1 Указания к заданию 1. Развертывание приложения в публичном облаке.

Цель: Создать и развернуть облачное приложение.

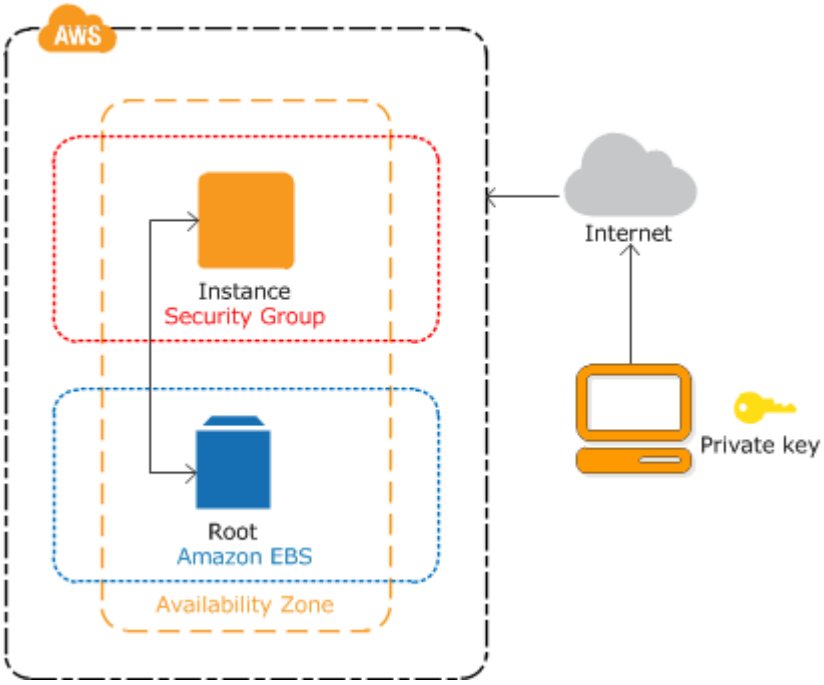
Критерии оценивания

Зарегистрироваться в системе Amazon Web Services
Запустить экземпляр виртуальной машины, показать возможность удаленного доступа/управления
Развернуть на тестовой машине сервер LAMP
Развернуть на тестовой машине сервер WordPress
Используя сервисы Amazon S3, AWS Lambda и DynamoDB создать базовый статический веб-сайт

Методические указания:

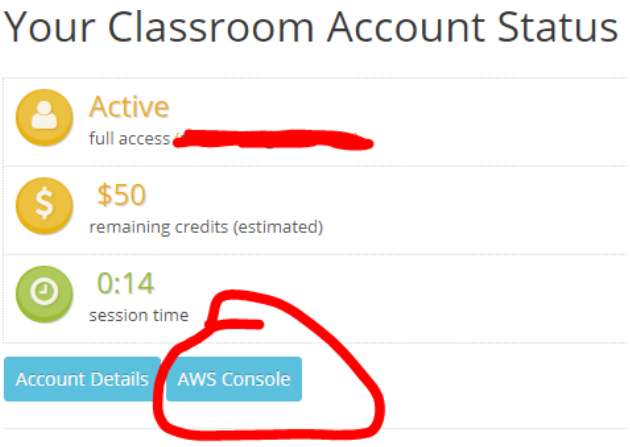
Начало работы с Amazon Elastic Compute Cloud (Amazon EC2) начинается с запуска, подключения и использования экземпляра (instance) Linux. Экземпляр — это виртуальный сервер в облаке AWS. С помощью Amazon EC2 вы можете настраивать и конфигурировать операционную систему и приложения, работающие на вашей машине.

Экземпляр виртуальной машины — это поддерживаемый Amazon EBS экземпляр. Вы можете либо указать зону доступности, в которой работает ваш экземпляр, либо позволить Amazon EC2 выбрать зону доступности для вас. Когда вы запускаете экземпляр, вы защищаете его, указывая пару ключей и группу безопасности. При подключении к экземпляру вы должны указать закрытый ключ пары ключей, который вы указали при запуске экземпляра.



1.1 Шаг 0. Регистрация в AWS

Всем студентам были высланы приглашения в AWS Educate Classroom Account по курсу « Introduction to service-oriented computing» на платформе vocareum.com. Каждый, кто прошел регистрацию получил доступ к ресурсам AWS общим объемом в $50. Для использования этих ресурсов, необходимо зайти в платформу vocareum.com с вашим e-mail от системы edu.susu.ru, выбрать класс и нажать «AWS Console».

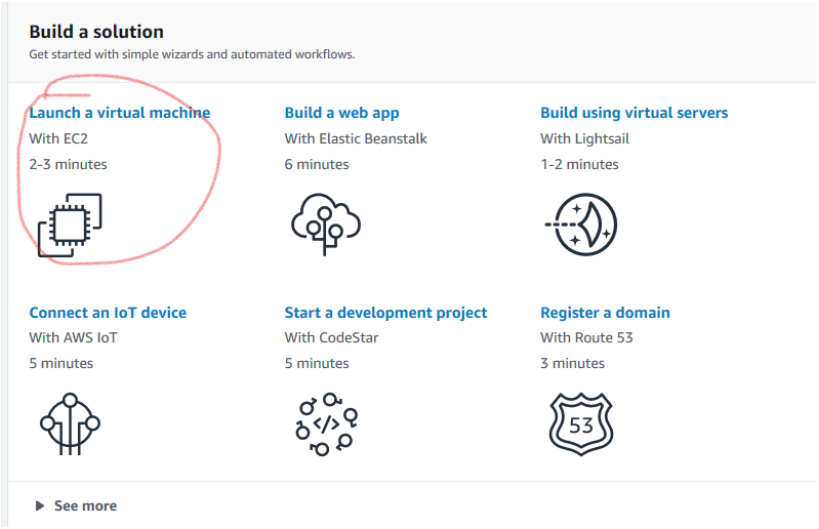


1.2 Шаг 1. Запуск экземпляра

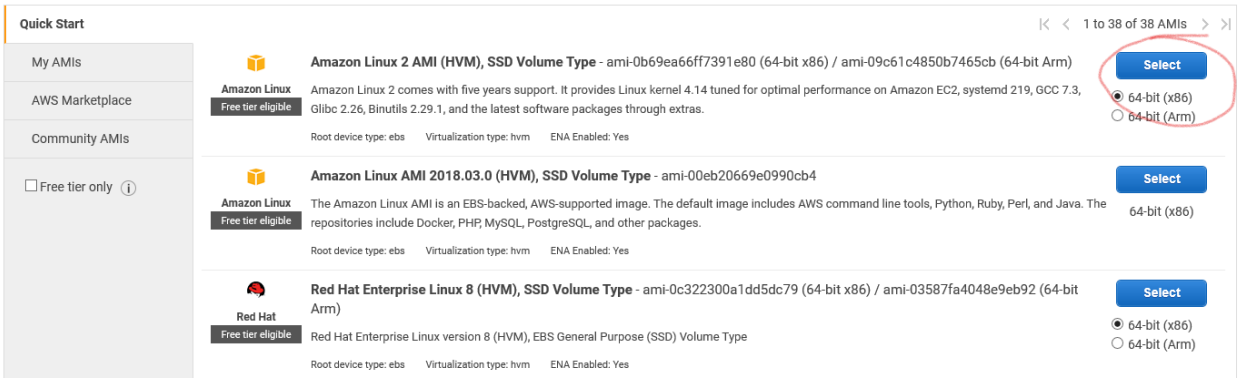
Вы можете запустить экземпляр Linux с помощью консоли управления AWS Management Console, как описано далее. Дополнительные сведения о дополнительных параметрах см. в разделе Запуск экземпляра.

Для запуска экземпляра

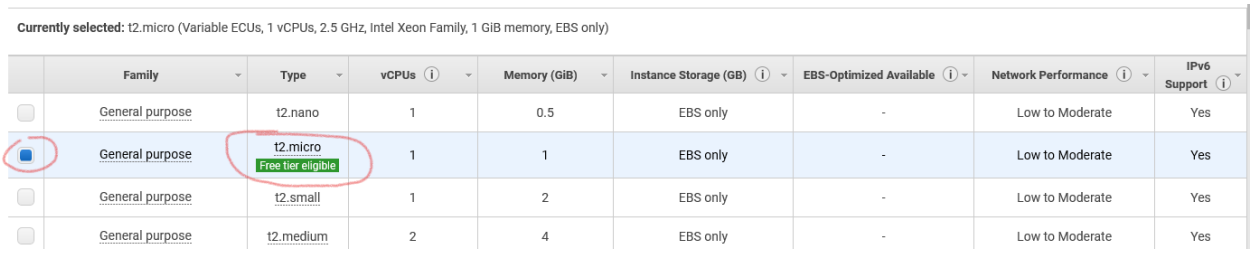
1. Откройте консоль Amazon EC2 по адресу https://console.aws.amazon.com/ec2/.
2. На приборной панели консоли выберите пункт **Запуск экземпляра (Launch Instance)**



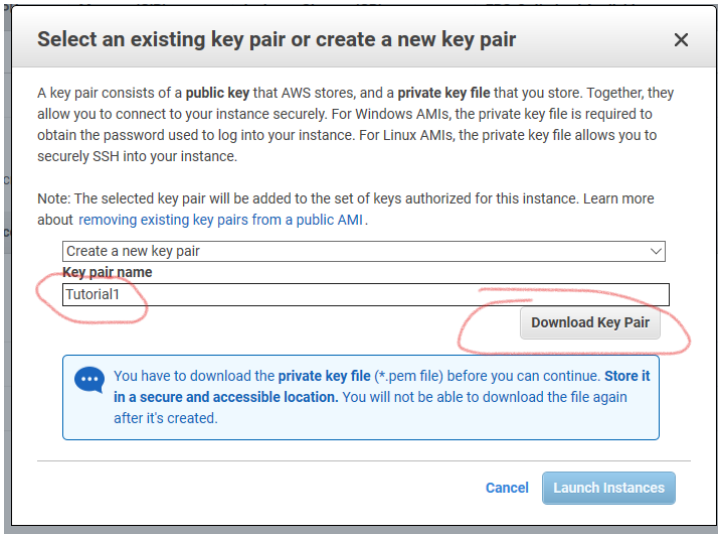
3. На странице **Choose an Amazon Machine Image (AMI)** отображается список основных конфигураций, называемых образами машин Amazon (*Amazon Machine Images,* AMI), которые служат шаблонами экземпляров виртуальных машин. Выберите HVM версию **Amazon Linux 2** и обратите внимание, что эти образы помечены как "Free Tier eligible".



4. На странице **Выбор типа экземпляра (Choose an Instance Type)** вы можете выбрать конфигурацию аппаратного обеспечения вашего экземпляра. Выберите тип `t2.micro`, который выбран по умолчанию. Обратите внимание, что данный тип инстанций подходит для бесплатного уровня.



5. Выберите **Review and Launch**, чтобы мастер выполнил все остальные настройки конфигурации за вас.
6. На странице **Review Instance Launch** в разделе **Security Groups** вы увидите, что мастер создал и выбрал группу безопасности для вас. Вы можете использовать эту группу безопасности или выбрать группу безопасности, которую вы создали при настройке с помощью следующих шагов:
    a. Выберете **Edit security groups**.
    b. На странице **Configure Security Group**, выберете **Select an existing security group**.
    c. Выберите свою группу безопасности из списка существующих групп безопасности, а затем выберите **Review and Launch**.
7. На странице **Review Instance Launch**, выберете **Launch**.
8. Когда появится запрос на пару ключей, выберете **Create a new key pair**, введите имя для пары ключей, а затем выберите **Download Key Pair**. Это единственный шанс для вас сохранить файл закрытого ключа, поэтому не забудьте его скачать. Сохраните файл закрытого ключа в безопасном месте. Вам нужно будет указывать имя вашей пары ключей при каждом запуске экземпляра и соответствующий закрытый ключ при каждом подключении к экземпляру.

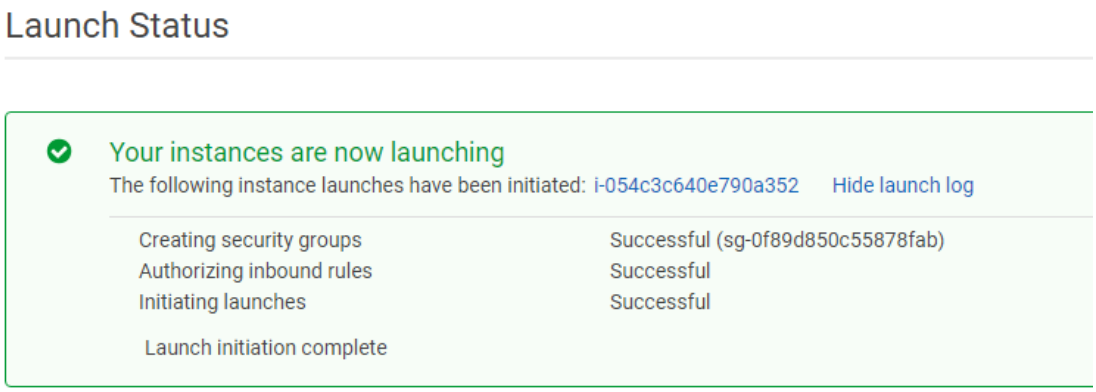


Кроме того, можно использовать ранее созданную пару ключей. Для этого выберете вариант **Choose an existing key pair**, затем выберите пару ключей, которую вы создали при настройке.

