения практической части следующих модулей, поскольку мы начнём работать с другими сервисами, но вы всегда можете зайти в IAM, используя root аккаунт и добавить вашему пользователю новые политики.



Заметьте, что политики можно комбинировать, и, если к AmazonEC2FullAccess добавить ReadOnlyAccess, то ваш пользователь будет иметь право на просмотр любых ресурсов, но создание, изменение и удаление только ресурсов EC2.

Policies > ReadOnlyAccess

Summary

Policy ARN	arn:aws:iam::aws:policy/ReadOnlyAccess
Description	Provides read-only access to AWS services and resources.

Permissions | **Policy usage** | **Policy versions** | **Access Advisor**

Policy summary | { } JSON

Filter

Service	Access level	Resource
Allow (162 of 257 services) Show remaining 95		
Access Analyzer	**Full**: List, Read	All resources
Alexa for Business	**Full**: List **Limited**: Read	All resources
Amplify	**Limited**: List, Read	All resources
API Gateway	**Full**: Read	All resources

Практические шаги в последующих уроках исходят из использования вами PowerUserAccess.

В следующем шаге можно назначить теги на пользователя: в данном случае это не так важно, но когда вы начнёте управлять командным проектом, мы советуем назначать каждому пользователю как минимум такие теги, как Name, E-mail, Department.

Add tags (optional)

IAM tags are key-value pairs you can add to your user. Tags can include user information, such as an email address, or can be descriptive, such as a job title. You can use the tags to organize, track, or control access for this user. Learn more

Key	Value (optional)	Remove
Email	evilcoder@evilcoding.io	✖
Name	Vasilii Ivanov	✖
Team	Development	✖
Add new key		

You can add 47 more tags.

На следующей странице вам предложат проверить введённые данные и убедиться, что вас всё устраивает.

Review

Review your choices. After you create the user, you can view and download the autogenerated password and access key.

User details

User name	superuser
AWS access type	Programmatic access and AWS Management Console access
Console password type	Custom
Require password reset	No
Permissions boundary	Permissions boundary is not set

Permissions summary

The following policies will be attached to the user shown above.

Type	Name
Managed policy	PowerUserAccess

Tags

The new user will receive the following tags

Key	Value
Email	evilcoder@evilcoding.io
Name	Vasilii Ivanov
Team	Development

Cancel Previous Create user

Всё верно? Жмите на кнопку “Create User”, мы близки к победе!

Success

You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.

Users with AWS Management Console access can sign-in at: https://971702022395.signin.aws.amazon.com/console

