**«ОБЛАЧНЫЕ ТЕХНОЛОГИИ»**

**ЛАБОРАТОРНАЯ РАБОТА № 6**

# «Программное управление облачной инфраструктурой с помощью шаблонов сервиса CloudFormation»

***Цели работы:***

1. Ознакомиться с принципами управления облачной инфраструктурой с помощью программных интерфейсов REST API AWS.
2. Познакомиться с подходом автоматического развертывания облачной инфраструктуры “Инфраструктура как код”. Изучить язык описания инфраструктуры JIML (JSON Infrastructure Marrkup Language).
3. Освоить приемы работы с интерфейсом командной строки AWS (AWS CLI).
4. Изучить приемы создания инфраструктуры проекта с помощью SDK AWS для языка высокого уровня, на примере, Node.js.
5. Изучить структуру и шаблона сервиса CloudFormation, используемого для автоматического развертывания инфраструктуры проекта.
6. Познакомиться с описанием шаблонов сервиса CloudFormation с помощью языков JSON и YAML.
7. Создать шаблон сервиса CloudFormation и развернуть инфраструктуру проекта с его помощью.

## Управление развертыванием инфраструктурой облачных вычислений в облаке AWS с помощью API AWS.

Облако AWS позволяет управлять всей инфраструктурой, создаваемой пользователем с помощью программных интерфейсов (API) ко всем основным сервисам облака AWS. Эти API реализуют отправку соответствующих вызовов (запросов) к REST API сервисов по протоколу HTTPS, как показано на рисунке 1.

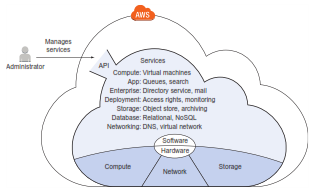


Рис. 1 Использование REST API для взаимодействия с AWS

Через API AWS доступны практически все сервисы AWS. Например, с помощью всего одного запроса (вызова) можно запустить экземпляр виртуальной машины, создать хранилище объемом 1ТВ или запустить кластер Hadoop.

Рассмотрим работу API на примере сервиса S3. Представим себе, что мы загрузим в корзину S3 несколько файлов и хотим получить список этих файлов с тем, чтобы убедиться в успешности загрузки. Получить список загруженных файлов можно с помощью простого запроса по протоколу HTTP:

GET / HTTP/1.1

Host: BucketName.s3.amazonaws.com

Authorization: [...]

HTTP ответ будет выглядеть так:

HTTP/1.1 200 OK

x-amz-id-2: [...]

x-amz-request-id: […]

Date: Mon, 09 Feb 2015 10:32:16 GMT

Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>

<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">

[...]

</ListBucketResult>

Выполнять обращение к API S3 с помощью "сырого" запроса HTTPS не очень удобно. Более простым является использование соответствующего ПО CLI AWS или SDK, которое необходимо развернуть на машине клиента AWS.

### 1.1 Понятие "Инфраструктура как код"

"Инфраструктура как код" - это понятие, которое описывает методологию управления инфраструктурой программных систем с помощью языков высокого уровня. Инфраструктура облака Amazon может включать в себя любые ресурсы AWS, включая сетевую топологию, балансировщик нагрузки, DNS записи и т.д. При разработке ПО применение таких инструментов, как автоматическое тестирование, репозитории кодов, серверы, используемые для развертывания компиляции и построения билдов (сборов) программных продуктов, позволяет значительно повысить качество проектирования. Если рассматривать инфраструктуру как код, то все эти инструменты могут быть использованы для управления инфраструктурой проектов и улучшения качества этой инфраструктуры.

### 1.2 Автоматизация развертывания ПО и движение DevOps

Движение DevOps (как новое направление в профессиональной культуре организации процесса проектирования ПО) нацелено на сближение команд администраторов инфраструктуры проекта (Operations) и разработчиков ПО (Developers) с целью организации их совместной работы на этапах тестирования и развертывания в производственной среде созданного программного продукта. Это сближение реализуется одним из двух способов:

Формированием смешанных команд, состоящих из администраторов и разработчиков. В этом случае разработчики становятся ответственными за выполнение административных задач в режиме работы по вызову. Администраторы в свою очередь вовлекаются в цикл разработки, что позволяет создавать ПО более простым для администрирования на этапах его развертывания и эксплуатации.

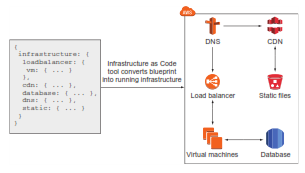
Введение новой роли, которая устраняет разрыв между разработчиками ПО и администраторами. Эта роль обеспечивает связь между администраторами и разработчиками и отвечает за выполнение работ, касающихся обеих групп.

Цель использования методов DevOps состоит в том, чтобы разрабатывать ПО и разворачивать его оперативно у пользователя без негативных последствий на качество продукта. Совместная работа администраторов и разработчиков в конечном итоге позволяет автоматизировать процесс развертывания новых выпусков в промышленном окружении с обеспечением полной изоляции различных выпусков одного и того же программного продукта друг от друга.

Появляется возможность автоматически создавать новый выпуск ПО для каждого изменения, вносимого в код, обеспечивая, таким образом, большое число развертываний в течение одного рабочего дня. Каждый выпуск при этом тестируется, независимо от других, и в случае успешного завершения тестов разворачивается в продакшен окружении. При этом обеспечивается создание своей инфраструктуры для развертывания, состоящей из виртуальных машин, баз данных, сетевых топологий и т.п., которая будет полностью изолирована от инфраструктур, в которых разворачиваются другие выпуски того же ПО.

### 1.3 Описание инфраструктуры

Рассмотрим в качестве примера языка описание инфраструктуры как кода язык JIML-JSON Infrastructure Markup Language. На рисунке ниже показана инфраструктура, которая далее будет описана с помощью этого языка.



Эта инфраструктура состоит из следующих элементов:

1. балансировщик нагрузки (LB-Load balancer);
2. виртуальные машины (VMs);
3. база данных (DB);
4. DNS записи (DNS entry);
5. сеть доставки контента (Content Delivery Network - CDN);
6. корзина (Bucket) для хранения статических файлов.