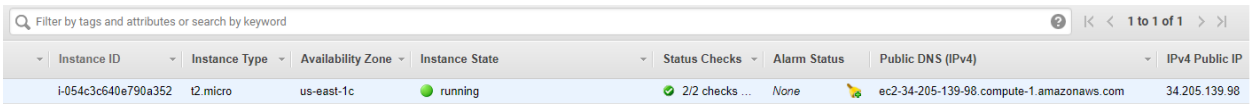
ВНИМАНИЕ!

Не выбирайте вариант **Proceed without a key pair**. Если вы запускаете экземпляр без пары ключей, то не сможете подключиться к нему.

Когда вы будете готовы, установите флажок подтверждения, а затем выберите **Launch Instances**.



9. Страница подтверждения позволяет вам узнать, что ваш экземпляр запускается. Выберите **View Instances**, чтобы закрыть страницу подтверждения и вернуться к консоли.
10. На экране **Instances** (Экземпляры) можно просмотреть состояние запуска. Запуск виртуальной машины занимает некоторое время. Когда вы запускаете экземпляр, его первоначальное состояние `pending` (в ожидании). После запуска экземпляр переходит в состояние `running` (исполняется) и получает публичное DNS имя. (Если столбец **Public DNS (IPv4)** скрыт, выберите **Show/Hide Columns** (Показать/скрыть столбцы) (значок в форме шестерни) в правом верхнем углу страницы, а затем выберите **Public DNS (IPv4)**.



1.3 Шаг 2. Подключение к экземпляру виртуальной машины

Существует несколько способов подключения к экземпляру Linux.

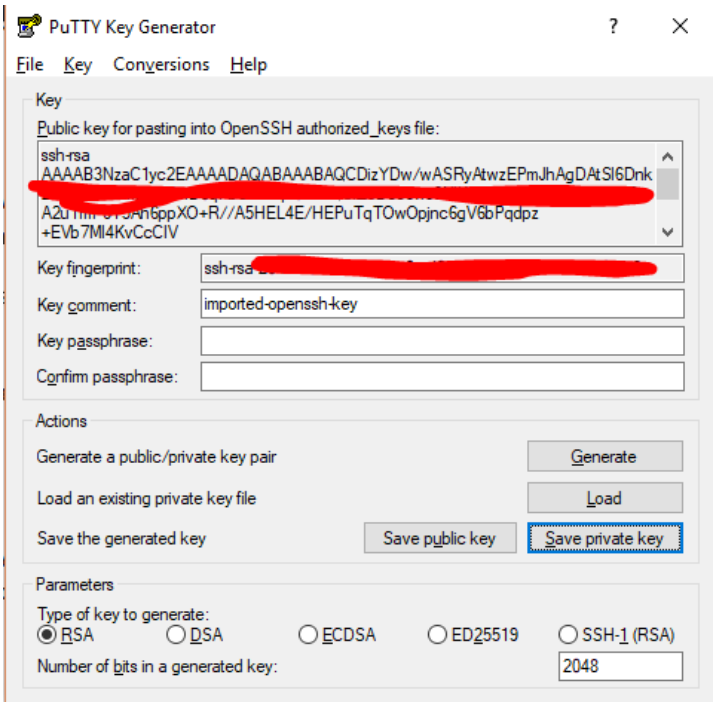
Рассмотрим процесс подключения при работе с ОС Windows.

1. При запуске вы должны сгенерировать и загрузить приватный ключ (*.pem файл)

2. Установите SSH-клиент для Windows Putty (https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html)

3. Запустите *PuTTYgen* для конвертации *.pem файла в формат RSA. Нажмите “Load”, в появившемся окне выберете «Все файлы» и откройте загруженный *.pem файл. После успешной загрузки нажмите “Save private key”.

4. Будет создан файл с расширением *.ppk



5. Откройте *Putty*

6. В панели Category выберете «**Session».**

7. В поле “Host Name” необходимо указать запись типа «*user_name@public_dns_name»*

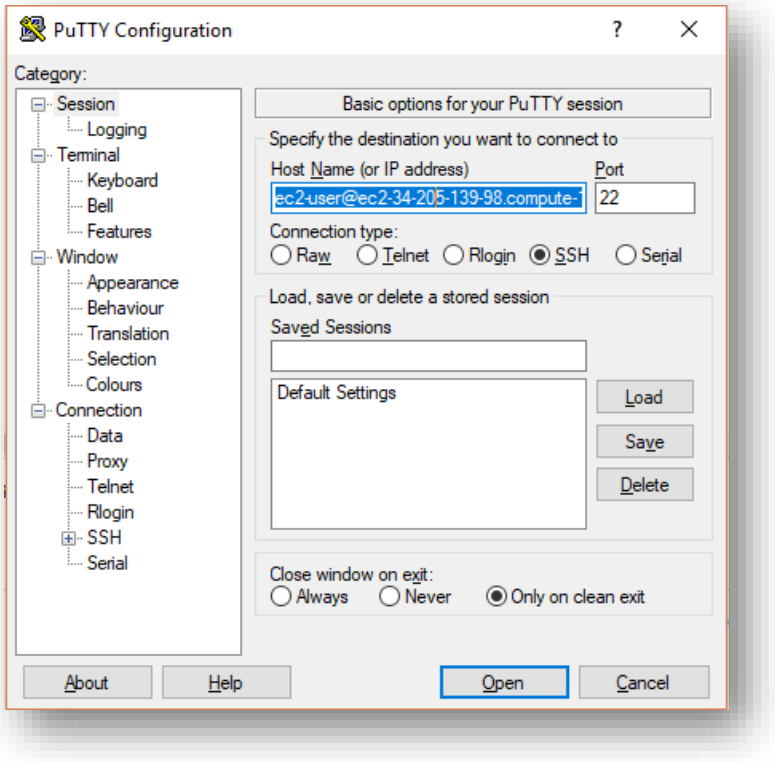
8. *user_name* – зависит от того, какой тип AMI вы выбрали:

   - Amazon Linux: ec2-user
   - Centos: centos
   - Ubuntu AMI: ubuntu.

- В других случаях – скорее всего ec2-user или root

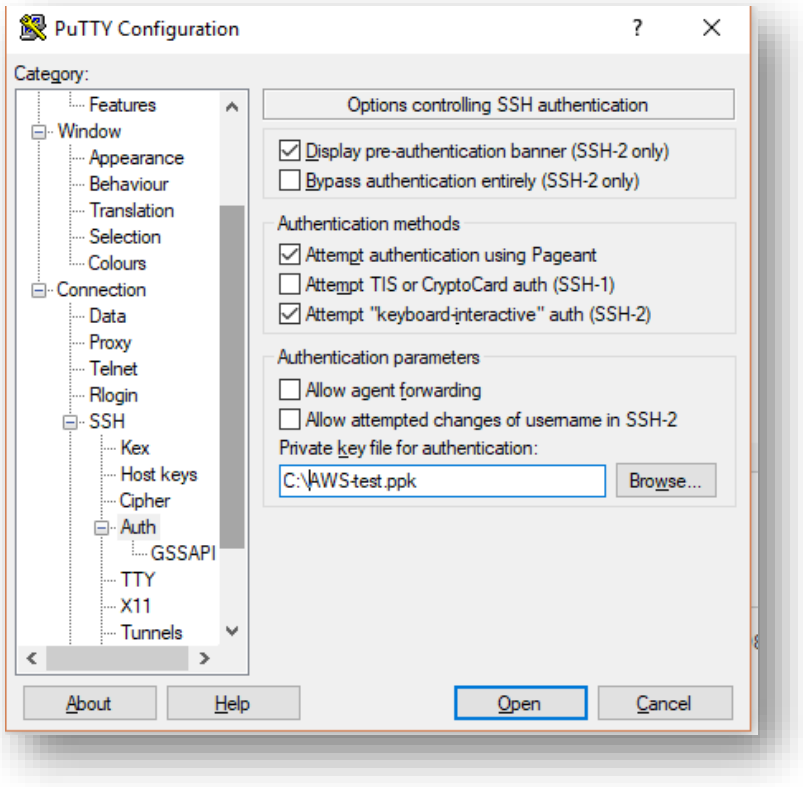
5. *public_dns_name* – вы найдете в таблице просмотра запущенных инстансов

В моем случае, в поле "Host Name" было указано: ec2-user@ec2-34-205-139-98.compute-1.amazonaws.com

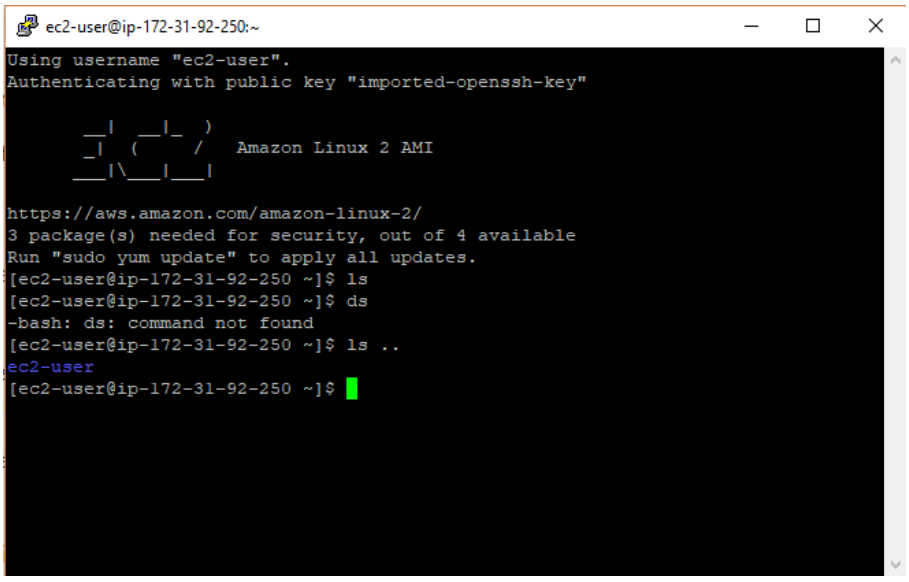


6. В панели Category выберете «Connection->SSH->Auth» и выберете ключ в поле «Private key file for authentication», нажав "Browse.." и выбрав *.ppk файл, который вы создали в приложении PuTTYgen.

7. Нажмите "Open"



6. Наслаждайтесь вашей виртуальной машиной в облаке



Дополнительные сведения, а также процесс подключения к виртуальной машине на базе других ОС см. в разделе [Подключение к экземпляру ОС Linux]

ВАЖНО!

Вы не сможете подключиться к вашему экземпляру, если вы не запустили его с парой ключей, для которой у вас есть `.pem` файл, и вы запустили его с группой безопасности, которая разрешает доступ SSH с вашего компьютера. Если вы не

можете подключиться к вашему экземпляру, обратитесь за помощью по ссылке Поиск и устранение неисправностей Подключение к вашему экземпляру.

1.4 Шаг 3: Очистка после завершения работы экземпляра (выполнять не обязательно, но необходимо знать)

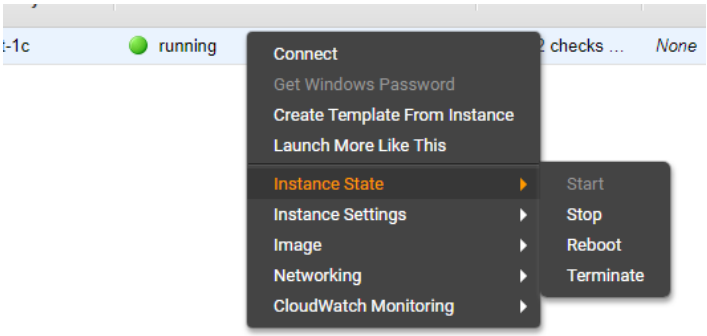
После того, как вы закончили работу с экземпляром виртуальной машины, вы должны очистить его, завершив работу.

ВАЖНО!
Завершение работы с экземпляром удаляет его; вы не можете повторно подключиться к нему после завершения работы.

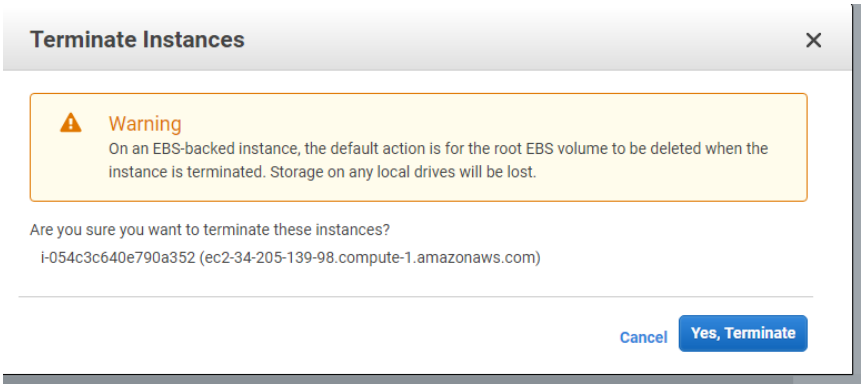
Если вы запустили экземпляр, не входящий в состав AWS Free Tier, вы перестанете нести расходы, например, как только статус экземпляра изменится на `shutting down` или `terminated`. Если вы хотите оставить копию на более поздний срок, но не платить за нее, вы можете остановить ее сейчас, а затем снова запустить позже.