Download .csv

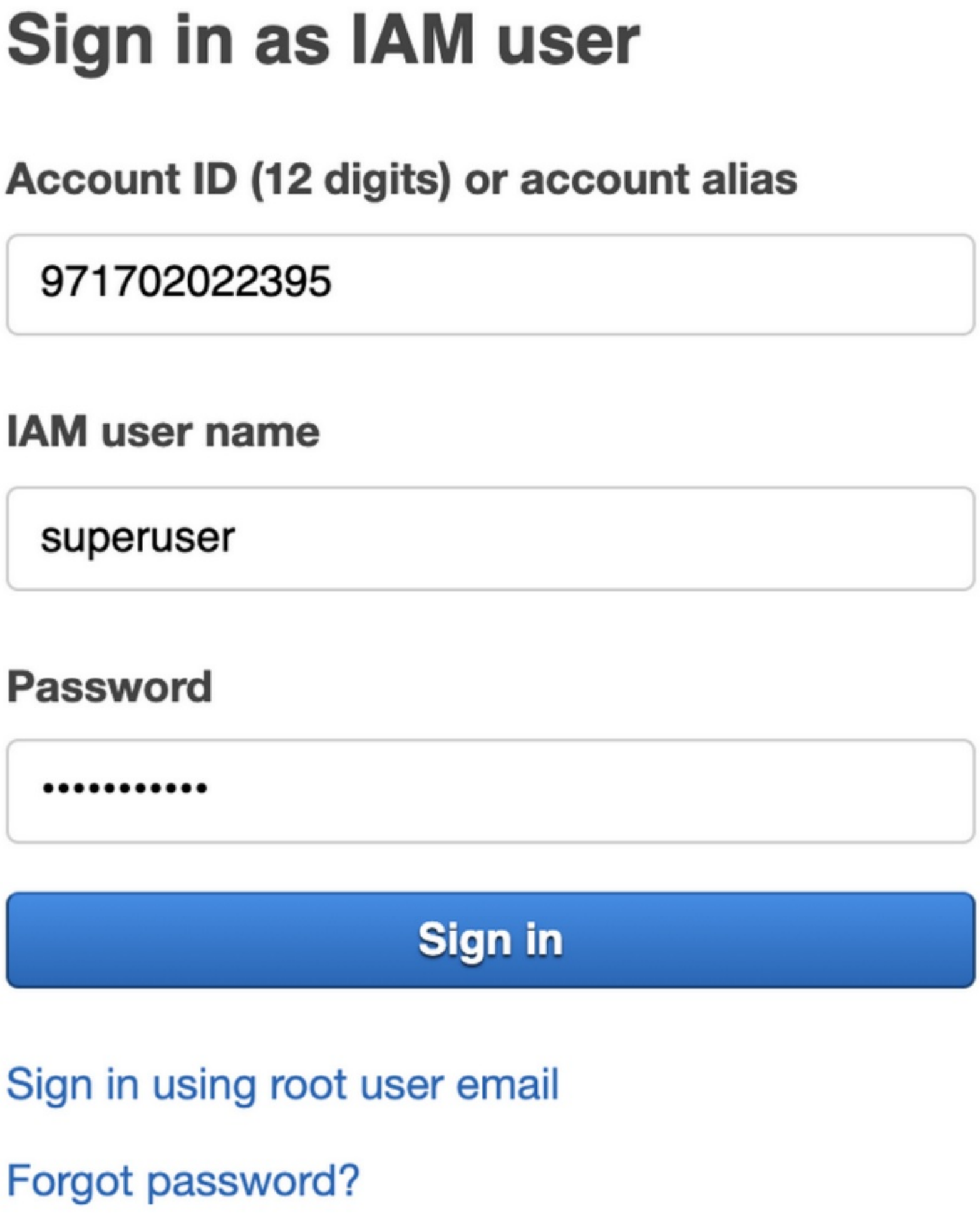
	User	Access key ID	Secret access key	Email login instructions
▾	superuser	AKIA6EPPUGD56IKIWO4X	********* Show	Send email

- Created user superuser
- Attached policy PowerUserAccess to user superuser
- Created access key for user superuser
- Created login profile for user superuser