Для простоты примем, что JIML имеет синтаксис JSON. В листинге 1 приведен код JIML, описывающий инфраструктуру, приведенную на рисунке выше. Знак "$"означает ссылку на ID элемента инфраструктуры.

Листинг 1

{

"region": "us-east-1",

"resources": [{

"type": "loadbalancer",

"id": "LB",

"config": {

"virtualmachines": 2,

"virtualmachine": {

"cpu": 2,

"ram": 4,

"os": "ubuntu"

}

},

"waitFor": "$DB"

}, {

"type": "cdn",

"id": "CDN",

"config": {

"defaultSource": "$LB",

"sources": [{

"path": "/static/\*",

"source": "$BUCKET"

}]

}

}, {

"type": "database",

"id": "DB",

"config": {

"password": "\*\*\*",

"engine": "MySQL"

}

}, {

"type": "dns",

"config": {

"from": "www.mydomain.com",

"to": "$CDN"

}

}, {

"type": "bucket",

"id": "BUCKET"

}]

}

Рассмотрим, каким образом этот файл может быть использован для выполнения запросов API AWS. Последовательность обработки следующая:

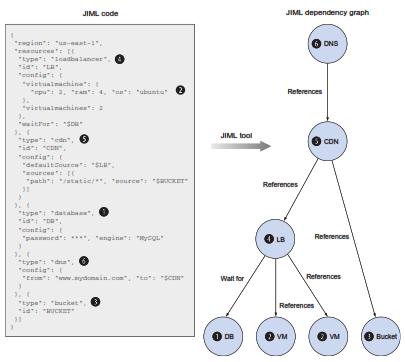
1. Файл JIML поступает на вход ПО AWS API и выполняется его синтаксический разбор (парсинг).

2. Инструмент обработки JIML создает граф зависимостей, связывая ресурсы с этими зависимостями.

3. Граф зависимостей обрабатывается от листьев к корню, последовательно обходя ветви одного уровня, в результате чего строится линейная последовательность команд на псевдоязыке.

4. Команды псевдоязыка транслируются средой исполнения JIML в вызовы API.

Вызовы API AWS должны создаваться для управления ресурсами, определенных в шаблоне JIML. В частности, вызовы должны посылаться в правильной последовательности. Граф зависимостей показан на рисунке ниже.



Последовательность команд на псевдоязыке, построенная в результате анализа графа зависимостей, приведена в листинге 2.

Листинг 2

$DB = database create {"password": "\*\*\*", "engine": "MySQL"}

$VM1 = virtualmachine create {"cpu": 2, "ram": 4, "os": "ubuntu"}

$VM2 = virtualmachine create {"cpu": 2, "ram": 4, "os": "ubuntu"}

$BUCKET = bucket create {}

await [$DB, $VM1, $VM2]

$LB = loadbalancer create {"virtualmachines": [$VM1, $VM2]}

await [$LB, $BUCKET]

$CDN = cdn create {...}

await $CDN

$DNS = dns create {...}

await $DNS

## Краткие сведения о сервисах AWS используемых в этой лабораторной работе

AWS CloudFormation – это один из наиболее мощных инструментов управления развертыванием (созданием) облачной инфраструктуры AWS. AWS CloudFormation основывается на использовании шаблонов, которые в терминологии CloudFormation называются **blueprints** (светокопия, синька, прототип, модель), в то время как сервис управления конфигурацией AWS использует термин **template** (шаблон образец, трафарет).

Шаблон является описанием создаваемой инфраструктуры.

## Порядок выполнения работы

## Задание 1. Подготовка к выполнению лабораторной работы

а. Изучить сведения, изложенные в разделах 1 и 2

б. В лабораторной работе используется предварительно созданные шаблоны и программы. Перед выполнением заданий эти шаблоны и программы необходимо скачать с GiHub по ссылке: <https://github.com/AWSinAction/code2>

Загрузите архив по ссылке: [https://](https://github.com/AWSinAction/code2/archive/master.zip) [github.com/AWSinAction/code2/archive/master.zip](https://github.com/AWSinAction/code2/archive/master.zip)

в. Сохраните файл архива приложения на Вашей локальной машине. В дальнейшем используйте файлы из этого архива для выполнения лабораторной работы, следуя соответствующим указаниям.

## Задание 2. Подготовка интерфейса командной строки AWS для создания облачной инфраструктуры

Порядок выполнения этого задания зависит от того, какая ОС установлена на машине пользователя и будет использоваться для работы с CLI AWS.

Для ОС Linux и Mac OS X:

### Шаг 1. Подготовка к установке CLI AWS для Linux/Mac OS X

Для развертывания ПО CLI AWS под этими ОС необходимо, чтобы на машинах были установлены пакеты:

- Python 2.6.5+ или Python 3.3+;

- pip (Python installing packet) – пакет, рекомендуемый для установки пакетов на базе Python.

а. Проверить наличие в системе установленного пакета Python и его версию можно с помощью команды, выполненной в окне консоли (терминала):

$ python --version

если пакет не установлен, или его версия не соответствует требуемой, то требуется установить или обновить его из репозитория.

Замечание. Найти нужные пакеты можно перейдя, по ссылке на страницу <http://aws.amazon.com/cli/>, на которой имеются соответствующие ссылки.

б. Проверить наличие в системе pip Можно с помощью команды:

$ pip --version

в. Если в результате будет выведен номер версии pip, то значит программа установлена, если номер версии не бет выведен, то программу pip необходимо установить выполнив следующие команды:

$ curl "https://bootstrap.pypa.io/get-pip.py" -o "get-pip.py"

$ sudo python get-pip.py

г. Вновь проверим установку pip с помощью команды:

$ pip --version

Если номер версии pip будет выведен, то pip был установлен успешно, иначе повторить установку pip.

### Шаг 2. Установка ПО CLI AWS для Linux/Mac OS X

а. Для установки ПО CLI необходимо выполнить команду:

$ sudo pip install awscli

б. Проверить результат установки можно с помощью команды:

$ aws --version

Если будет выведен номер версии 1.11.36 или выше, то установка прошла успешно, в противном случае установку ПО CLI AWS надо повторить.