Для удаления вашего экземпляра

1. В навигационной панели выберите **Instances** (экземпляры). Выберите нужный экземпляр в списке.
2. Выберете **Actions**, **Instance State**, **Terminate**.



3. Выберете **Yes, Terminate**.



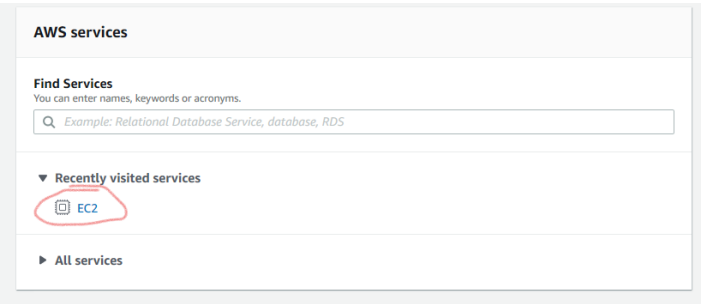
Amazon EC2 завершит работу вашего экземпляра. После завершения работы с вашим экземпляром он некоторое время будет виден в консоли, а затем запись будет удалена.

1.5 Шаг 4: Настройка виртуальной машины для запуска LAMP-сервера

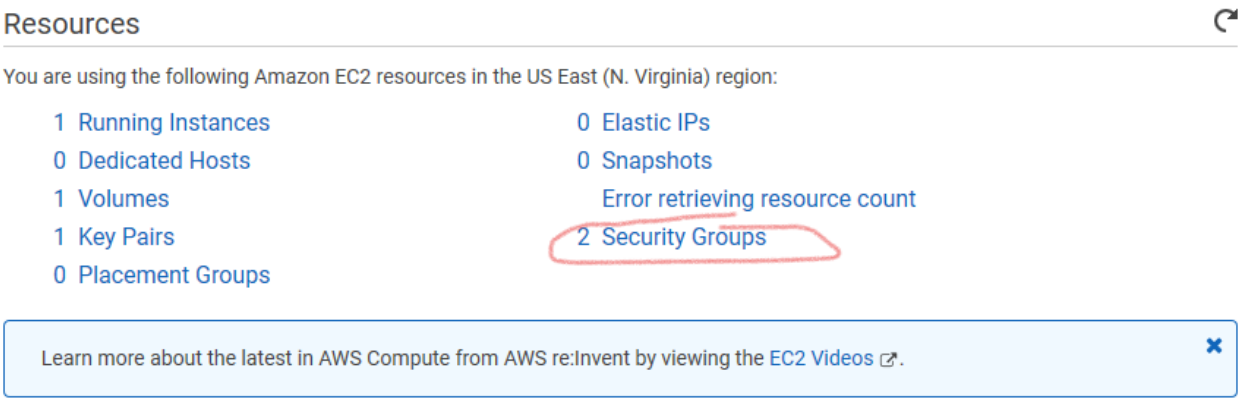
В рамках этой части практической работы вы установите веб-сервер Apache с поддержкой PHP и MariaDB (форк MySQL, разработанный сообществом) на ваш экземпляр Amazon Linux 2. Такой набор серверных приложений иногда называется веб-сервером LAMP или стеком LAMP. Вы можете использовать этот сервер для размещения статического веб-сайта или развертывания динамического PHP-приложения, которое считывает и записывает информацию в базу данных.

На данном шаге предполагается, что вы уже запустили новый экземпляр Amazon Linux 2 с публичным DNS именем, доступным из Интернета. Для работы с LAMP-сервером нам необходимо настроить свою группу безопасности, разрешив подключение по SSH (порт 22), HTTP (порт 80) и HTTPS (порт 443). Для этого необходимо:

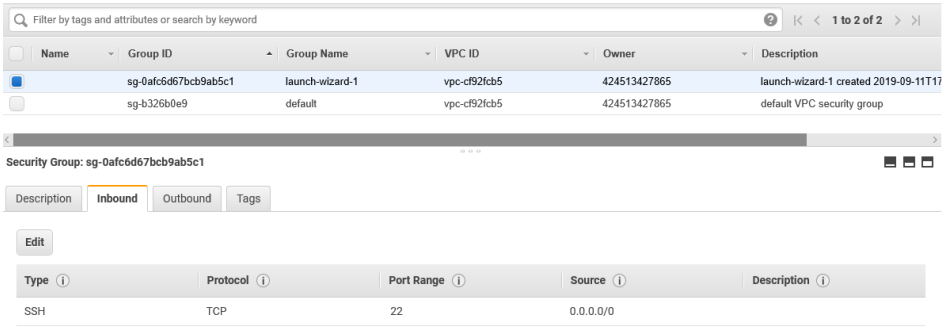
1. Перейдите в консоль управления AWS EC2. Это можно сделать с домашнего экрана, нажав на логотип AWS в верхней левой части экрана. Выберете там EC2.



2. Выберете управление **Security Groups**.



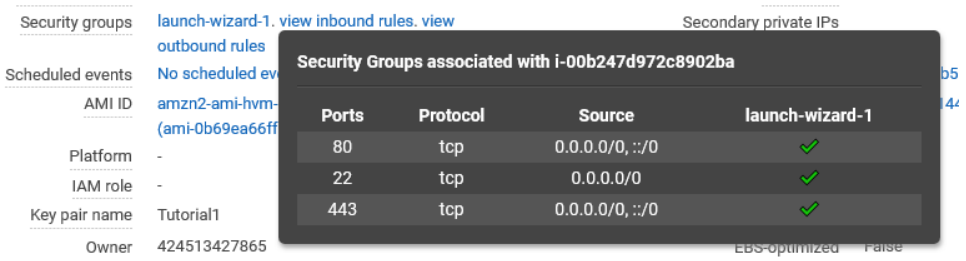
3. Выберете **Security Group** которая привязана к вашему экземпляру виртуальной машины. Внизу появится панель, где вы можете указать требуемые правила безопасности.



4. По умолчанию, виртуальная машина сконфигурирована на прием соединений по SSH (порт 22) для возможности удаленного подключения и конфигурирования. Для добавления дополнительных правил поддержки входных подключений, необходимо нажать **Edit** и добавить правила **Add Rule** для HTTP и HTTPS, после чего нажать **Save**.

5. Для проверки примененных правил, можно перейти в консоль управления экземплярами виртуальных машин, выбрать интересующую нас машину, и в нижней части информационной панели в группе Security

groups выбрать view inbound rules. Должна отобразиться панель, подтверждающая возможность подключения по портам 80, 22 и 443.



После установки необходимых правил, подключитесь к вашей виртуальной машине по SSH.

1.6 Шаг 5: Установка и запуск LAMP-сервера

1. Чтобы убедиться, что все ваши программные пакеты обновлены, выполните быстрое обновление программного обеспечения на вашей виртуальной машине. Этот процесс может занять несколько минут, но важно убедиться, что у вас есть последние обновления безопасности и исправления ошибок.

   Параметр `-y` позволяет установить обновления без запроса подтверждения. Если вы хотите изучить обновления перед установкой, вы можете опустить эту опцию.

   ```
   [ec2-user ~]$ sudo yum update -y
   ```

2. Установите репозитории light-mariadb10.2-php7.2 и php7.2 Amazon Linux Extras для получения последних версий пакетов LAMP MariaDB и PHP для Amazon Linux 2.

   ```
   [ec2-user ~]$ sudo amazon-linux-extras install -y lamp-mariadb10.2-php7.2 php7.2
   ```

   ВАЖНО

   Если вы получили сообщение об ошибке `sudo: amazon-linux-extras: command not found`, то ваш экземпляр не был запущен с Amazon Linux 2 AMI (возможно, вы используете Amazon Linux AMI). Вы можете просмотреть свою версию Amazon Linux со следующей командой.

   ```
   cat /etc/system-release
   ```

3. Теперь вы можете установить веб-сервер Apache, MariaDB и пакеты программного обеспечения PHP.

   Используйте команду **yum install** для одновременной установки нескольких пакетов программного обеспечения и всех связанных зависимостей.

```
[ec2-user ~]$ sudo yum install -y httpd mariadb-server
```

4. Запустите веб-сервер Apache.

```
[ec2-user ~]$ sudo systemctl start httpd
```

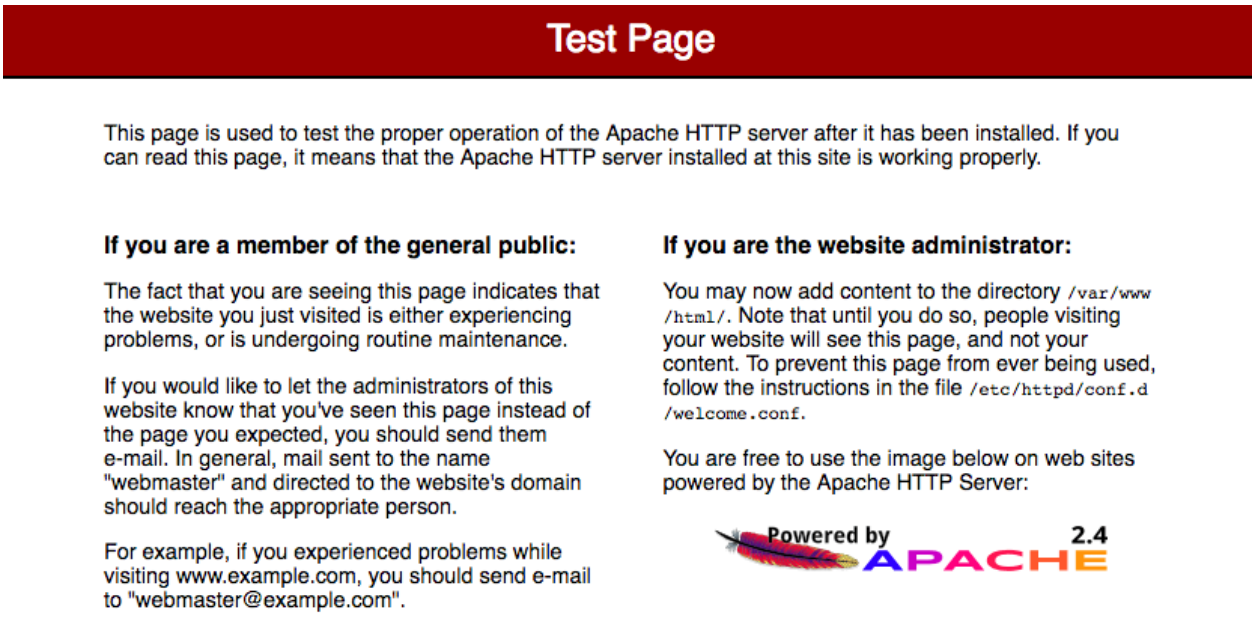
5. Используйте команду **systemctl** для настройки веб-сервера Apache на запуск при каждой загрузке системы.

```
[ec2-user ~]$ sudo systemctl enable httpd
```

   Вы можете проверить, что **httpd** включен, выполнив следующую команду:

```
[ec2-user ~]$ sudo systemctl is-enabled httpd
```

6. Добавьте правило безопасности, разрешающее входящие HTTP (порт 80) соединения к вашему экземпляру, если вы еще этого не сделали.
7. Проверьте работу вашего веб-сервера. В веб-браузере введите публичный DNS-адрес (или публичный IP-адрес) вашей виртуальной машины. Если в `/var/wwww/html` нет содержимого, вы должны увидеть тестовую страницу Apache. Вы можете получить публичный DNS для вашего примера с помощью консоли Amazon EC2.



8. Если вы не видите тестовую страницу Apache, убедитесь, что используемая вами группа безопасности содержит правило для разрешения траффика HTTP (порт 80).

Сервер Apache **httpd** предоставляет файлы, которые хранятся в каталоге «Apache document root». В Amazon Linux Apache это каталог `/var/wwww/html`, который по умолчанию принадлежит root.

Чтобы позволить учетной записи `ec2-user` управлять файлами в этом каталоге, вы должны изменить права собственности и права доступа к этому каталогу. Есть много способов выполнить эту задачу. В этом руководстве мы добавим пользователя `ec2-user` группу `apache`, чтобы дать право собственности группе `apache` на директорию `/var/www` и назначить права записи для этой группы.

Установка правил доступа

1. Добавьте своего пользователя (в данном случае `ec2-user`) в группу apache.

```
[ec2-user ~]$ sudo usermod -a -G apache ec2-user
```

2. Выйдите из системы, затем снова войдите в нее, чтобы забрать новую группу, а затем проверьте свое членство.
    a. Выход (можно использовать команду exit или закрыть окно терминала):

    ```
    [ec2-user ~]$ exit
    ```

    b. Чтобы проверить принадлежность к группе apache, переподключитесь к экземпляру, а затем выполните следующую команду:
    c.
    ```
    [ec2-user ~]$ groups
    ec2-user adm wheel apache systemd-journal
    ```

3. Измените принадлежность группы /var/www и ее содержимого на apache group.

```
[ec2-user ~]$ sudo chown -R ec2-user:apache /var/www
```

4. Чтобы добавить права на запись в группу и установить идентификатор группы для будущих подкаталогов, измените права доступа к каталогам /var/www и его подкаталогов.

```
[ec2-user ~]$ sudo chmod 2775 /var/www && find /var/www -type d -exec sudo
chmod 2775 {} \;
```

5. Чтобы добавить права на групповую запись, измените рекурсивно права доступа к файлам /var/www и его подкаталогов.:

```
[ec2-user ~]$ find /var/www -type f -exec sudo chmod 0664 {} \;
```

Теперь ec2-user (и все будущие члены группы apache) может добавлять, удалять и редактировать файлы в корне документа Apache, позволяя вам добавлять содержимое, например, статический веб-сайт или PHP-приложение.