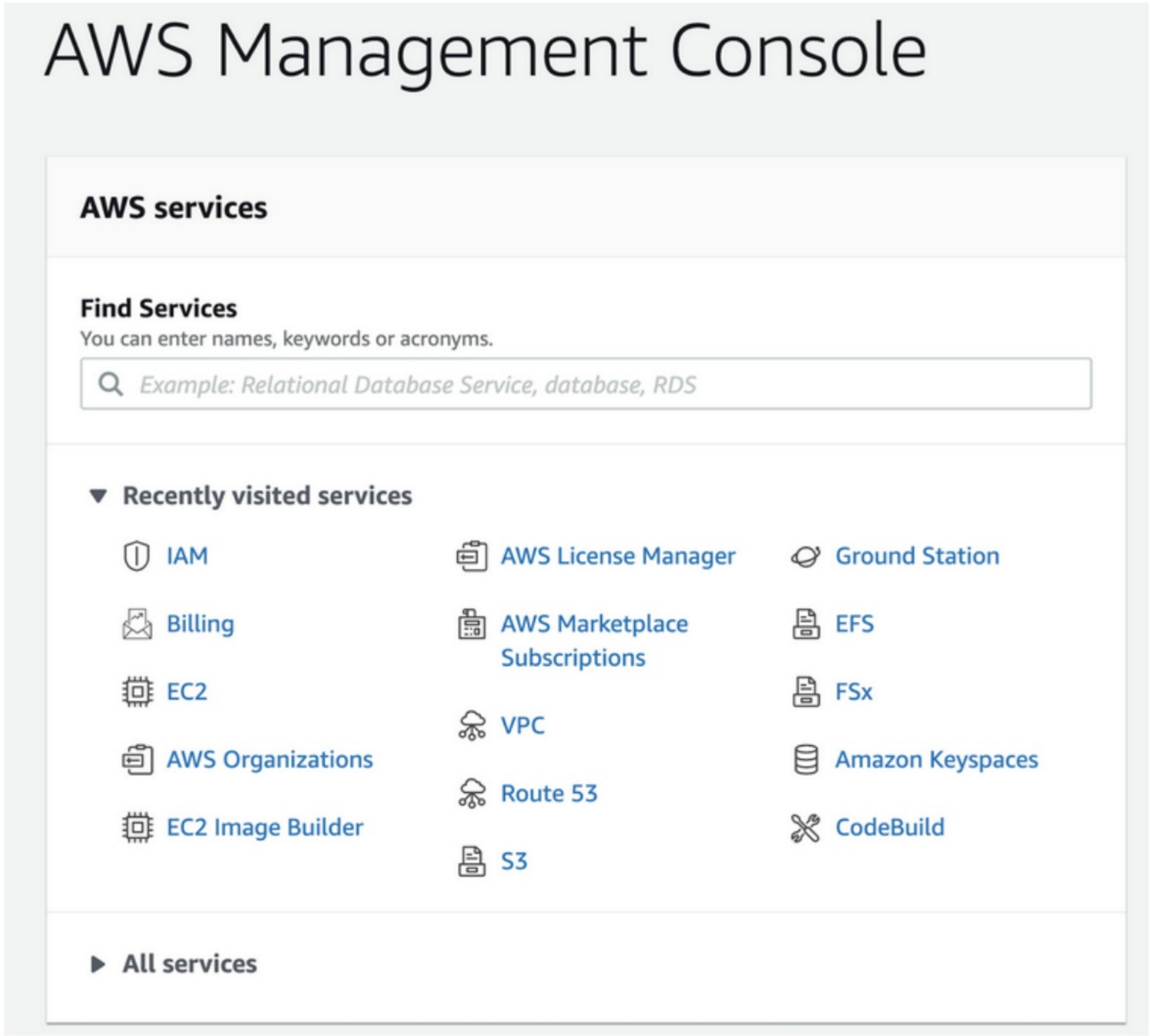
Только что IAM сервис создал для нас пользователя, назначил ему политику безопасности, создал ключи, необходимые для работы через API (они нам скоро понадобятся) и установил заданный нами пароль для входа.

Не забудьте сохранить эти данные в безопасном месте: можно просто скачать csv-файл нажав на кнопку “Download .csv” — там будет всё необходимое для работы под этой учётной записью: ссылка для входа, пароль, коды доступа. Разумеется, вы можете сохранить необходимые данные любым другим удобным вам способом, но помните о безопасности.

Готово! Осталось лишь зайти в систему, используя новую учетную запись: пройдите по предложенной ссылке, это автоматически выведет вас из системы как root-пользователя. Теперь необходимо ввести данные созданного IAM аккаунта: имя и пароль.