Для ОС Windows:

### Шаг 1. Подготовка к установке CLI AWS для Windows

а. Со страницы <http://aws.amazon.com/cli/> загрузите установщик ПО AWS CLI (32-bit или 64-bit).

### Шаг 2. Установка ПО CLI AWS для Windows

а. Запустить на выполнение установщик и выполнить установку ПО CLI следуя, указаниям мастера.

б. Запустить консоль PowerShell от имени Администратора, для чего в меню Start выберите Все программы > Админисирирование > Windows PowerShell Modules и из контекстного меню выбрать пункт "Запуск от имени Администратора".

в. Перейти в окно консоли PowerShell и выполнить команду:

> Set-ExecutionPolicy Unrestricted

которая, установит политику разрешающую исполнение неподписанных скриптов.

г. Закрыть консоль PowerShell, и в дальнейшем при выполнении данной лабораторной работы больше никогда не работать с правами Администратора

д. Вновь открыть консоль PowerShell c помощью меню Start. Проверить версию установленного ПО AWS CLI с помощью команды:

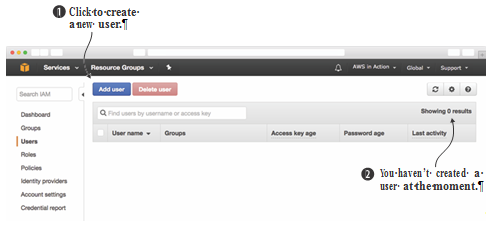
> aws --version

Если будет выведен номер версии 1.11.36 или выше, то установка прошла успешно, в противном случае установку ПО CLI AWS надо повторить.

### Шаг 3. Создание нового пользователя CLI AWS

Прежде чем, можно будет начать работу с помощью CLI AWS, требуется создать новый аккаунт пользователя с помощью сервиса IAM (AWS Identity and Access Management).

а. Для создание нового пользователя IAM необходимо перейти на страницу <https://console.aws.amazon.com> и кликнуть ссылку Services в области навигации, затем выбрать сервис IAM. Откроется страница:



в которой, необходимо:

1. Выбрать ссылку Users в области навигации (левая область страницы);

2. Кликнуть кнопку Add user, будет выполнен переход на начальную страницу мастера создания учетной записи пользователя.

На этой странице (приведена ниже) необходимо выполнить:

1. Ввести имя пользователя в виде:

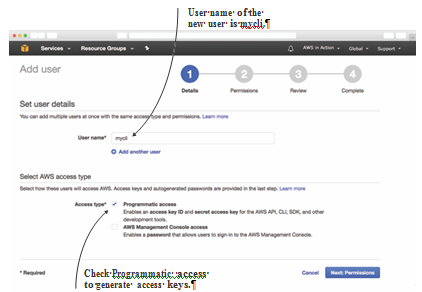
xx-yy-sssss-cli

где: xx – номер группы;

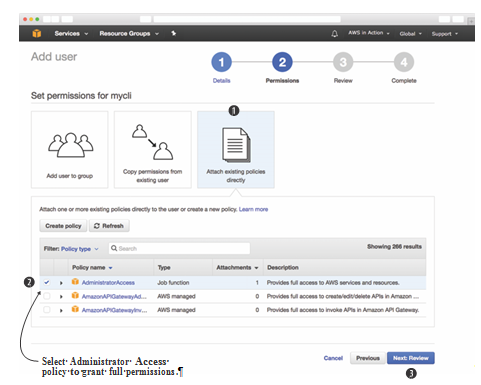
yy – номер студента по журналу.

sssss – специальность (isit, poit, dewi и т.п.)

2. В поле Acess Type выбрать Programmatic Access.



3. Кликнуть кнопку Next: Permissions. Будет выполнен переход на следующую страницу мастера (Permissions).



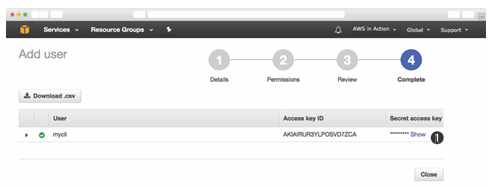
б. На этой странице надо:

1. Кликнуть картинку "Attach Existing Policies Directly".

2. Установить отметку в боксе строки "AdministratorAccess", что соответствует выбору политики, дающей все полномочия доступа.

3. Кликнуть Next:Review, для перехода на страницу проверки и подтверждения выбранных настоек для учетной записи пользователя.

в. На странице Review (она не показана) выводятся выбранные настройки аккаунта создаваемого пользователя. Убедитесь в их правильности и нажмите Create User. Откроется страница завершения создания нового пользователя (страница "Complete").



### Шаг 4. Конфигурирование параметров безопасности CLI AWS

а. На странице "Complete" мастера создания нового пользователя необходимо кликнуть ссылку Show, для того чтобы отобразить значения сгенерированных параметров безопасности (credentials) пользователя, которые будут использованы при работе с CLI AWS.

б. Эти параметры безопасности необходимо скопировать в конфигурацию CLI AWS. Для этого надо выполнить:

1. Открыть окно консоли (терминала) локальной машины. Для Linux/Mac OS X это будет консоль bash, а для Windows консоль PowerShell.

В окне локальной консоли (не AWS Management Console) выполнить команду:

aws configure

2. Будет запрошено 4 блока информации:

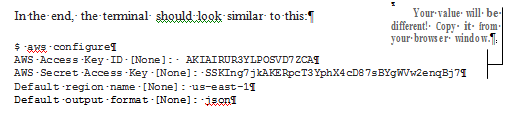
а) AWS access key ID - необходимо скопировать соответствующий блок из колонки Access key ID страницы "Complete", открытой в браузере и вставить его в соответствующую строку локальной консоли.

б) AWS secret access key скопировать соответствующий блок из колонки Secret access key ID страницы "Complete", открытой в браузере и вставить его в соответствующую строку локальной консоли.

в) Default region name — Ввести наименование региона en-central-1.