1.7 Шаг 6: Проверка работы сервера LAMP

Если ваш сервер установлен и запущен, и ваши права доступа к файлам установлены правильно, ваша учетная запись ec2-user должна иметь возможность создать PHP файл в каталоге `/var/www/htm`, доступном из Интернета.

1. Создайте PHP-файл в корне документа Apache.

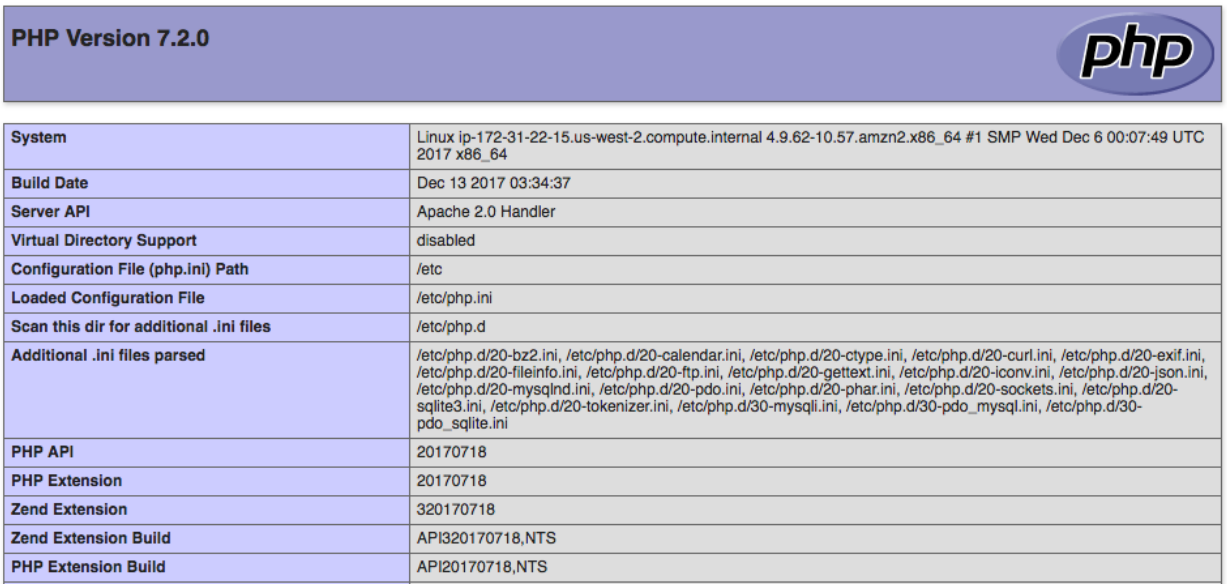
   ```
   [ec2-user ~]$ echo "<?php phpinfo(); ?>" > /var/www/html/phpinfo.php
   ```

   Если при попытке выполнить эту команду вы получите ошибку "Permission denied", попробуйте выйти из системы и войти в нее еще раз, чтобы получить соответствующие разрешения для настроенных вами групп.

2. Введите URL-адрес только что созданного файла в веб-браузере. Этот URL-адрес является публичным DNS-адресом вашей виртуальной машины, за которым следует обратная черта и имя файла. Например

   ```
   http://my.public.dns.amazonaws.com/phpinfo.php
   ```

   Должна появиться информация PHP:



PHP Version 7.2.0

System	Linux ip-172-31-22-15.us-west-2.compute.internal 4.9.62-10.57.amzn2.x86_64 #1 SMP Wed Dec 6 00:07:49 UTC 2017 x86_64
Build Date	Dec 13 2017 03:34:37
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc
Loaded Configuration File	/etc/php.ini
Scan this dir for additional .ini files	/etc/php.d
Additional .ini files parsed	/etc/php.d/20-bz2.ini, /etc/php.d/20-calendar.ini, /etc/php.d/20-ctype.ini, /etc/php.d/20-curl.ini, /etc/php.d/20-exif.ini, /etc/php.d/20-fileinfo.ini, /etc/php.d/20-ftp.ini, /etc/php.d/20-gettext.ini, /etc/php.d/20-iconv.ini, /etc/php.d/20-json.ini, /etc/php.d/20-mysqlnd.ini, /etc/php.d/20-pdo.ini, /etc/php.d/20-phar.ini, /etc/php.d/20-sockets.ini, /etc/php.d/20-sqlite3.ini, /etc/php.d/20-tokenizer.ini, /etc/php.d/30-mysqli.ini, /etc/php.d/30-pdo_mysql.ini, /etc/php.d/30-pdo_sqlite.ini
PHP API	20170718
PHP Extension	20170718
Zend Extension	320170718
Zend Extension Build	API320170718,NTS
PHP Extension Build	API20170718,NTS

ВАЖНО

Если вы не видите этой страницы, убедитесь, что файл `/var/www/html/phpinfo.php` был создан правильно на предыдущем этапе. Вы также можете проверить, что все необходимые пакеты были установлены с помощью следующей команды.

```
[ec2-user ~]$ sudo yum list installed httpd mariadb-server php-mysqlnd
```

Если один из требуемых пакетов не указан в выводах, установите их с помощью команды **sudo yum install** ***package***.

3. Удалите файл `phpinfo.php`. Хотя эта информация может быть полезной, она не должна передаваться в Интернет по соображениям безопасности.

```
[ec2-user ~]$ rm /var/www/html/phpinfo.php
```

Теперь у вас должен быть полностью функциональный веб-сервер LAMP. Если вы добавляете содержимое в корень документа Apache по адресу `/var/www/html`, вы должны иметь возможность просматривать это содержимое по публичному DNS адресу вашей виртуальной машины.

1.8 Шаг 7. Настройка сервера баз данных MariaDB

Стандартная установка сервера MariaDB имеет несколько особенностей, которые отлично подходят для тестирования и разработки, но они должны быть отключены или удалены для производственных серверов. Команда **mysql_secure_installation** проведет вас через процесс установки пароля root и удаления незащищенных функций из вашей установки. Даже если вы не планируете использовать сервер MariaDB, рекомендуется выполнить эту процедуру.

1. Запустите сервер MariaDB.

```
[ec2-user ~]$ sudo systemctl start mariadb
```

2. Запустите **mysql_secure_installation**.

```
[ec2-user ~]$ sudo mysql_secure_installation
```

   a. Когда появится запрос, введите пароль для учетной записи root.
      i. Введите текущий пароль root. По умолчанию учетная запись root не имеет установленного пароля. Нажмите Enter.
      ii. Введите Y, чтобы установить пароль, и дважды введите надежный пароль.
   b. Введите Y, чтобы удалить анонимные учетные записи пользователей.

c. Введите Y, чтобы отключить удаленный вход в систему.
d. Введите Y, чтобы удалить тестовую базу данных.
e. Введите Y, чтобы перезагрузить таблицы привилегий и сохранить изменения.

3. (Необязательно) Если вы не планируете использовать сервер MariaDB сразу же, остановите его. Вы можете перезапустить его, когда понадобится.

```
[ec2-user ~]$ sudo systemctl stop mariadb
```

4. (Необязательно) Если вы хотите, чтобы сервер MariaDB запускался при каждой загрузке, введите следующую команду.

```
[ec2-user ~]$ sudo systemctl enable mariadb
```

1.9 Шаг 8. Установка phpMyAdmin

phpMyAdmin - это веб-инструмент управления базами данных, который вы можете использовать для просмотра и редактирования баз данных MySQL на вашем экземпляре Amzon EC2. Не рекомендуется использование phpMyAdmin для доступа к LAMP серверу, если вы не включили поддержку SSL/TLS в веб-сервере Apache; иначе пароль администратора базы данных и другие данные будут небезопасно переданы через Интернет. Рекомендации по безопасности от разработчиков доступны по ссылке: https://docs.phpmyadmin.net/en/latest/setup.html#securing-your-phpmyadmin-installation

1. Установите необходимые зависимости

```
[ec2-user ~]$ sudo yum install php-mbstring -y
```

2. Перезагрузите Apache

```
[ec2-user ~]$ sudo systemctl restart httpd
```

3. Перезапустите `php-fpm`.

```
[ec2-user ~]$ sudo systemctl restart php-fpm
```

4. Перейдите к корню документов Apache `/var/www/html`.

```
[ec2-user ~]$ cd /var/www/html
```

5. Выберите пакет исходных текстов для последней версии phpMyAdmin на сайте https://www.phpmyadmin.net/downloads Чтобы загрузить файл непосредственно в вашу виртуальную машину, скопируйте ссылку и вставьте ее в команду wget, как в данном примере:

```
[ec2-user html]$ wget https://www.phpmyadmin.net/downloads/phpMyAdmin-latest-all-languages.tar.gz
```

6. Создайте каталог `phpMyAdmin` folder и разархивируйте в него содержимое пакета.

```
[ec2-user html]$ mkdir phpMyAdmin && tar -xvzf phpMyAdmin-latest-all-languages.tar.gz -C phpMyAdmin --strip-components 1
```

7. Удалите *phpMyAdmin-latest-all-languages.tar.gz*.

```
[ec2-user html]$ rm phpMyAdmin-latest-all-languages.tar.gz
```

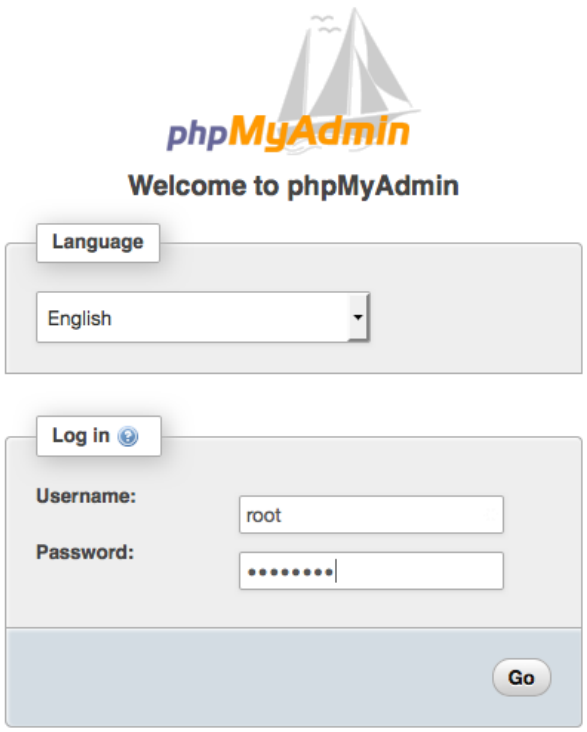
8. (Необязательно) Если MariaDB сервер не запущен, запустите его сейчас

```
[ec2-user ~]$ sudo systemctl start mariadb
```

9. В веб-браузере введите URL-адрес вашей установки phpMyAdmin. Этот URL-адрес является публичным DNS-адресом (или публичным IP-адресом) вашего экземпляра, за которым следует обратная черта и имя вашего каталога установки. Например:

```
http://my.public.dns.amazonaws.com/phpMyAdmin
```

Вы должны увидеть страницу входа в систему phpMyAdmin:



10. Войдите в систему phpMyAdmin с именем пользователя root и корневым паролем MySQL, который вы создали ранее.

Чтобы настроить phpMyAdmin, вы можете вручную создать конфигурационный файл или использовать консоль настройки.

1.10 Шаг 9. Задание на самостоятельную работу

На основе развернутой установки LAMP предлагается самостоятельно развернуть на подготовленной машине экземпляр платформы WordPress.

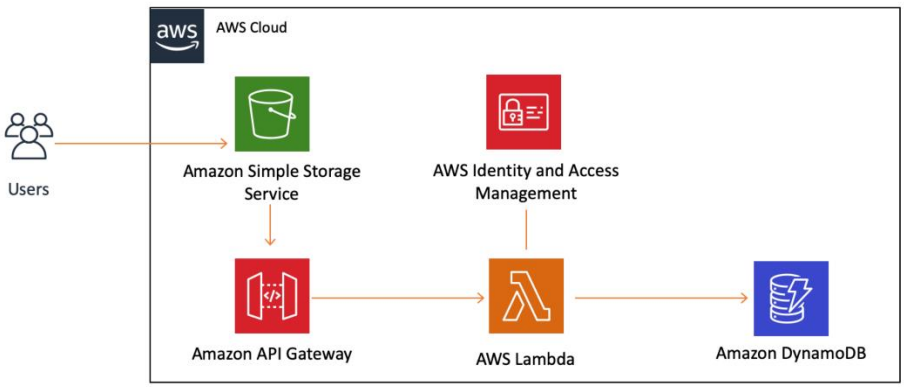
2 Развертывание статического приложения на основе инструментов Amazon Web Services

Данная инструкция подготовлена на основе материалов официального сайта Amazon Web Services доступной по ссылке: https://aws.amazon.com/ru/getting-started/hands-on/build-web-app-s3-lambda-api-gateway-dynamodb/. Для ее реализации вам необходим аккаунт с разрешенным доступом к следующим сервисам AWS:

- Amazon Simple Storage Service (S3)
- AWS Lambda
- API Gateway
- Amazon DynamoDB
- AWS Identity and Access Management

В ходе выполнения этого задания вы создадите простое интернет-приложение. Для начала мы создадим статический веб-сайт типа Hello World. Затем вы узнаете, как расширить функциональность веб-сайта, чтобы на нем отображался необходимый пользователю текст.