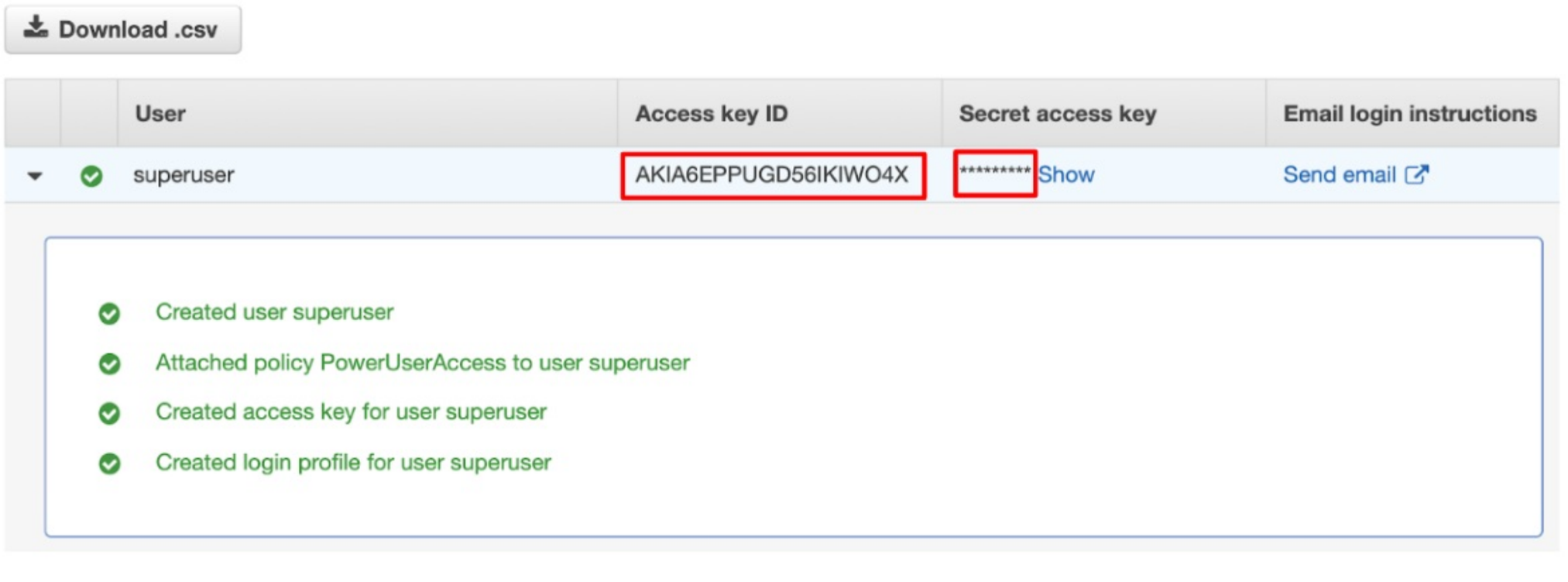
Если что-то пошло не так, вы можете повторно зайти как root, используя ссылку “Sign in using root user email” и повторить шаги, удалив пользователя и создав ещё одного. Ну а если всё прошло благополучно и вы видите AWS Management Console — поздравляем, вы великолепны!