г) Default output format – Ввести json/

В результате окно локальной консоли должно выглядеть так:



г. Кликнуть кнопку Close на странице "Complete" мастера создания пользователя.

### Шаг 5. Проверка правильности развертывания ПО CLI AWS

На предыдущем шаге CLI AWS был сконфигурирован для работы пользователя xx-yy-sssss-cli.

Сейчас необходимо проверить проверить работоспособность развернутого и настроенного CLI AWS. Для этого:

а. Перейти в окно локальной консоли и выполнить в ней команду:

$ aws ec2 describe-regions

{

"Regions": [

{

"Endpoint": "ec2.eu-central-1.amazonaws.com",

"RegionName": "eu-central-1"

},

{

"Endpoint": "ec2.sa-east-1.amazonaws.com", "RegionName": "sa-east-1"

}, [...]

{

"Endpoint": "ec2.ap-southeast-2.amazonaws.com",

"RegionName": "ap-southeast-2"

},

{

"Endpoint": "ec2.ap-southeast-1.amazonaws.com", "RegionName": "ap-southeast-1"

}

]

}

Если в ответ будет выдан список регионов, то это значит, что AWS CLI работает !

## Задание 2. Использование командной строки AWS для создания облачной инфраструктуры

### Шаг 1. Знакомство с некоторыми командами CLI AWS

Общий формат команды CLI AWC имеет вид:

$ aws <service> <action> [--key value ...]

где: <service> наименование сервиса AWS

<action> выполняемое действие

--key value значение ключей (параметров) команды

а. Выполните следующие команды:

1. Вывести список экземпляров виртуальных машин типа t2.micro запущенных пользователем.

$ aws ec2 describe-instances --filters "Name=instance-type,Values=t2.micro"

Результат:

{

"Reservations": []

}

Список пустой, т.к. пользователем не было запущено ни одного инстанса ВМ.

2. Получение справки по языку CLI

aws help – Вывести список всех доступных сервисов.

aws <service> help - Вывести список всех действий поддерживаемых сервисом.

aws <service> <action> help - Вывести список всех опций доступных при обращении к определенному действию выполняемому сервисом.

### Шаг 2. Знакомство со скриптами CLI AWS

а. В качестве примера рассмотрим скрипт virtualmachine.sh (Листинг 1) для ОС Linux. Он создает виртуальную машину, сеть, группу безопасности, запускает ВМ (строка 8), подключается к ней по SSH (строка 13), выполняет на ВМ некоторый тест работоспособности созданного экземпляра (например, команду ожидания ввода информации) (строка 13-14), а затем завершает работу ВМ и прекращает ее использование как ресурса AWS (строка 15-18).

Листинг 1. virtualmachine.sh

1. #!/bin/bash -e
2. # You need to install the AWS Command Line Interface from http://aws.amazon.com/cli/
3. AMIID="$(aws ec2 describe-images --filters "Name=name,Values=amzn-ami-hvm-2017.09.1.\*-x86\_64-gp2" --query "Images[0].ImageId" --output text)"
4. VPCID="$(aws ec2 describe-vpcs --filter "Name=isDefault, Values=true" --query "Vpcs[0].VpcId" --output text)"
5. SUBNETID="$(aws ec2 describe-subnets --filters "Name=vpc-id, Values=$VPCID" --query "Subnets[0].SubnetId" --output text)"
6. SGID="$(aws ec2 create-security-group --group-name mysecuritygroup --description "My security group" --vpc-id "$VPCID" --output text)"
7. aws ec2 authorize-security-group-ingress --group-id "$SGID" --protocol tcp --port 22 --cidr 0.0.0.0/0
8. INSTANCEID="$(aws ec2 run-instances --image-id "$AMIID" --key-name mykey --instance-type t2.micro --security-group-ids "$SGID" --subnet-id "$SUBNETID" --query "Instances[0].InstanceId" --output text)"
9. echo "waiting for $INSTANCEID ..."
10. aws ec2 wait instance-running --instance-ids "$INSTANCEID"
11. PUBLICNAME="$(aws ec2 describe-instances --instance-ids "$INSTANCEID" --query "Reservations[0].Instances[0].PublicDnsName" --output text)"
12. echo "$INSTANCEID is accepting SSH connections under $PUBLICNAME"
13. echo "ssh -i mykey.pem ec2-user@$PUBLICNAME"
14. read -r -p "Press [Enter] key to terminate $INSTANCEID ..."
15. aws ec2 terminate-instances --instance-ids "$INSTANCEID"
16. echo "terminating $INSTANCEID ..."
17. aws ec2 wait instance-terminated --instance-ids "$INSTANCEID"
18. aws ec2 delete-security-group --group-id "$SGID"
19. echo "done."

Это скрипт запускается командой:

$ ./virtualmashine.sh

При выполнении данный скрипт порождает следующий вывод

…

waiting for i-c033f117 ...

accepting SSH connections under ec2-54-164-72-62.compute-1.amazonaws.com

ssh –i mykey.pem ec2-54-164-72-62.compute-1.amazonaws.com

Press [Enter] key to terminate i-c033f117 ...

[...]

terminating I –c033f117

done.

Виртуальная машина, создаваемая этим скриптом, будет работать до нажатия клавиши Enter. После нажатия этой клавиши машина будет уничтожена (действие aws ec2 terminate-instances – строка 15 листинга 1)

Использование таких скриптов имеет ряд ограничений:

- в каждуй момент времени CLI может кправлять одной ВМ;

- скрипты для различных ОС пишутся на различных языках, например для Linux это bash, а для Windows PowerShell.

- это консольное приложение, оно не поддерживает GUI.

Однако, использование CLI оправдано в следующих случаях:

- для быстрого автоматического создания виртуальных машин;

- для получения публичного имени виртуальной машины, для последующего подключения к ней с помощью SSH;

- для автоматического удаления ВМ если они становятся не нужными.

Скрипт? приведенный в листинге 1, показывает, как с помощью CLI AWS можно извлекать значения идентификаторов назначаемых автоматически AWS, например образов инстансов (ImageId). Для этих целей предусмотрена опция –-query, которая использует специальный язык JMESPath (JSON Matching Expression paths), предназначенный для извлечения данных из файлов в формате JSON.