На схеме ниже наглядно представлены сервисы, используемые в данном обучающем курсе, и то, как они связаны. Как видно на рисунке ниже, в этом приложении используются сервисы Amazon Simple Storage Service (S3), Amazon API Gateway, AWS Lambda и Amazon DynamoDB.



В процессе обучения мы подробно обсудим сервисы и укажем ресурсы, которые помогут вам познакомиться с ними поближе.

2.1 Шаг 1. Создание статического веб-сайта

На этом шаге мы настроим сервис Amazon Simple Storage Service (S3), чтобы разместить в нем статические ресурсы для своего интернет-приложения. В следующих шагах мы добавим к этим страницам динамические функции, используя AWS Lambda и Amazon API Gateway для вызова удаленных RESTful API.

Для нашего примера мы будем использовать предоставленный сервисом Amazon S3 URL-адрес веб-сайта. Он будет указан в форме:

```
http://{your-bucket-name}.s3-website.{region}.amazonaws.com
```

Весь ваш статический интернет-контент, включая HTML, CSS, JavaScript, изображения и другие файлы, будет храниться в сервисе Amazon S3. Мы выбрали сервис S3, так как он предоставляет хранилище объектов, в котором можно непосредственно хранить и извлекать файлы. Конечные пользователи смогут легко получать доступ к веб-сайту, используя URL-адрес объекта, предоставляемый сервисом S3.

Если вас беспокоит, что придется работать с таким большим количеством компонентов, не волнуйтесь! В этом модуле не будут использоваться другие сервисы AWS, а также не потребуется запускать никакие веб-серверы («сервер» – это программное или аппаратное устройство, которое принимает запросы по сети и отвечает на них), чтобы сделать веб-сайт доступным.

Наш веб-сайт будет крайне простой страницей Hello World, а в дальнейших модулях мы добавим другие функции.

Для работы большинства реальных приложений вам потребуется использовать пользовательский домен, чтобы разместить свой сайт. («Пользовательский домен» – это уникальное фирменное имя, идентифицирующее веб-сайт, например www.amazon.com. Хотя эти темы не рассматриваются в данном модуле, если вас интересует использование собственного домена, вы можете следовать инструкциям из этого руководства).

2.1.1 Основные понятия

Статический веб-сайт. Статический веб-сайт содержит фиксированный контент, в отличие от динамических веб-сайтов. Статические веб-сайты – это самый

простой тип веб-сайтов, который проще всего создать. Достаточно создать несколько страниц HTML и опубликовать их на веб-сервере!

Ресурс – объект, с которым можно работать (например, корзина Amazon S3).

Хранилище объектов – архитектура хранения данных на компьютерах, которая управляет данными в виде объектов, в отличие от других архитектур хранилищ, где данные хранятся в виде файлов или блоков.

Объекты S3 – ресурс, который содержит данные и метаданные (набор данных, которые описывают другие данные и предоставляют информацию о них) и связан с уникальным ключом, идентифицирующим его.

Корзина S3 – ресурс, используемый для группирования множества хранимых объектов. Его имя должно быть уникальным в глобальном масштабе.

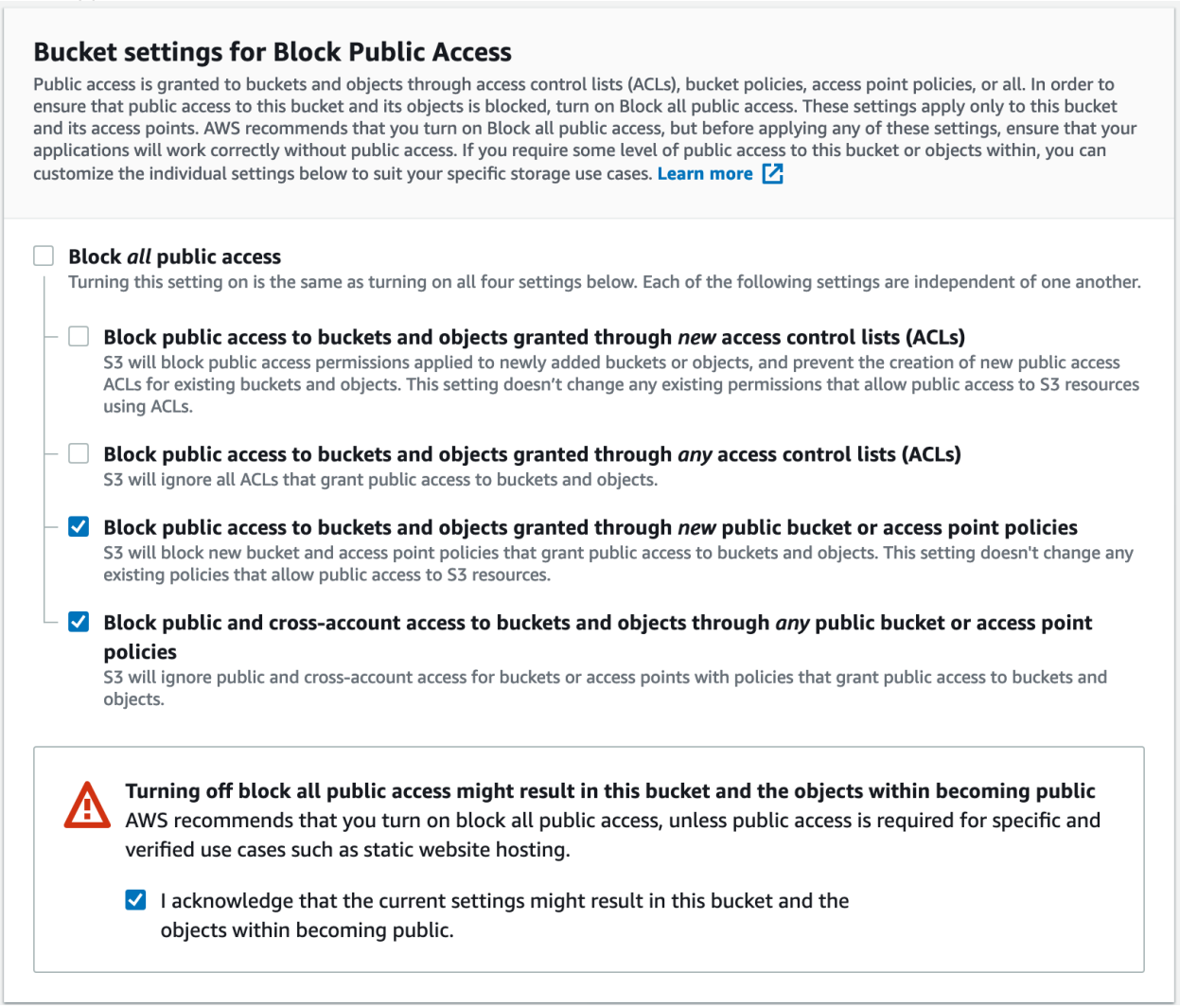
URL-адрес объекта S3 – доступный из Интернета уникальный адрес, предоставляемый сервисом S3 для хранящегося в нем объекта.

Регионы AWS – отдельные географические области, которые AWS использует для размещения своей инфраструктуры. Они распределены по всему миру, чтобы клиенты могли выбрать ближайший регион для размещения своей облачной инфраструктуры.

2.1.2 Создание корзины S3 с публичным доступом

1. Используя данные для доступа AWS, войдите в консоль Amazon S3.
2. Нажмите синюю кнопку Create bucket (Создать корзину).
3. Дайте корзине какое-нибудь имя. Все корзины Amazon S3, независимо от того, в каком аккаунте они созданы, используют одно и то же пространство имен, поэтому это имя должно быть уникальным.
4. Выберите регион, в котором будет располагаться корзина. Если вы используете учебный аккаунт AWS Educate, то выберете «**N. Virginia**»
5. Снимите флажок Block *all* public access (Полностью блокировать публичный доступ). Так как мы размещаем на хостинге веб-сайт, люди должны иметь доступ к нему.
6. Установите флажок Block public access to buckets and objects granted through new public bucket or access point policies (Блокировать публичный доступ к корзинам и объектам, предоставленный в рамках новых правил для публичных корзин или точек доступа).
7. Установите флажок Block public and cross-account access to buckets and objects through any public bucket or access point policies (Блокировать публичный доступ и доступ из других аккаунтов к корзинам и объектам по любым правилам для публичных корзин или точек доступа).
8. Установите флажок, подтверждающий, что содержимое корзины может стать общедоступным. Мы не будем хранить никаких конфиденциальных данных в этой

корзине, поскольку ее содержимое может стать общедоступным. Вот как это должно выглядеть:

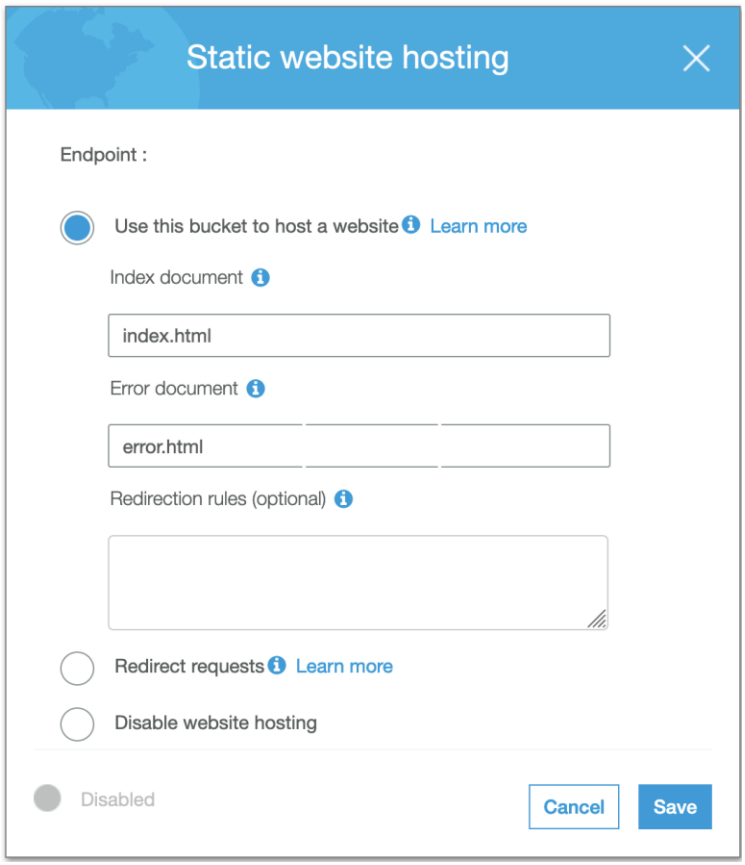


9. Нажмите синюю кнопку Create bucket (Создать корзину).

10. В верхней части экрана должно отобразиться зеленое окно с сообщением: Successfully created bucket (Корзина успешно создана).

2.1.3 Настройка корзины для размещения веб-сайта

1. Найдите свою новую корзину в списке ниже и щелкните ее.
2. Щелкните **серую** вкладку Properties (Свойства).
3. Щелкните поле Static website hosting (Размещение статического веб-сайта).
4. Щелкните переключатель Use this bucket to host a website (Использовать эту корзину для размещения веб-сайта).
5. В поле Index Document (Начальный документ) введите **index.html**.
6. В поле Error Document (Документ ошибки) введите **error.html**. Вот как это должно выглядеть:



7. Нажмите **синюю** кнопку Save (Сохранить).

8. Нажмите **серую** кнопку Overview (Обзор).

9. Не закрывая браузер, откройте на своем компьютере удобный для вас текстовый редактор. Создайте новый файл и вставьте в него приведенный ниже HTML-код.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Hello World</title>
</head>

<body>
    Hello World
</body>
</html>
```

10. Сохраните файл под именем *index.html*.

11. Создайте еще один текстовый файл и вставьте в него следующее…

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Error Page</title>
</head>

<body>
    Error Page
</body>
```

```
</html>
```

12. Сохраните второй файл под именем *error.html*.

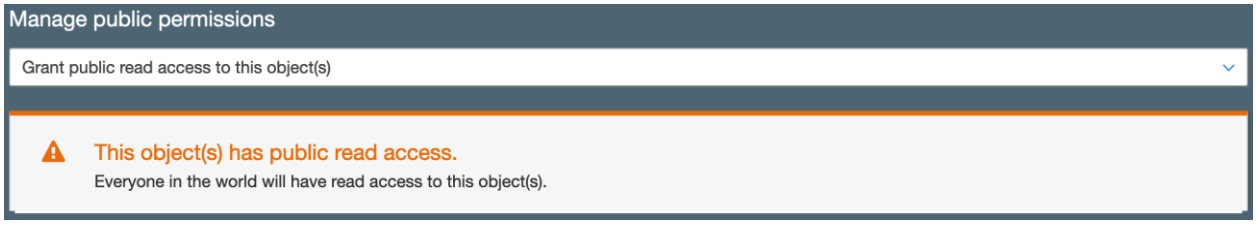
13. Перейдите обратно в браузер.

14. Нажмите **синюю** кнопку Upload (Загрузить).

15. Выберите для загрузки файлы *index.html* и *error.html*.