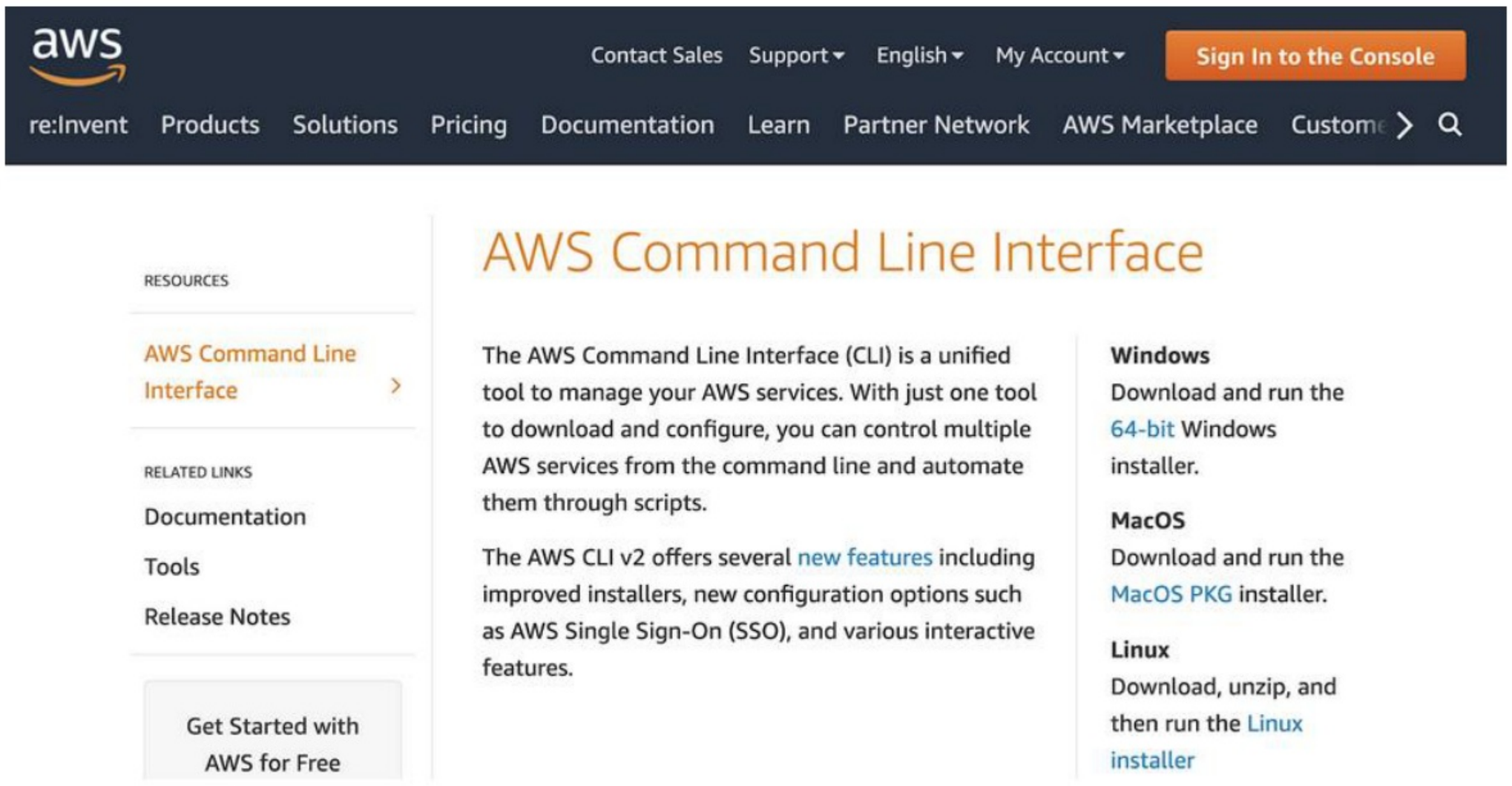
Подключение AWS Cli и пробуем создать сервер для теста

Помните, когда мы создавали IAM пользователя, я попросил вас сохранить данные? Надеюсь, что ключи доступа в наличии.

Если ключи доступа не сохранились — ничего страшного, просто создайте ещё одного пользователя IAM с Programmatic Access и доступом к EC2 (присоединена политика PowerUserAccess или EC2FullAccess). Помните, как это сделать?



В первую очередь, давайте перейдём aws.amazon.com/cli. В правой части экрана приведены ссылки на скачивание консольного клиента AWS: в наличии версии под Windows, MacOS и, конечно, Linux.



- 64-bit Windows installer
- MacOS PKG installer
- Linux installer

После вполне привычной установки «далее - далее - принять соглашение - установить», в вашей системе появится новое приложение, которое позволит нам управлять облачными сервисами, вообще не открывая браузер.

Но сначала его нужно будет сконфигурировать. Если раньше вы не применяли AWS Cli и он не был установлен, то самый простой способ сделать это — команда aws configure. Откройте командную строку и попробуйте выполнить эту команду. Если Cli начал запрашивать конфигурационные данные, значит всё хорошо!

```
wks1 21:38:21 ~  $ aws configure
AWS Access Key ID [None]: AKIA6EPPUGD56IKIWO4X
AWS Secret Access Key [None]: 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
Default region name [eu-central-1]: eu-central-1
Default output format [None]:
```

На первый вопрос AWS Access Key ID нужно ответить Access Key вашего IAM пользователя (если вы вдруг создали Access Key для Root пользователя - удалите его немедленно!). На второй вопрос придётся ввести Secret Access Key, а на третий — предпочитаемый регион.

Если вы живёте ближе к Европе, то ближайшими регионами будут, скорее всего, eu-central-1 (Франкфурт) или eu-north-1 (Стокгольм). Для жителей более восточных регионов ближайшим может оказаться, например, ap-northeast-2 (Сеул). Default Output Format пока что лучше оставить пустым.