Предположим, что JSON файл содержит:

$ aws ec2 describe-images

{

"Images": [

{

"ImageId": "ami-146e2a7c", "State": "available"

}, [...]

{

"ImageId": "ami-b66ed3de", "State": "available"

}

]

}

Для того, чтобы извлечь из него значения идентификатора для первого образа ВМ, надо выполнить команду

$ aws ec2 describe-images --query "Images[0].ImageId"

Вывод будет иметь вид

"ami-146e2a7c"

Чтобы извлечь из него значения идентификатора для второго образа ВМ, надо выполнить команду

$ aws ec2 describe-images --query "Images[1].ImageId"

Вывод будет иметь вид

"ami-b66ed3de"

Для извлечения значения состояния обеих ВМ:

$ aws ec2 describe-images --query "Images[\*].State"

Результат:

["available", "available"]

б. Выполнить скрипт virtualmachine

Для ОС Linux/MacOS X

Текст скрипта предназначенного для исполнения под ОС Linux находиться в файле

/chapter04/virtualmachine.sh

архива скриптов и ПО, скаченного с GitHub и сохраненного на локальной машине перед выполнением задания 1.

Скопировать скрипт virtualmachine.sh в папку пользователя, установить права на исполнение это файла и запустить на выполнение с помощью команды:

$ chmod +x virtualmachine.sh && ./virtualma-chine.sh

Результат выполнения должен иметь вид подобный приведенному в п.а.

Для ОС Windows

Текст скрипта предназначенного для исполнения под ОС Linux находиться в файле

/chapter04/virtualmachine.ps1

архива скриптов и ПО, скаченного с GitHub и сохраненного на локальной машине перед выполнением задания 1.

Откройте каталог в котором располагается файл virtualmachine.ps1, выберите его правой кнопкой и в контекстном меню выберите пункт Run with PowerShell, для его запуска на выполнение.

### Шаг 3. Очистка ресурсов AWS

Внимание! После выполнения скрипта virtualmachine независимо от используемой ОС, необходимо убедиться, что виртуальная машина, созданная скриптом действительно была удалена. Это можно сделать либо с помощью CLI AWS, либо с помощью консоли сервиса EC2.

а. Выполните проверку с помощью CLI AWS используя знания полученные при выполнении шага 1, данного задания.

б. Выполните проверку с помощью GUI консоли сервиса EC2, используя знания полученные при выполнении ранее выполненных лабораторных работ.

## Задание 3. Создание (программирование) облачной инфраструктуры с помощью Node.js SDK AWS

SDK AWS поддерживает целый ряд языков включая такие: Java, JavaScript, .NET, PHP, Node.js, Python, Ruby C++, Go, а также Android и iOS.

SDK AWS предоставляет удобные способы выполнять обращения к AWS API из Вашего любимого языка. SDK обеспечивает выполнение аутентификации, получение сообщений об ошибках, коммуникации по HTTPS, а также сериализацию и де сериализацию данных в форматах XML и JSON.

В этом задании будут использоваться примеры на языке JavaScript, исполняемые в окружении времени исполнения Node.js.

### Шаг 1. Установка Node.js

а. Загрузите дистрибутив Node.js для Вашей локальной ОС с сайта <https://nodejs.org>

Все примеры были проверены для Node.js 8.

б. Разверните Node.js в своей ОС исполняемой на локальной машине.

в. Проверьте работоспособность платформы Node.js, выполнив в консоли Вашей ОС команду:

node --version

Ответ должен быть:

v8.\*

Убедитесь в том, что инструмент пакетного менеджера npm также был установлен:

npm –version

В ответ на эту команду не должно быть сообщений об ошибках.

### Шаг 2. Установка программы nodecc

Для демонстрации возможностей по управлению облачной инфраструктурой AWS с помощью SDK AWS, будет использоваться программа nodecc (Node Control Center for AWS). Для установки этой программы на локальную машину необходимо:

а. Перейти в директорию в которой был развернут архив ПО и скриптов скаченный с GitHub перед началом выполнения задания 1, и далее перейти в каталог /chapter04/nodecc/.

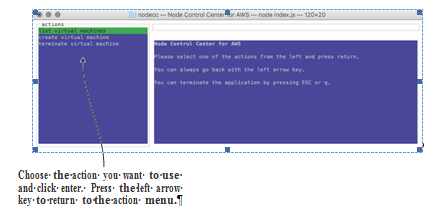
б. Выполнить в консоли ОС команду:

npm install

в. Для запуска nodecc на выполнение исполнить команду:

node index.js

Откроется окно nodecc. В левой части окна nodecc содержится перечень действий которые можно выполнить, а в правой содержиться список образов виртуальных машин с которыми можно работать. Этот список загружается из файла lib/listAMIs.js



В левую часть окна всегда можно перейти с помощью клавиши "Стрелка влево".

Nodecc использует все настройки выполненные при установке AWS CLI для пользователя xx-yy-sssss-cli (см. задание 1).

Приложение nodecc можно закрыть с помощью клавиши ESC, или путем ввода символа "q".

### Шаг 3. Анализ примера кода node.js для работы с SDK AWS

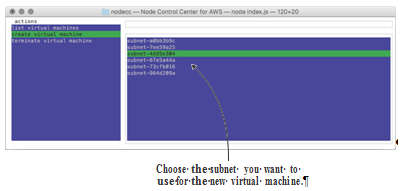
а. Ознакомиться с кодом модуля lib/listAMIs.js.

Откройте код модуля в любом текстовом редакторе и проанализируйте его код, отметьте операторы обращения к SDK и укажите функции, выполняемые этими операторами. Результат анализа приведите в отчете.

### Шаг 4. Создание виртуальной машины с помощью nodecc

а. Для запуска ВМ выберите какой-либо образ ВМ в правой половине окно nodecc, а затем выберите действие "create virtual machine" в левой половине.

б. В правой части окна nodecc будет отображен список подсетей, в этом списке надо выбрать какую-нибудь сеть, после этого будет запущен экземпляр ВМ.