16. Нажмите **синюю** кнопку Next (Далее).

17. В раскрывающемся меню Manage public permissions (Управление публичными разрешениями) выберите пункт Grant public read access to this object(s) (Предоставить публичный доступ для чтения к объектам). Это необходимо, чтобы любой пользователь мог открыть сайт в своем браузере, в том числе и мы при тестировании.

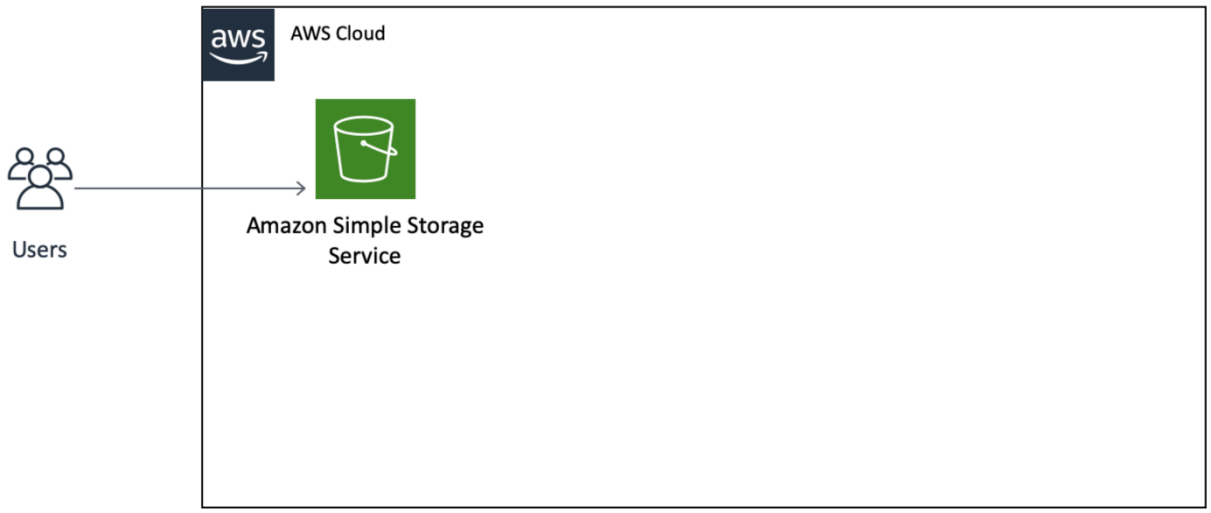


18. Нажмите синюю кнопку Upload (Загрузить) слева.

2.1.4 Тестирование статического веб-сайта

1. Щелкните в списке пункт *index.html*.
2. **Щелкните ссылку** в области Object URL (URL-адрес объекта).
3. Должна загрузиться новая вкладка браузера, на которой отобразится собственно базовая веб-страница Hello World.
4. Поздравляем, вы разместили статический веб-сайт!

Сейчас наша архитектура выглядит так:



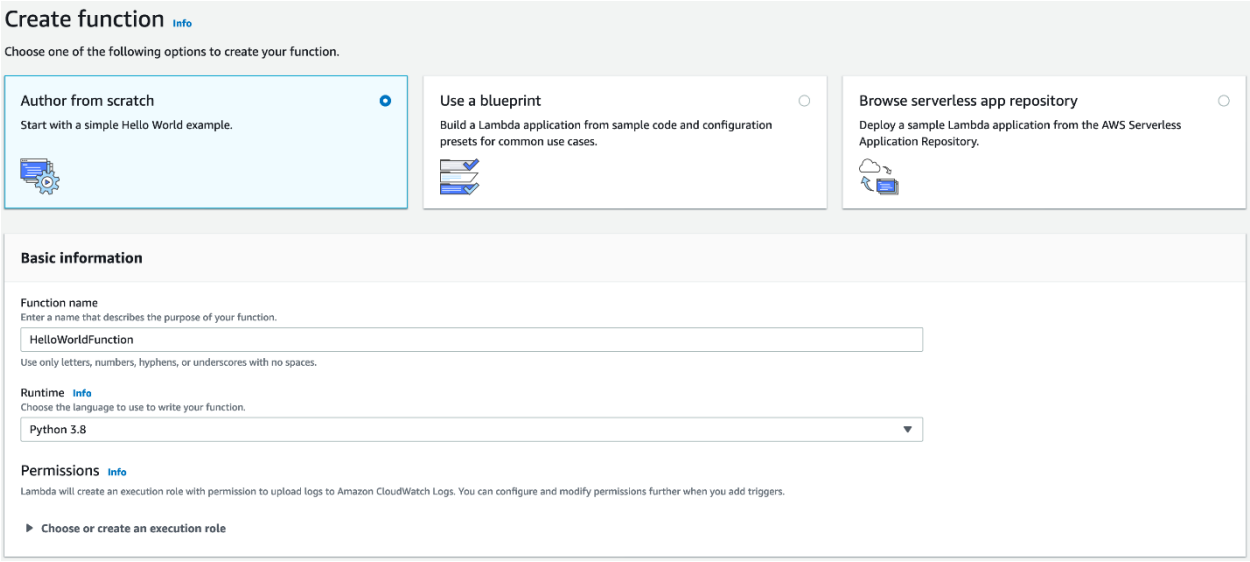
Пока что она крайне проста, ведь вы используем только сервис S3. Мы убедились, что пользователи могут получить доступ к интернет-приложению, хранящемуся в S3. Затем мы создадим функцию AWS Lambda.

2.2 Шаг 2. Создание бессерверной функции

В этом модуле вы напишете небольшой код на языке Python который в последующих модулях понадобится для добавления интерактивных элементов на вашу веб-страницу. Мы воспользуемся AWS Lambda, вычислительным сервисом, который позволяет создавать бессерверные функции. («Бессерверные» функции дают возможность разработчику не задумываться над управлением программным и аппаратным обеспечением. Вместо этого приложения разделяются на конкретные функции, которые можно вызывать и масштабировать по отдельности.)

Эти бессерверные функции запускаются при конкретных событиях, указанных вами в коде. Кроме того, это очень выгодный сервис, поскольку вы платите только за количество обработанных событий, а не за время простоя. Более того, вам не нужно заботиться об управлении серверами!

1. В новой вкладке браузера войдите в консоль AWS Lambda.
2. Убедитесь, **что знаете, для какого региона создаете функцию**. Регион указан в самом верху страницы, рядом с именем аккаунта.
3. Нажмите оранжевую кнопку Create Function (Создать функцию).
4. В поле Function Name (*Имя функции*) введите *HelloWorldFunction*.
5. В раскрывающемся списке среды исполнения выберите **Python 3.8**.



6. Нажмите оранжевую кнопку Create Function (Создать функцию).

7. В верхней части экрана должно отобразиться окно зеленого цвета с сообщением: «Successfully created the function» (Функция успешно создана).

8. Замените код в области Function Code (Код функции) следующим:

```
# import the JSON utility package since we will be working with a JSON object
import json
# define the handler function that the Lambda service will use an entry point
def lambda_handler(event, context):
# extract values from the event object we got from the Lambda service
    name = event['firstName'] +' '+ event['lastName']
# return a properly formatted JSON object
    return {
    'statusCode': 200,
    'body': json.dumps('Hello from Lambda, ' + name)
    }
```

9. Нажмите оранжевую кнопку Save (Сохранить) в верхней части экрана.

10. Давайте протестируем нашу новую функцию. Щелкните ссылку Select a test event (Выбрать тестовое событие) в верхней части экрана.

11. В раскрывающемся меню щелкните ссылку Configure test events (Настроить тестовые события).

12. В поле Event Name (Имя события) введите *HelloWorldTestEvent*.

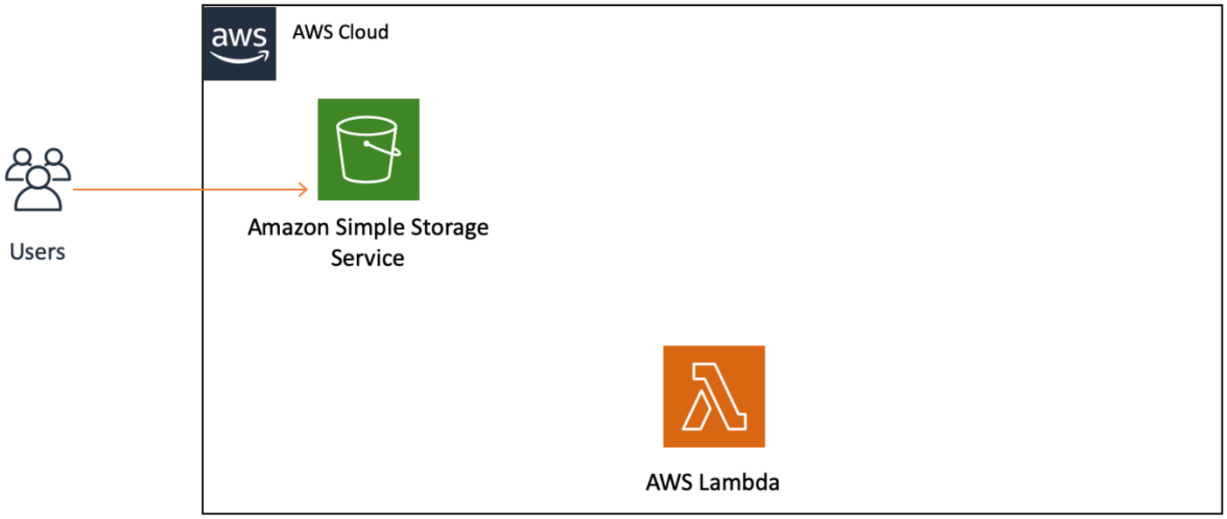
13. Скопируйте и вставьте следующий объект JSON, заменив им объект по умолчанию:

```
{
"firstName": "Ada",
"lastName": "Lovelace"
}
```

14. Нажмите оранжевую кнопку Create (Создать) в нижней части страницы.
15. Нажмите **серую** кнопку Test (Тестировать) в верхней части страницы.
16. В верхней части страницы отобразится окно **светло-зеленого** цвета с сообщением: «Execution result: succeeded» (Результат выполнения: успешно выполнено). Если щелкнуть ссылку Details (Сведения), можно просмотреть событие, которое вернула функция.

17. Все готово! Теперь у вас есть работающая функция Lambda.

По завершении данного шага наша архитектура буде выглядеть следующим образом:



Вы видите, что мы добавили сервис AWS Lambda, но он еще не подключен к хранилищу S3 или нашим пользователям. Мы займемся этим в следующем шаге.

2.3 Шаг 3. Привязка бессерверной функции к веб-сайту

В этом шаге мы будем использовать Amazon API Gateway для создания RESTful API, с помощью которого можно будет вызывать функцию Lambda из веб-клиента (как правило, это веб-браузер пользователя). API Gateway будет работать как промежуточный уровень между клиентом HTML, созданным в модуле 1, и бессерверным внутренним компонентом, созданным в модуле 2.

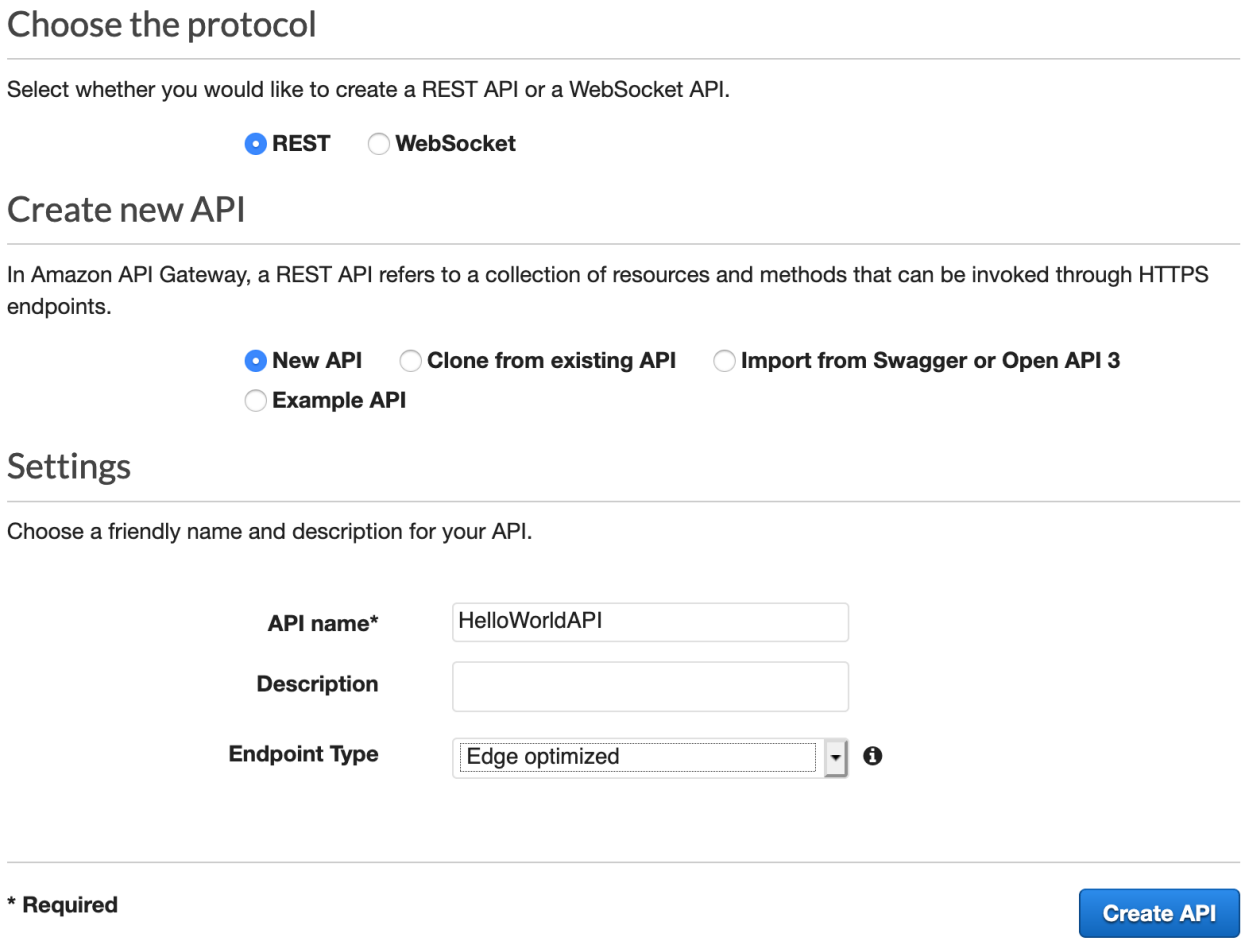
Для реализации этого шага вам необходимо знать о том, что такое CORS. Механизм **CORS (Cross Origin Resource Sharing)** использует заголовки HTTP, чтобы браузер разрешил интернет-приложению выполняться в одном источнике (домене) и мог получать доступ к выбранным ресурсам с сервера в другом источнике.

2.3.1 Создание нового REST API

1. Войдите в консоль API Gateway.
2. Нажмите оранжевую кнопку Create API (Создать API).
3. Найдите поле REST API и нажмите в нем оранжевую кнопку Build (Собрать).
4. В поле Choose the protocol (Выбор протокола) выберите REST.
5. В поле Create new API (Создать новый API) выберите New API (Новый API).
6. В поле API name (*Имя API*) введите *HelloWorldAPI*.
7. Выберите пункт Edge optimized (Оптимизированный для периферии) в раскрывающемся списке Endpoint Type (Тип адреса). Обратите внимание, что адреса, оптимизированные для периферии, лучше всего подходят для географически распределенных клиентов. Это

делает их хорошим вариантом для публичных сервисов, доступных через Интернет. Как правило, региональные адреса используются для интерфейсов API, доступ к которым осуществляется преимущественно из одного и того же региона AWS.

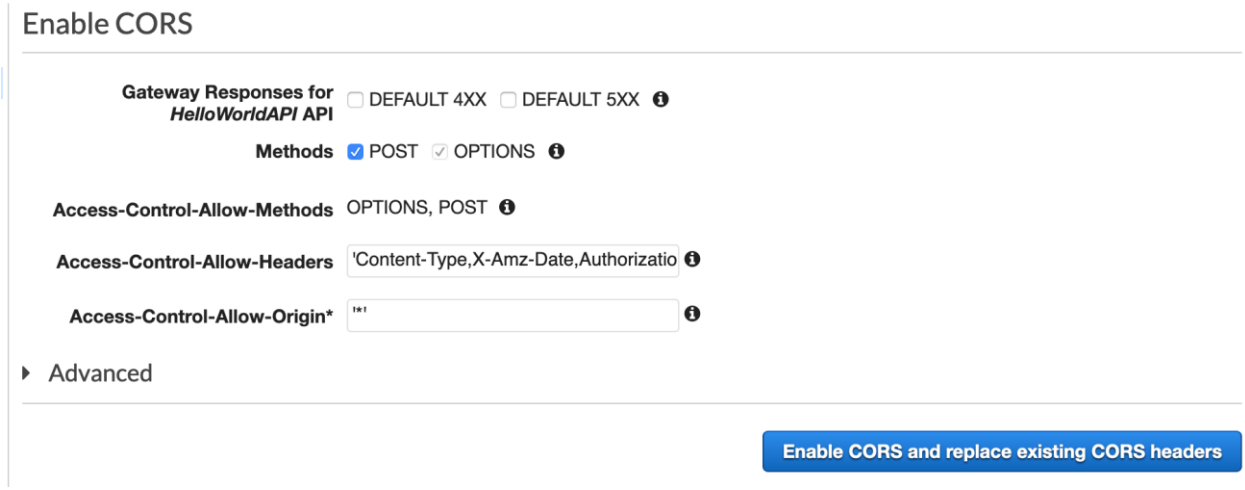
8. Нажмите синюю кнопку Create API (Создать API). Параметры должны выглядеть примерно так, как на следующем снимке экрана:



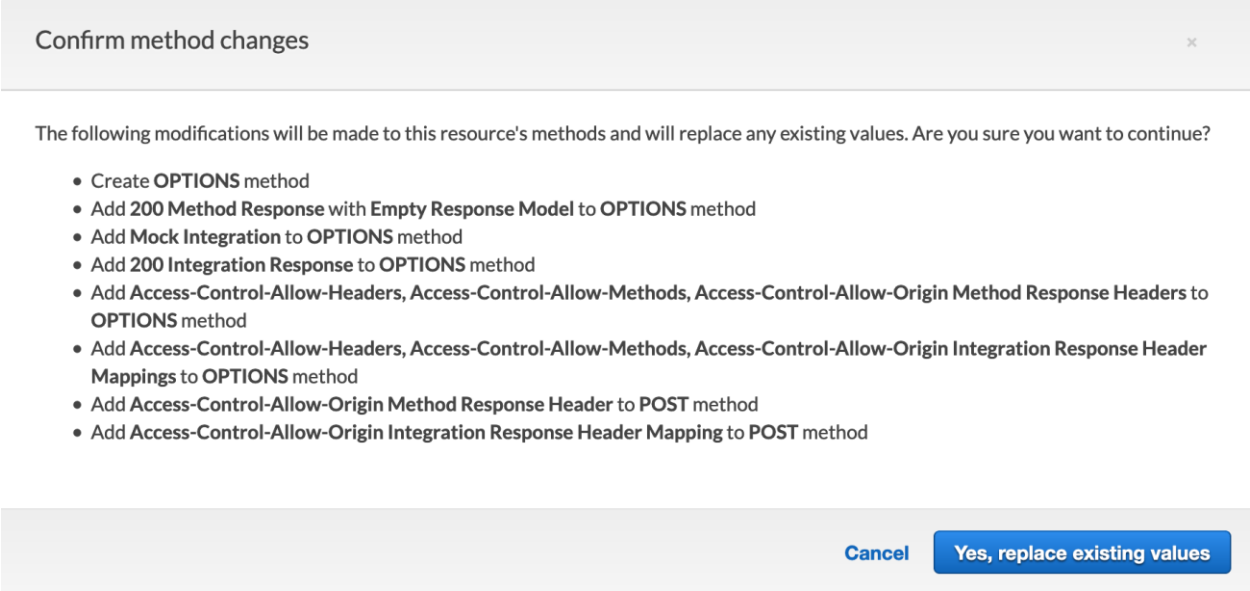
2.3.2 Создание нового ресурса и метода

1. В меню навигации слева щелкните Resources (Ресурсы) под API HelloWorld.
2. Выбрав ресурс «/», нажмите Create Method (Создать метод) в раскрывающемся меню Action (Действие).
3. Выберите POST в появившемся раскрывающемся списке, а затем установите флажок.
4. Выберите значение Lambda Function (Функция Lambda) в качестве типа интеграции.
5. Введите *HelloWorldFunction* в поле Function (Функция).
6. Нажмите синюю кнопку Save (Сохранить).
7. Должно появиться сообщение о том, что вы предоставляете создаваемому интерфейсу API разрешение на вызов функции Lambda. Нажмите кнопку ОК.
8. Выбрав созданный метод POST, нажмите Enable CORS (Включить CORS) в раскрывающемся меню Action (Действие).

9. Оставьте флажок POST установленным и нажмите синюю кнопку Enable CORS and replace existing CORS headers (Включить CORS и заменить существующие заголовки CORS).



10. Должно появиться сообщение с предложением подтвердить изменения метода. Нажмите синюю кнопку Yes, replace existing values (Да, заменить существующие значения).



2.3.3 Развертывание и тестирование API

1. В раскрывающемся списке Actions (Действия) выберите пункт Deploy API (Развертывание API).

2. Выберите пункт [New Stage] (Новый этап) в раскрывающемся списке Deployment stage (Этап развертывания).

3. В качестве значения параметра Stage Name (Имя этапа) введите *dev*.

4. Выберите Deploy (Развернуть).

5. Скопируйте и сохраните URL-адрес из поля Invoke URL (URL-адрес для вызова). Он потребуется в модуле 5.

6. В левой области навигации нажмите Resources (Ресурсы).

7. Справа появится список методов для нашего API. Выберите метод POST.

8. Нажмите маленький синий значок молнии.

9. Вставьте в поле Request Body (Текст запроса) следующий текст:

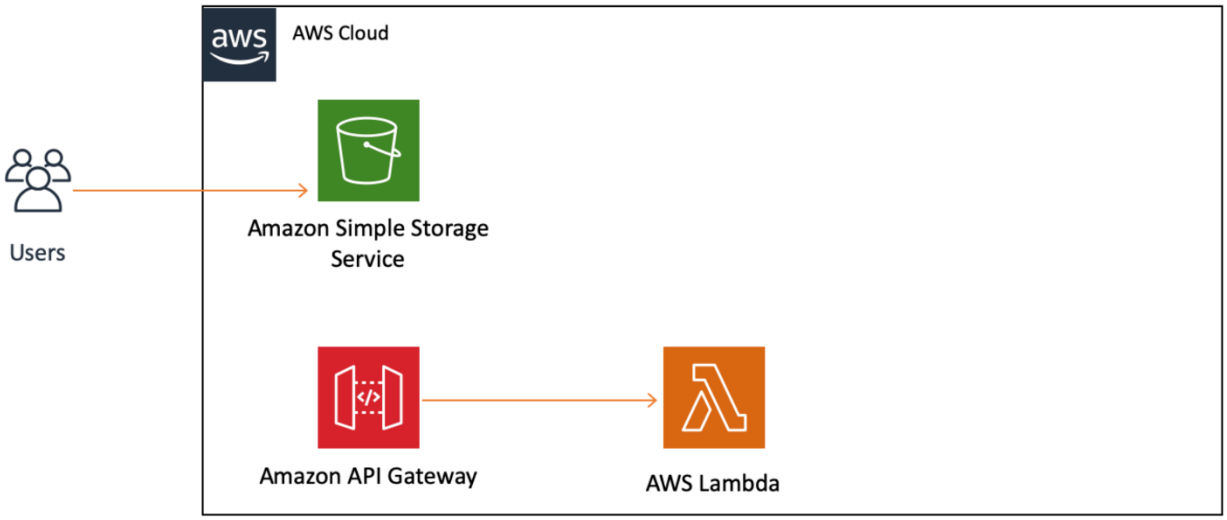
```
{
    "firstName":"Grace",
    "lastName":"Hopper"
}
```

5. Нажмите синюю кнопку Test (Тестировать).

6. С правой стороны должен отобразиться ответ с кодом 200.

7. Отлично! Мы создали и протестировали API для вызова нашей функции Lambda.

Шаг 3 завершен. Пришло время проверить нашу архитектуру:



Мы добавили API Gateway и подключили его к существующей функции Lambda. Теперь мы можем активировать нашу функцию с помощью вызова API. Мы по-прежнему не можем совершать этот вызов из нашего веб-клиента. Сначала мы добавим нашу таблицу данных на шаге 4, а затем соединим все компоненты на шаге 5.

2.4 Шаг 4. Создание таблицы данных

В этом модуле мы создадим таблицу для хранения данных с помощью Amazon DynamoDB. DynamoDB – это сервис баз данных типа «ключ-значение», поэтому нам не нужно создавать схему для своих данных. Он обеспечивает стабильную работу в любых масштабах, и при его использовании не требуется управлять никакими серверами.

Кроме того, мы будем использовать сервис AWS Identity and Access Management (IAM), чтобы безопасно предоставлять нашим сервисам необходимые разрешения для взаимодействия друг с другом. В частности, мы разрешим функции Lambda, созданной на шаге 2, записывать данные в новую таблицу

DynamoDB, используя политику IAM. Для этого мы воспользуемся пакетом SDK AWS (Python, JavaScript или Java) из нашей функции Lambda.

2.4.1 Создание таблицы Amazon DynamoDB

1. Войдите в консоль Amazon DynamoDB.
2. Убедитесь, что знаете, для какого региона создаете функцию. Регион указан в самом верху страницы, рядом с именем аккаунта.
3. Нажмите синюю кнопку Create table (Создать таблицу).
4. В поле Table name (Имя таблицы) введите *HelloWorldDatabase*.
5. В поле Primary Key (Первичный ключ) введите *id*.
6. Нажмите синюю кнопку Create (Создать).
7. Скопируйте значение Amazon Resource Name (ARN) для таблицы с правой панели (оно потребуется позже в этом модуле).

2.4.1 Создание политики IAM и ее добавление к функции Lambda

1. Теперь, когда у нас есть таблица, давайте изменим функцию Lambda, чтобы в нее можно было записывать данные. В *новом* окне браузера откройте консоль AWS Lambda.
2. Щелкните функцию, созданную в модуле 2 (если вы использовали наши примеры, она будет называться HelloWorldFunction).
3. Мы будем добавлять разрешения к функции, чтобы она могла использовать сервис DynamoDB, с помощью сервиса AWS Identity and Access Management (IAM).
4. Откройте вкладку Permissions (Разрешения).
5. В поле Execution role (Роль для выполнения) щелкните нужную роль. Откроется новая вкладка браузера.
6. Нажмите Add inline policy (Добавить встроенную политику) справа от поля Permissions policies (Политики разрешений).
7. Откройте вкладку «JSON».
8. Вставьте следующую политику в области текста, заменив ARN таблицы в поле Resource (Ресурс) в строке 15:

```
{
"Version": "2012-10-17",
"Statement": [
    {
        "Sid": "VisualEditor0",
        "Effect": "Allow",
        "Action": [
            "dynamodb:PutItem",
            "dynamodb:DeleteItem",
            "dynamodb:GetItem",
            "dynamodb:Scan",
            "dynamodb:Query",
            "dynamodb:UpdateItem"
        ],
        "Resource": "YOUR-TABLE-ARN"
    }
    ]
```

```
}
```

9. Эта политика разрешит нашей функции Lambda считывать, редактировать и удалять элементы, но только в созданной нами таблице.

10. Нажмите синюю кнопку Review Policy (Просмотреть политику).

11. В поле Name (Имя) введите *HelloWorldDynamoPolicy*.

12. Нажмите синюю кнопку Create Policy (Создать политику).

13. Теперь вы можете закрыть эту вкладку браузера и вернуться на вкладку функции Lambda.

2.4.2 Изменение функции Lambda для записи в таблицу DynamoDB

1. Откройте вкладку Configuration (Настройка).
2. Замените код функции следующим кодом:

```
# import the json utility package since we will be working with a JSON object
import json
# import the AWS SDK (for Python the package name is boto3)
import boto3
# import two packages to help us with dates and date formatting
from time import gmtime, strftime

# create a DynamoDB object using the AWS SDK
dynamodb = boto3.resource('dynamodb')
# use the DynamoDB object to select our table
table = dynamodb.Table('HelloWorldDatabase')
# store the current time in a human readable format in a variable
now = strftime("%a, %d %b %Y %H:%M:%S +0000", gmtime())

# define the handler function that the Lambda service will use as an entry point
def lambda_handler(event, context):
# extract values from the event object we got from the Lambda service and store in
a variable
    name = event['firstName'] +' '+ event['lastName']
# write name and time to the DynamoDB table using the object we instantiated and
save response in a variable
    response = table.put_item(
        Item={
            'ID': name,
            'LatestGreetingTime':now
            })
# return a properly formatted JSON object
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda, ' + name)
    }
```

3. Нажмите оранжевую кнопку Save (Сохранить) в верхней части экрана.

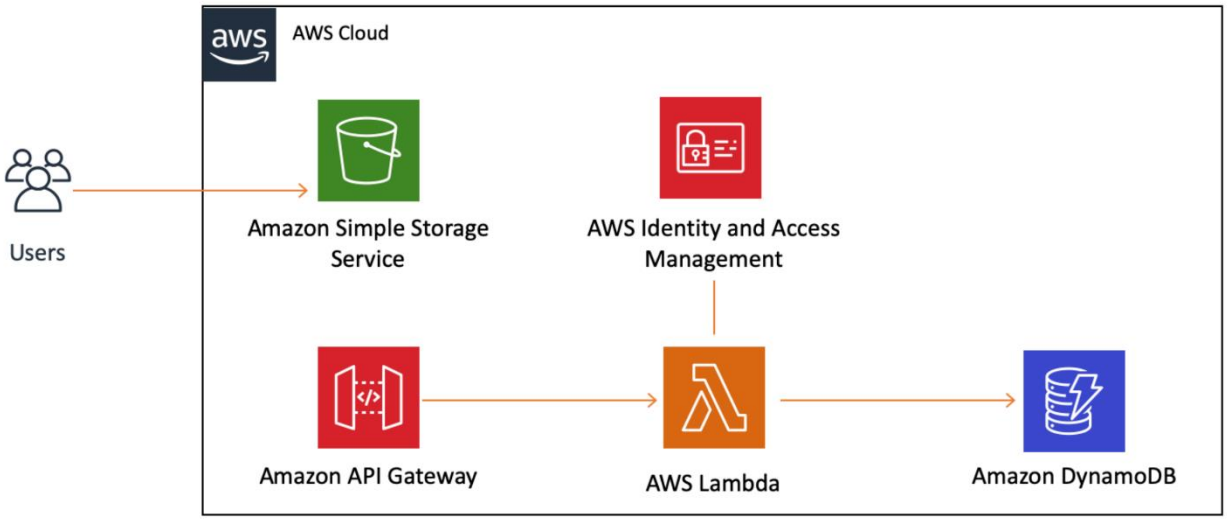
2.4.3 Тестирование изменений

1. Нажмите белую кнопку Test (Тестировать).
2. Должно появиться сообщение Execution result: succeeded (Результат выполнения: успешно) на зеленом фоне.

> Если выполнение теста происходит с ошибкой, то проверьте совпадает ли имя первичного ключа, которое вы указали при создании таблицы (п. 5 раздела 2.4.1) и то имя, что указано в коде Lambda-функции (строка 22). Имена должны совпадать с точностью до регистра. Если они не совпадают, внесите корректировки в код Lambda-функции.

3. На *новой* вкладке браузера откройте консоль DynamoDB.
4. Щелкните элемент Tables (Таблицы) на панели навигации слева.
5. Выберите таблицу *HelloWorldDatabase*, созданную ранее в этом модуле.
6. Откройте вкладку Items (Элементы) справа.
7. На ней должны отображаться элементы, соответствующие тесту. Если вы использовали наши примеры, то должен отображаться идентификатор элемента «Ada Lovelace».
8. При каждом выполнении функции Lambda таблица DynamoDB будет обновляться. Если используется такое же имя, изменится только отметка времени.

Теперь, когда шаг 4 завершен, давайте рассмотрим нашу текущую архитектуру:



В этом модуле мы добавили два сервиса: DynamoDB (для хранения данных) и IAM (для безопасного управления разрешениями). Оба сервиса подключены к нашей функции Lambda, чтобы она могла выполнять запись в нашей базе данных. Остался последний шаг – добавление кода для вызова API Gateway в клиент.

2.5 Шаг 5. Добавление интерактивных элементов на веб-сайт

В рамках этого шага мы обновим статический веб-сайт, созданный на шаге 1, чтобы можно было вызывать интерфейс REST API, который мы создали на шаге 3. Это даст возможность отображать на сайте текст, который вводит пользователь.

2.5.1 Обновление HTML-файла, сохраненного в сервисе S3

1. Откройте файл *index.html*, который вы создали в модуле 1.

2. Замените имеющийся в нем код следующим:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Hello World</title>
    <!-- Add some CSS to change client UI -->
    <style>
    body {
        background-color: #232F3E;
        }
    label, button {
        color: #FF9900;
        font-family: Arial, Helvetica, sans-serif;
        font-size: 20px;
        margin-left: 40px;
        }
     input {
        color: #232F3E;
        font-family: Arial, Helvetica, sans-serif;
        font-size: 20px;
        margin-left: 20px;
        }
    </style>
    <script>
        // define the callAPI function that takes a first name and last name as
parameters
        var callAPI = (firstName,lastName)=>{
            // instantiate a headers object
            var myHeaders = new Headers();
            // add content type header to object
            myHeaders.append("Content-Type", "application/json");
            // using built in JSON utility package turn object to string and store
in a variable
            var raw = JSON.stringify({"firstName":firstName,"lastName":lastName});
            // create a JSON object with parameters for API call and store in a
variable
            var requestOptions = {
                method: 'POST',
                headers: myHeaders,
                body: raw,
                redirect: 'follow'
            };
            // make API call with parameters and use promises to get response
            fetch("YOUR-API-INVOKE-URL", requestOptions)
            .then(response => response.text())
            .then(result => alert(JSON.parse(result).body))
            .catch(error => console.log('error', error));
        }
    </script>
</head>
<body>
    <form>
        <label>First Name :</label>
        <input type="text" id="fName">
        <label>Last Name :</label>
```

```
        <input type="text" id="lName">
        <!-- set button onClick method to call function we defined passing input
values as parameters -->
        <button type="button"
onclick="callAPI(document.getElementById('fName').value,document.getElementById('l
Name').value)">Call API</button>
    </form>
</body>
</html>
```

3. Убедитесь, что в строку 41 добавлен URL вызова API (из модуля 3). ПРИМЕЧАНИЕ. Если у вас нет URL вашего API, его можно посмотреть в консоли API Gateway, выбрав нужный API, а затем щелкнув панель Stages (Этапы).

4. Сохраните файл.

5. Откройте консоль S3.

6. Щелкните корзину, созданную в модуле 1.

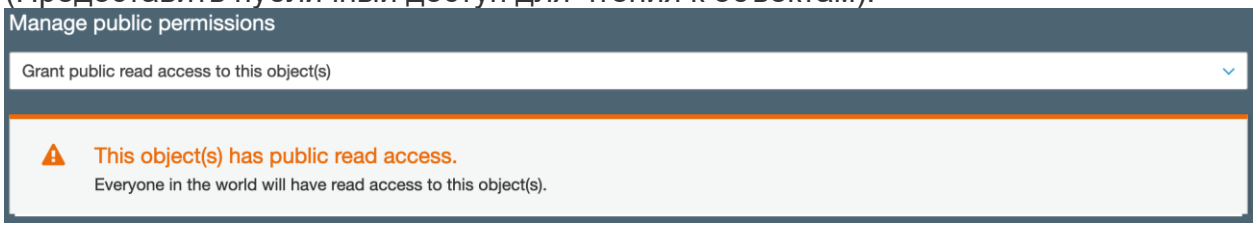
7. Нажмите синюю кнопку Upload (Загрузить).

8. Нажмите синюю кнопку Add Files (Добавить файлы).

9. Выберите обновленный файл *index.html*.

10. Нажмите синюю кнопку Next (Далее).

11. В раскрывающемся списке Manage public permissions (Управление публичными разрешениями) выберите пункт Grant public read access to this object(s) (Предоставить публичный доступ для чтения к объектам).

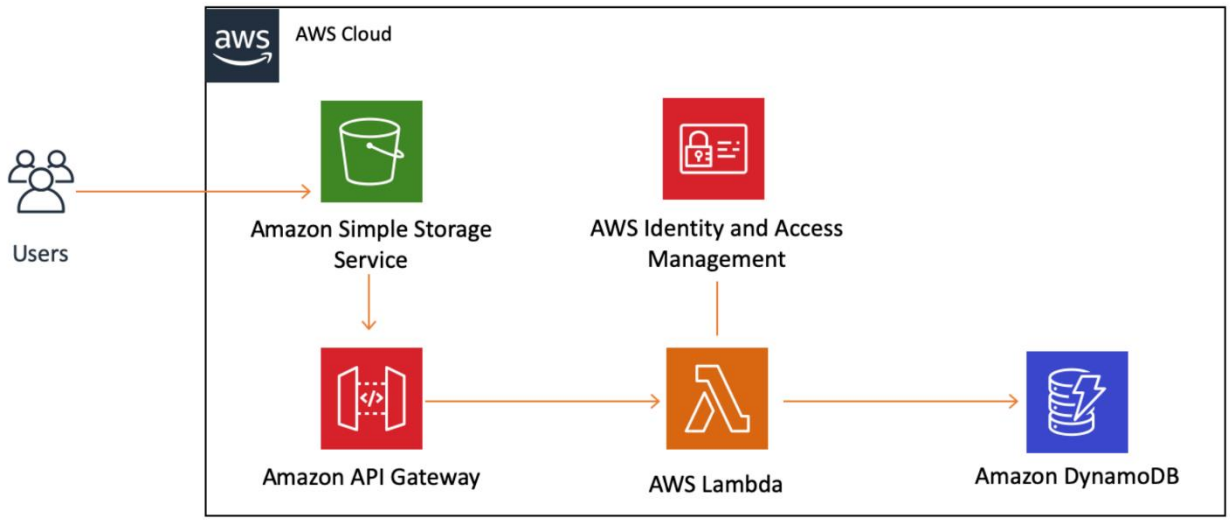


12. Нажмите синюю кнопку Upload (Загрузить) (слева).

2.5.2 Тестирование обновленного веб-клиента

1. Щелкните файл *index.html*.
2. Найдите в нижней части экрана строку Object URL и щелкните ее.
3. Ваше обновленное интернет-приложение должно загрузиться в браузере.
4. Введите свое имя (или любой другой текст) и нажмите кнопку Call API (Вызов API).
5. Должно отобразиться приветствие «Hello from Lambda» и введенный вами текст.
6. Поздравляем, теперь у вас есть работающий веб-сайт в сервисе S3, который может вызывать функцию Lambda через API Gateway!

После выполнения всех шагов выстроенная нами архитектура будет иметь следующий вид.



Все сервисы AWS, которые мы настроили, могут безопасно обмениваться данными друг с другом. Если пользователь нажимает какую-нибудь кнопку в HTML-клиенте, происходит обращение к нашему API, который вызывает функцию Lambda. Эта функция Lambda производит запись в базу данных и отправляет ответное сообщение клиенту посредством шлюза API Gateway. Всеми разрешениями управляет IAM.