Теперь, когда AWS Cli готов к работе, давайте вызовем команду aws ec2 describe-instances. Её легко разобрать:

- aws — обращаемся к Cli
- ec2 — будем работать с сервисом ECC
- describe-instances — просим показать существующие инстансы.

```
wks1 21:38:55 ~  $ aws ec2 describe-instances
{
    "Reservations": [
        {
            "Groups": [],
            "Instances": [
                {
                    "AmiLaunchIndex": 0,
                    "ImageId": "ami-0bd39c806c2335b95",
                    "InstanceId": "i-00e8878680bdf3a44",
                    "InstanceType": "t2.micro",
                    "LaunchTime": "2020-12-10T16:14:41+00:00",
                    "Monitoring": {
                        "State": "disabled"
```

Вы могли удивиться количеству выведенных данных, это нормально. Дело в том, что по умолчанию EC2 отображает подробную информацию по всем инстансам, причём не только запущенным, но и недавно уничтоженным. Таким образом, извлечь что-то полезное из такого вывода будет непросто. Если Cli не показал вам большой объём данных, но не выдал ошибку, значит между уничтожением последнего сервера и запросом данных прошло слишком много данных. В этом случае сначала выполните команду aws ec2 run-instances описанную в конце этой практики.

Давайте исполним эту команду, добавив несколько полезных опций:

- --query 'Reservations[*].Instances[*].[InstanceId]' — опция query позволяет уточнить, какие именно данные нам нужны. В данном случае нам интересны только идентификаторы.

- --filters Name=instance-state-name,Values=running - опция filters помогает отобрать нужные нам ресурсы, где Name это имя свойства, а Values — значения свойства. Выглядит немного громоздко, но позволяет фильтровать одним и тем же способом по самым разным свойствам самых разных ресурсов. В данном случае мы отбираем по свойству instance-state-name (имя статуса инстанса) значение “running”, то есть просим показать только запущенные ноды.

- --output text - так мы сообщим Cli, что вывод данных должен быть в формате text, а не json.

Попробуем целиком: aws ec2 describe-instances --query 'Reservations[*].Instances[*].[InstanceId]' --filters Name=instance-state-name,Values=running --output text

```
wks1 21:47:16 ~  $ aws ec2 describe-instances --query 'Reservations[*].Instances[*].[InstanceId]' --filters Name=instance-state-name,Values=running --output text
i-0d53a2dc4ac510327
wks1 21:47:20 ~  $
```

Получилось?

Если запущенных серверов у вас не осталось, то и показать Cli ничего не сможет. Давайте запустим что-нибудь интересное? Команда будет выглядеть примерно так:

```
aws ec2 run-instances --image-id ami-0bd39c806c2335b95 --instance-type t2.micro --key-name Aleks --security-group-ids sg-0c0c45ac39cbd706b
```

Некоторые значения вам придётся поменять. Разберём её по частям:

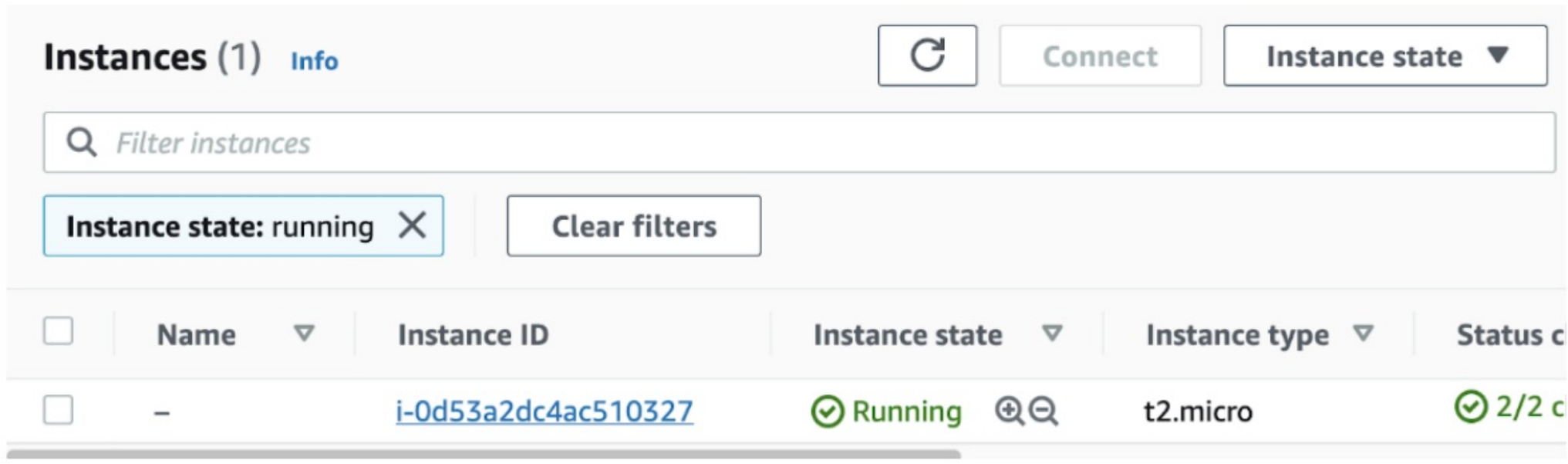
- aws ec2 run-instances - эта часть должна вам быть уже понятна: запускаем сервера EC2
- --image-id ami-0bd39c806c2335b95 - какой AMI использовать. Эта опция требует указать идентификатор AMI, который будет как отличаться от региона к региону, так и просто меняться со временем (обновление AMI, выпуск новых версий и так далее). Вам может понадобиться его обновить. Узнать актуальный AMI ID для вашего региона можно при помощи команды aws ssm get-parameters --names /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2 --query 'Parameters[*].[Value]'
- --instance-type t2.micro - эта опция должна быть понятна без пояснений - здесь мы указываем тип инстанса t2.micro
- --key-name Aleks - мы можем задать keypair, ключ, который должен быть подключен к этому инстансу. Эту опцию можно и убрать, если вы не планируете подключаться к этому серверу по SSH. Если хотите оставить, укажите соответсвующее имя keypair, созданного вами в предыдущих шагах. Просмотреть существующие ключи можно при помощи команды aws ec2

describe-key-pairs. Обратите внимание, что указать надо не ID, а именно имя ключа
- --security-group-ids sg-0c0c45ac39cbd706b - последнее значение тоже опциональное: идентификатор группы безопасности (security group). Просмотреть текущие группы можно в веб-интерфейсе EC2 или при помощи команды aws ec2 describe-security-groups

Нам пришлось немало потрудится, но теперь команда готова. Давайте выполним её и посмотрим на результат:

```
wks1 21:42:41 ~  $ aws ec2 run-instances --image-id ami-0bd39c806c2335b95 --instance-type t2.micro --key-name Aleks --security-group-ids sg-0c0c45ac39cbd706b
{
    "Groups": [],
    "Instances": [
        {
            "AmiLaunchIndex": 0,
            "ImageId": "ami-0bd39c806c2335b95",
            "InstanceId": "i-0d53a2dc4ac510327",
            "InstanceType": "t2.micro",
            "KeyName": "Aleks",
            "LaunchTime": "2020-12-10T20:45:23+00:00",
            "Monitoring": {
                "State": "disabled"
            },
            "Placement": {
                "AvailabilityZone": "eu-central-1b",
                "GroupName": "",
                "Tenancy": "default"
            },
            "PrivateDnsName": "ip-172-31-36-112.eu-central-1.compute.internal",
            "PrivateIpAddress": "172.31.36.112",
            "ProductCodes": [],
            "PublicDnsName": "",
            "State": {
                "Code": 0,
                "Name": "pending"
            },
```

AWS Cli показывает наш инстанс в состоянии pending, сейчас он находится в процессе запуска. Разумеется его же можно найти во вкладке Instances сервиса EC2.



Поздравляем, теперь этот сервер ваш! Делайте с ним что захотите, но не забудьте уничтожить, когда он вам станет не нужен!