Запущенные экземпляры ВМ отображаются в правой части окно nodecc в виде списка идентификаторов инстансов.

### Шаг 5. Просмотр параметров (настроек) виртуальной машины с помощью nodecc

Одной из наиболее полезных функций, поддерживаемых nodecc является возможность получения параметров необходимых для подключения к запущенному, в облаке инстансу по протоколу SSH. Для этого надо:

а. Выбрать инстанс для подключения по SSH в левой части окна nodecc. В результате в этой же части окна появятся настройки ВМ. Для подключения к ВМ по SSH нам необходимо знать ее IP адрес или DNS имя.

б В правой части окна найти параметр PublicDnsName. Теперь можно подключиться к инстансу по SSH.

в. Используя клиент SSH, имеющийся в Вашей ОС попытайтесь подключиться к запущенному экземпляру ВМ.

### Шаг 6. Удаление виртуальной машины с помощью nodecc

а. В правой части окна nodecc выберите экземпляр ВМ, который Вы хотите удалить.

б. В левой части выберите действие "terminate virtual machine ". Машина будет удалена.

в. Закройте программу nodecc.

г. Выполните проверку отсутствия запущенных Вами машин в облаке AWS с помощью CLI AWS, используя знания полученные при выполнении шага 1, предыдущего задания.

д. Выполните проверку отсутствия запущенных Вами машин в облаке AWS с помощью GUI консоли сервиса EC2, используя знания, полученные при выполнении ранее выполненных лабораторных работ.

## Задание 4. Использование шаблонов CloudFormation для создания облачной инфраструктуры

### Шаг 1. Знакомство с шаблонами CloudFormation

AWS предлагает сервис AWS Cloud, как инструмент Formation создания шаблонов развертывания облачной инфраструктуры на базе шаблонов подготовленных на YAML и JSON. Описание инфраструктуры на этих языках представляет собой декларативное определение того, как должна выглядеть создаваемая облачная инфраструктура, но не описание последовательности действий, которые необходимо предпринять, чтобы создать эту инфраструктуру.

Ниже предлагается изучить шаблон развертывания облачной инфраструктуры CloudFormation.

**а. Структура шаблона CloudFormation**

Базовый шаблон CloudFormation содержит 5 частей:

1) Версия (**Format version**). Имеет формат YYYY-MM-DD. При наличии нескольких шаблонов используется тот, который имеет более позднюю версию.

2) Описание (**Description**) - описание шаблона.

3) Параметры (**Parameters**) - параметры, используемые для настройки разворачиваемой инфраструктуры такие как: - доменное имя: ID пользователя: пароль доступа к базе данных и т.п.

4) Ресурсы (**Resources**) - ресурс - это наименьший блок, который можно описать в шаблоне. Ресурсом может быть: - виртуальная машина, балансировщик нагрузки или эластичный IP адрес.

5) Выдачи (**Ounputs**). Выдачи можно сравнить с параметрами, но их назначение иное. Выдача возвращает нечто, указанное в шаблоне, например, публичное имя экземпляра Ec2.

Базовый шаблон имеет вид, приведенный в листинге 2.

Листинг 2

--AWSTemplateFormatVersion: '2010-09-09'

Description: 'CloudFormation template structure'

Parameters:

# [...]

Resources

# [...]

Outputs:

# [...]

Формат Версии (**Format Version**). В примере, который рассматривается в этой лабораторной работе, переменная AWSTemplateFormatVersion может принимать только значение 2010-09-09. Если формат версии будет иным, то это может вызвать серьезные проблемы.

Описание (**Description**) - этот раздел не является обязательным. Однако не следует пренебрегать им, так как хорошее описание поможет в будущем вспомнить все необходимые детали, связанные с создаваемым шаблоном.

Параметры (**Parameters**). Каждый параметр имеет имя и тип. Также рекомендуется добавлять к ним комментарии (описание). Структура параметра CloudFormation приведена ниже:

Parameters:

Demo:

Type: Number

Description: 'This parameter is for demonstration'

Допустимые типы параметров приведены в таблице 3.

Таблица 3. Допустимые типы праметров.

|  |  |
| --- | --- |
| Тип | Описание |
| String  CommaDelimitedList | Строка или список строк через запятую |
| Number  List<Number> | Целое или с пл. запятой или список чисел |
| AWS::EC2::AvailabilityZone::Name  List<AWS::EC2::AvailabilityZone::Name> | Зона доступности, например us-west-2a, или список зон доступности |
| AWS::EC2::Image::Id  List<AWS::EC2::Image::Id> | Идентификатор образа виртуальной машины (AMI – Amazon Machine Image) |
| AWS::EC2::Instance::Id  List<AWS::EC2::Instance::Id> | Идентификатор экземпляра виртуальной машины EC2 |
| AWS::EC2::KeyPair::KeyName | Имя пары ключей доступа к виhтуаотной машине EC2 |
| AWS::EC2::SecurityGroup::Id  List<AWS::EC2::SecurityGroup::Id> | Идентификатор группы безопасности или список идентификаторов групп безопасности |
| AWS::EC2::Subnet::Id  List<AWS::EC2::Subnet::Id> | Идентификатор подсети или список идентификаторов подсетей |
| AWS::EC2::Volume::Id  List<AWS::EC2::Volume::Id> | Идентификатор тома EBS (Elasuic Block Storage) (подключенного сетевого хранилища) или список таких идентификаторов |
| AWS::EC2::VPC::Id  List<AWS::EC2::VPC::Id> | Идентификатор виртуального частного облака (VPC – virtual private cloud) или список таких идентификаторов. |
| AWS::Route53::HostedZone::Id  List<AWS::Route53::HostedZone::Id> | Идентификатор зоны DNS или список идентификаторов зон DNS |

Параметры могут быть расширены за счет использования свойств. Описание некоторых основных свойств параметров, приведено в таблице 4.

Таблица 4. Свойства параметров.

|  |  |  |
| --- | --- | --- |
| Свойство | Описание | Пример |
| Default | Значение параметра по-умолчанию | Default: 'm5.large' | |
| NoEcho | Скрывает значение параметра во всех графических инструментах (например, в поле Password) | NoEcho: true | |
| AllowedValues | Определяет возможные значения для параметра | AllowedValues: [1, 2, 3] | |
| AllowedPattern | регулярное выражение для определения шаблона для ввода параметра | AllowedPattern: '[a-zA-Z0-9]\*'allows only a–z, A–Z, and 0–9 with any  length | |
| MinLength, MaxLength | Определяет длину поля ввода параметра | MinLength: 12 | |
| MinValue, Max Value | Совместно с типом Number определяет нижнею и верхнею границы значения параметра | MaxValue: 10 | |
| ConstrainDescription | Строка которая определяет ограничения, если ограничения разрешены. | ConstraintDescription:  'Maximum value is 10.' | |

Секция параметров шаблона CloudFormation может выглядеть так:

Parameters:

KeyName:

Description: 'Key Pair name'

Type: 'AWS::EC2::KeyPair::KeyName'

NumberOfVirtualMachines:

Description: 'How many virtual machine do you like?'

Type: Number

Default: 1

MinValue: 1

MaxValue: 5

WordPressVersion:

Description: 'Which version of WordPress do you want?'

Type: String

AllowedValues: ['4.1.1', '4.0.1']

**Ресурсы.** Ресурс имеет имя, тип и несколько свойств. Например:

Resources:

VM:

Type: 'AWS::EC2::Instance'

Properties::

#[…]

При определении ресурсов необходимо знать их типы и свойства типов. Рассмотрим пример описания единичного инстанса EC2:

Resources:

VM:

Type: 'AWS::EC2::Instance'

Properties:

ImageId: 'ami-6057e21a'

InstanceType: 't2.micro'

KeyName: mykey

NetworkInterfaces:

- AssociatePublicIpAddress: true

DeleteOnTermination: true

DeviceIndex: 0

GroupSet:

- 'sg-123456'

SubnetId: 'subnet-123456'

Выдачи (**Outputs**). Выдачи шаблонов CloudFormation включают как минимум имя выдачи и значение. Однако рекомендуется вносить и описание, что позволит более точно описывать выдачи. Выдачи (Outputs) имеют структуру:

Outputs:

Name Outputs:

Value: '1'

Description: 'This output is always 1'

Статические выдачи не очень информативны и полезны. В большинстве случаев используются значения, которые ссылаются на имена ресурсов или атрибутов ресурсов, подробно приведены ниже:

Outputs:

ID:

Value: !Ref Server

escription: 'ID of the EC2 instance'

PublicName:

Value: !GetAtt 'Server.PublicDnsName'

Description: 'Public name of the EC2 instance

Сведения, изложенные выше, можно использовать при создании собственных шаблонов CloudFormation.

**б. Создание шаблона развертывания CloudFormation.** Шаблон развертывания CloudFormation можно создать разными способами:

1. с нуля, с помощью текстового редактора или средствами визуальной среды разработки (IDE);
2. использовать CloudFormation Designer - графический пользовательский интерфейс, предлагаемый AWS;
3. начать с шаблона из публичной библиотеки и адаптировать его к своим нуждам;
4. использовать инструмент CloudFormer, предназначенный для создания шаблонов, для существующих инфраструктур, представляемый AWS;
5. использовать шаблоны, предлагаемые другими поставщиками.

AWS и его партнеры предлагают шаблоны для CloudFormation, предназначенные для развертывания наиболее популярных решений: - AWS Quick Start, которые доступны по ссылке: https://aws.amazon.com/quickstart.

Кроме того, имеются библиотеки шаблонов с открытым кодом, используемые для повседневной работы, например, Git Hub: https://github.com/Widdx/aws-cf-templates.

Декларативный подход, реализованный CloudFormation достаточно простой. Для изменения типа экземпляра VM достаточно изменить свойство InstanceType и обновить шаблон. Значение свойства InstanceType может быть передано в шаблон через параметры. Ниже приведен листинг шаблона создания инстанса EC2.

Листинг 3.

--

AWSTemplateFormatVersion:

'2010-09-09'

Description: 'AWS in Action: chapter 4'

Parameters:

KeyName:

Description: 'Key Pair name'

Type: 'AWS::EC2::KeyPair::KeyName'

Default: mykey

VPC:

# [...]

Subnet:

# [...]

InstanceType:

Description: 'Select one of the possible instance types'

Type: String

Default: 't2.micro'

AllowedValues": ['t2.micro', 't2.small', 't2.medium']

Resources:

SecurityGroup:

Type: 'AWS::EC2::SecurityGroup'

Properties:

# [...]

VM:

Type: 'AWS::EC2::Instance'

Properties:

ImageId: 'ami-6057e21a'

InstanceType: !Ref InstanceType

KeyName: !Ref KeyName

NetworkInterfaces:

- AssociatePublicIpAddress: true

DeleteOnTermination: true

DeviceIndex: 0

GroupSet:

- !Ref SecurityGroup

SubnetId: !Ref Subnet

Outputs:

PublicName:

Value: !GetAtt 'Server.PublicDnsName'

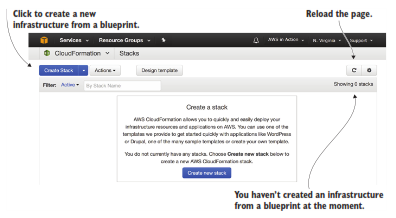
Description: 'Public name (connect via SSH as user ec2-user)'

В качестве примера шаблона далее будем использовать шаблон /chapter04/virtualmachine.yaml, который был загружен и сохранен на локальной машине перед началом выполнения заданий.

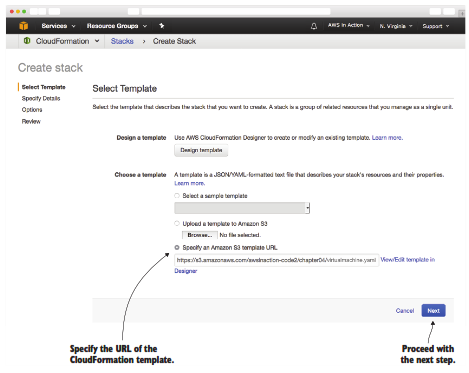
Инфраструктура, создаваемая на основе шаблонов CloudFormation в терминологии AWS, называется стек (stack). Из одного шаблона может быть создано множество стеков.

### Шаг 2. Создание первого шаблона

а). Откройте AWS Management Console (https://console.aws.amazon.com). Кликните Services в области навигации, а затем кликните CloudFormation service.



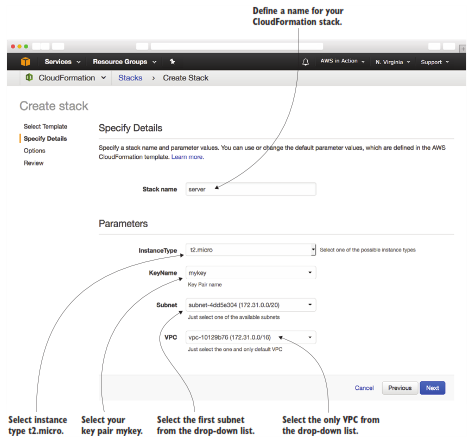
б). Кликните кнопку "Create Stack" для запуска мастера. Откроется страница четырех-шагового мастера создания стека. Откроется начальная страница сервиса CloudFormation, в которой отображается состояние всех стеков, запущенных пользователем.



На этой странице выберите селектор "Specify in Amazon S3 template URL" и в поле, расположенном ниже селектора, введите место расположения шаблона:

https://S3.amazonaws/com/awsinaction-code2/chapter04/virtualmachine.yaml.

Нажать кнопку Next. Откроется страница "Specify Details".



в). Далее необходимо определить имя стека и параметры. Рекомендуется следующий формат имени стека:

xx-yy-sssss-Server

где: xx – номер группы;

yy – номер студента по журналу.

sssss – специальность (isit, poit, dewi и т.п.)

И следующие значения параметров:

InstanceType - t2.micro

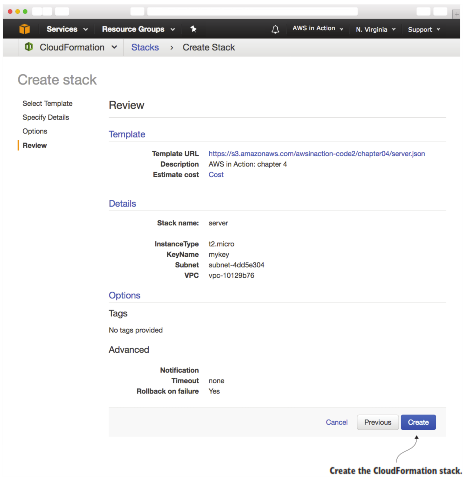
KeyName - xx-yy-mykey

Subnet - выбрать первое значение из выпадающего списка.

VPC - выбрать первое значение из выпадающего списка.

Кликните Next.

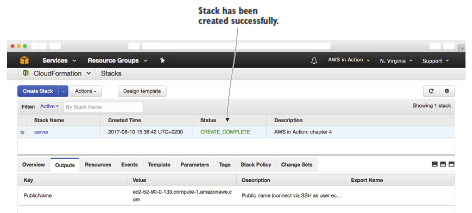
г. На следующем шаге можно определить опциональные (не обязательные) тэги стека и дополнительные настройки. Этот шаг в данной лабораторной работе пропускается. Нажмите Next в открывшемся окне опционных настроек (третье окно мастера). Настройки, для которых не были указаны значения, будут установлены по умолчанию. Откроется страница "Review".



д. Проверьте настройки создаваемого стека и кликните "Create" для его создания, иначе измените настройки, и опять "Create" стек.

Начнется процесс создания стека (развертывания инфраструктуры).

Если создание стека пройдет успешно, то откроется страница сервиса CloudFormation, в которой будет отражено текущее состояние стека.



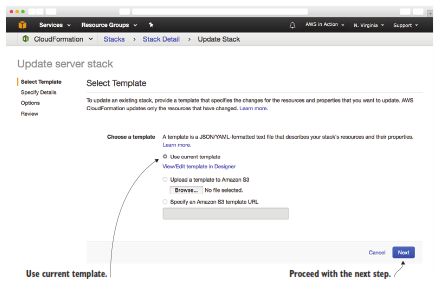
CREATE\_IN\_PROGRESS означает, что стек создается, CREATE-COMPLETE - стек успешно создан.

е. Когда сервис будет создан, выберите стек по его имени и кликните ссылку на закладку Outputs для того, чтобы увидеть публичное имя инстанса ЕС2, которое отображается колонке "Kye".

Сейчас можно протестировать возможность модификации типа инстанса в развернутом стеке.

### Шаг 3. Изменение стека

а. На странице CloudFomation Stacks выберите стек и кликните кнопку "Update Stack". Будет запущен мастер, подобный мастеру создания стека.



На первой странице мастера (Select Template) выберите селектор "Use current template" и кликните Next.

б. На втором шаге мастера можно изменить параметр InstanceType, выбрав значение t2.small для увеличения в 2 раза производительности сервера или t2.medium для четырех кратного увеличения производительности.

Замечание. Запуск инстансов t2.small или t2.medium потребует дополнительной оплаты. Верните параметру InstanceType значение t2. micro и кликните Next.

в. В третьем окне мастера модификации стека ничего изменять не надо. Просто кликните Next.

г. Четвертая страница отображает данные изменения, проверьте их и кликните Update. Дождитесь завершения перезапуска стека. И получите публичное имя стека.

### Шаг 4. Удаление стека

а. Удалите стек, выбрав его на странице CloudFormation Stack, и кликните кнопку "Delete Stack".

б. Убедитесь, что стек удалился, иначе повторите процедуру удаления.

## Задание 5. Подготовить отчет о выполнении лабораторной работы

В отчете необходимо кратко описать все выполненные действия и привести скриншоты с результатами выполнения каждого из заданий. Теоретически описать процесс привязки эластичного IP адреса к имени сайта. В отчет также должны быть включены ответы на контрольные вопросы.

## Контрольные вопросы: