OTUS

Облачные сервисы

Курс для разработчиков , которые хотят научиться развертывать инфраструктуру и проектировать архитектуру на базе облачных сервисов AWS и GCP

Длительность курса: 152 академических часа

1 Cloud computing in theory

1	**Введение в облачные сервисы**	что такое облако. Виды предоставляемых сервисов (виртуальные машины, контейнеры, сервисы). Обзор популярных облачных решений (AWS, GCP, Azure, IBM bluemix) и их возможностей.
2	**Подходы к выстраиванию инфраструктуры**	рассмотрим основные подходы к организации инфраструктуры в разрезе решаемых задач. Поговорим о жизненном цикле разработки ПО, архитектурных решениях и их влиянии на инфраструктуру. Затронем проблему “бесшовной” поставки ПО (CI / CD). Рассмотрим паттерн Infrastructure as code, и обсудим проблемы мониторинга.

3 **Docker**

кластеры и развертка приложений. Обзор docker swarm и Kubernetes.

Домашние задания

1 Docker

Цель: Разобраться с основами docker, с образа, эко системой docker в целом.

Описание ДЗ в документе

2 AWS services

1 **Введение. Основные компоненты платформы AWS**

разбор основных компонентов платформы AWS (которые будут изучены в рамках курса). Обзор системы управления аккаунтами и доступами AWS IAM.

Домашние задания

1 AWS IAM

Цель: Получить первичные навыки работы с aws web console.
Получить первичные навыки работы с aws iam.
Получить первичные навыки работы с aws ec2.
Получить первичные навыки работы с aws s3.

Зайти в aws тестовый аккаунт https://otus-test.signin.aws.amazon.com/console,
перейти в регион Stockholm (eu-north-1),
создать себе keypair,
создать ec2 инстанс t3.micro с образа ami-1dab2163 (ubuntu 18.04) с тегом student, равный вашему имени пользователя и вашей Security group (otus-$вашлогин*),
добавить к вашей security group доступ по ssh,
просмотреть файлы в s3 корзине otus-test,
убедиться, что доступа у вас нет,
создать aws iam политику с доступом к s3,
создать роль с установленным permissions boundary otus-students-permissions-boundary,
применить к полученной роли вашу политику,
применить полученную роль (а точнее instance profile с этой ролью),
просмотреть файлы в s3 корзине otus-test,
скачать файлы из корня корзины,
попытаться положить найденные файлы в корзину s3 в каталог $вашлогин,
убедиться, что доступа у вас нет,
получить для своего пользователя access key,
положить найденные файлы (или любые другие)

в корзину s3 в каталог $вашлогин,
удалите ваш ec2 инстанс.

---

2 **EC2 инстансы**

типы инстансов, их возможности и конфигурирование. Работа с инстансами (включение / выключение, подключение по ssh, работа с дисками (EBS, EFS)). Развертка простого веб-сервера.

Домашние задания

1 Создать аккаунт на AWS. Создать инстанс на EC2, и развернуть на нем простое веб приложение.

Цель: Получить готовый шаблон запуска EC2 (EC2 Launch Template) с простейшим веб сервером.

1. Войдите в тестовый аккаунт https://otus-test.signin.aws.amazon.com/ со своим логином.
2. Перейдите в AWS EC2 web console региона Stockholm.
3. Перейдите в раздел Launch Templates.
4. Нажмите Create launch template.
5. Укажите имя для шаблона, нажмите Show Tags и добавьте тег student со вашим логином (установите такой же тег и для Instance tags).
6. В качестве AMI ID укажите ami-1dab2163 (это ubuntu-bionic-18.04-amd64-server).
7. Для параметров launch template укажите те же настройки (Instance Type - t3.micro, Key Pair Name - ваш ключ, Security Groups - otus-$вашлогин-*, Instance tags) что и в занятии “AWS IAM” (другие настройки лучше не меняйте).
8. Нажмите Advanced Details, установите ваш IAM instance profile (также, из занятия “AWS IAM”) и в Userdata укажите строки из примера ниже (вы можете сделать тот сервер, который вам привычен, использовав нужные пакеты и файлы).
9. Нажмите Create launch template.

10. Вернитесь в раздел Launch Templates.
11. Выберите ваш шаблон и намите Actions > Launch instance from template.
12. Выберите Source template version и нажмите Launch instance from template (другие настройки лучше не меняйте).
13. Проверьте, правильно ли отработал ваш Userdata (например, вы можете зайти на инстанс и посмотреть файл /var/log/cloud-init-output.log или просто в браузере открыв адрес вашего инстанса по http).
14. Если всё хорошо, удалите ваш инстанс (НО НЕ ШАБЛОН).

Пример Userdata для простейшего веб сервера:

```
#!/bin/bash -xe
export DEBIAN_FRONTEND=noninteractive
apt-get update
apt-get -y upgrade
apt-get -y install apache2 php7.2 php7.2-xml unzip awscli
wget --output-document /tmp/aws.zip https://docs.aws.amazon.com/aws-sdk-php/v3/download/aws.zip
mkdir /var/www/aws-php-sdk
unzip /tmp/aws.zip -d /var/www/aws-php-sdk/
aws s3 cp s3://otus-test/s3-test.php /var/www/html/index.php
rm -rvf /var/www/html/index.html
service apache2 restart
```

Содержание файла s3://otus-test/s3-test.php из примера Userdata

```
<?php
require '../aws-php-sdk/aws-autoloader.php';
use Aws\S3\S3Client;
use Aws\S3\Exception\S3Exception;
$bucket = 'otus-test';
$keyname = 'NGC6543.jpg';
$s3 = new S3Client([
```

```
'version' => 'latest',
'region' => 'eu-north-1'
]);
$s3object = $s3->getCommand('GetObject', [
'Bucket' => $bucket,
'Key' => $keyname
]);
$s3request = $s3-
>createPresignedRequest($s3object, '+10 minutes');
$s3signedUrl = (string) $s3request->getUri();
echo "<img src='".$s3signedUrl."'>";
?>
```

3	**Работа с хранилищами S3**	принцип работы хранилища и типовые кейсы использования. Пример заливки простого файла. Интеграции S3 с EC2 инстансом на примере сайта с медиа контентом.

4	**Базы данных**	развертка базы данных на EC2 (на выбор, MongoDB / Postgres). Базовые стратегии бэкапирования и работа с дисками (EBS / EFS). Проблемы репликации данных и скалирования.

Домашние задания

1. Создать еще один инстанс на EC2 и развернуть на нем БД (любую).

   Цель: Создать еще один инстанс на EC2 и развернуть на нем БД (любую), далее связать созданное веб приложение (в рамках предыдущего урока) с данной базой.
   Кейс хранения данных - cockroach (или на ваш выбор).
   Пример: сохранять кол-во посещений за день конкретным ip адресом.

   1. Войдите в тестовый аккаунт https://otus-test.signin.aws.amazon.com/ со своим логином.
   2. Перейдите в AWS IAM web console региона

Stockholm.
3. Найдите вашу IAM роль для EC2 Instance Profile ($вашлогин*).
4. Добавьте в политику вашей IAM роли права на следующие дествия:
ec2:DescribeInstances
autoscaling:DescribeAutoScalingGroups
ec2:DescribeTags
5. Перейдите в AWS EC2 web console региона Stockholm.
6. Перейдите в раздел Launch Templates.
7. Нажмите Create launch template.
8. Укажите имя для шаблона, нажмите Show Tags и добавьте тег student со вашим логином (установите такой же тег и для Instance tags).
9. В качестве AMI ID укажите ami-1dab2163 (это ubuntu-bionic-18.04-amd64-server).
10. Для параметров launch template укажите те же настройки (Instance Type - t3.micro, Key Pair Name - ваш ключ, Security Groups - otus-$вашлогин-*, Instance tags) что и в занятии “AWS IAM” (другие настройки лучше не меняйте).
11. Важно - к Security Groups добавьте otus-ec2-db (sg-059fb38cbdb2c4598).
12. Нажмите Advanced Details, установите ваш IAM instance profile (также, из занятия “AWS IAM”) и в Userdata укажите строки из примера ниже (вы можете сделать тот сервер, который вам привычен, использовав нужные пакеты и файлы).
13. Нажмите Create launch template.
14. Вернитесь в раздел Launch Templates.
15. Выберите ваш шаблон и намите Actions > Launch instance from template.
16. Выберите Source template version и нажмите Launch instance from template (другие настройки лучше не меняйте).
17. Проверьте, правильно ли отработал ваш Userdata (например, вы можете зайти на инстанс и посмотреть файл /var/log/cloud-init-output.log , зайдя с инстанса в вашу базу данных и просто в браузере открыв адрес вашего инстанса по http).

18. Если всё хорошо, удалите ваш инстанс (НО НЕ ШАБЛОН).

Пример Userdata для простейшего веб сервера с базой данных:

```
#!/bin/bash -xe
export DEBIAN_FRONTEND=noninteractive

apt-get update
apt-get -y upgrade
apt-get -y install apache2 php php-xml php-pgsql unzip awscli chrony

echo 'server 169.254.169.123 prefer iburst minpoll 4 maxpoll 4
keyfile /etc/chrony/chrony.keys
driftfile /var/lib/chrony/chrony.drift
logdir /var/log/chrony
maxupdateskew 100.0
rtcsync
makestep 1 3' | tee /etc/chrony/chrony.conf
systemctl restart chrony

region=$(curl -s http://169.254.169.254/latest/dynamic/instance-identity/document|grep region|awk -F\" '{print $4}')
asginstanceips=()
for asginstanceid in `aws --region $region autoscaling describe-auto-scaling-groups --auto-scaling-group-name rteregulov-ec2-asg | grep -i instanceid | awk '{ print $2}' | cut -d',' -f1| sed -e 's/"//g'`; do
asginstanceips+=($(aws --region $region ec2 describe-instances --instance-ids $asginstanceid | grep -i PrivateIpAddress | awk '{ print $2 }' | head -1 | cut -d"," -f1 | sed -e 's/"//g'))
done
asginstanceips=$( IFS=$',' ; echo "${asginstanceips[*]}" )
instanceip=$(ec2metadata --local-ipv4)
```

```
wget -qO-
https://binaries.cockroachdb.com/cockroach-
v19.1.5.linux-amd64.tgz --output-document
/tmp/cockroach.tgz
tar -xzf /tmp/cockroach.tgz -C /usr/local/bin --strip 1

cockroach start --insecure --background --advertise-
addr=$instanceip --join=$asginstanceips



instanceid=$(ec2metadata --instance-id)
username=$(aws --region $region ec2 describe-tags
--filters "Name=resource-id,Values=${instanceid}"
"Name=key,Values=student" --output text | cut -f5)

echo 'CREATE USER IF NOT EXISTS '$username';
CREATE DATABASE IF NOT EXISTS
'$username'db;
GRANT ALL ON DATABASE '$username'db TO
'$username';
USE '$username'db;
CREATE TABLE IF NOT EXISTS views (
view_id SERIAL PRIMARY KEY,
client_ip INET,
instance_id STRING,
view_date TIMESTAMP
);' | tee /tmp/create.sql

cockroach sql --insecure < /tmp/create.sql

wget --output-document /tmp/aws.zip
https://docs.aws.amazon.com/aws-sdk-
php/v3/download/aws.zip
mkdir /var/www/aws-php-sdk
unzip /tmp/aws.zip -d /var/www/aws-php-sdk/
aws s3 cp s3://otus-test/db-test.php
/var/www/html/index.php
sed -i "s/student/$username/g"
/var/www/html/index.php
rm -rvf /var/www/html/index.html
service apache2 restart
```

Содержание файла s3://otus-test/db-test.php из примера Userdata

```
<?php
require '../aws-php-sdk/aws-autoloader.php';
use Aws\S3\S3Client;
use Aws\S3\Exception\S3Exception;
$bucket = 'otus-test';
$keyname = 'NGC6543.jpg';
$s3 = new S3Client([
'version' => 'latest',
'region' => 'eu-north-1'
]);
$s3object = $s3->getCommand('GetObject', [
'Bucket' => $bucket,
'Key' => $keyname
]);

$s3request = $s3-
>createPresignedRequest($s3object, '+10 minutes');

$s3signedUrl = (string) $s3request->getUri();
echo "<img src='".$s3signedUrl."'>";
$instance_id =
@file_get_contents("http://169.254.169.254/latest/me
ta-data/instance-id");
$client_ip = $_SERVER['REMOTE_ADDR'];
$timestamp = date('Y-m-d H:i:s');
try {
$dbh = new
PDO('pgsql:host=localhost;port=26257;dbname=stud
entdb;sslmode=disable',
'student', null, array(
PDO::ATTR_ERRMODE =>
PDO::ERRMODE_EXCEPTION,
PDO::ATTR_EMULATE_PREPARES => true,
PDO::ATTR_PERSISTENT => true
));

$dbh->exec(
"
INSERT INTO views (client_ip, instance_id,
```

```
view_date) VALUES ('$client_ip', '$instance_id',
'$timestamp');
"
);
} catch (Exception $e) {
print $e->getMessage() . "\r\n";
exit(1);
}

?>
```

5	**База данных как сервис**	понятие БД как сервис. Развертка БД на основе RDS / DocumentDB сервисов. Миграция данных (на примере развернутой базы на EC2 из предыдущего урока).

6	**Очереди сообщений**	обзор подходов к организации обмена сообщениями между сервисами. Обзор apache kafka, RabbitMQ, ActiveMQ. Развертка одной из них на EC2 инстансе. Домашние задания 1 Заменить БД на любую базу из предложенных AWS (RDS / DocumentDB). Цель: Заменить БД на любую базу из предложенных AWS (RDS / DocumentDB). На инстансе, где была БД, развернуть любую очередь. Далее необходимо доработать свое веб приложение: нужно добавить микро-сервис, который будет взаимодействовать с разработанным ранее веб приложением. Пример: микросервис может представлять из себя систему нотификаций, и когда с одного ip адреса будет поступать более 100 запросов в день, веб сервис кинет сообщение в очередь, которое заберет микро-сервис нотификации, и например, отправит email системному администратору.

7	**Очереди сообщений как сервис**	понятие очереди как сервиса. Обзор SQS сервиса. Замена развернутой очереди на EC2 (в рамках предыдущего урока) на SQS.
8	**Serverless applications**	обзор подхода. Создание lambda функций. Запуск функции на примере загрузки картинки в S3. Запуск функции на примере работы с очередью SQS. Домашние задания 1 Замена очереди сообщений (развернутой в рамках предыдущего ДЗ) на SQS. Цель: Замена очереди сообщений (развернутой в рамках предыдущего ДЗ) на SQS. Замена микросервиса по отправке уведомлений на lambda функцию.
9	**Балансировка нагрузки**	обзор ELB. Виды балансировщиков. Создание двух инстансов EC2 с балансировщиком. Другие примеры использования.

10	**Container registry**	идея репозитория контейнеров. Пример использования. Создание, деплой и развертка образов. Автоматизация. Домашние задания 1 Запустить еще один инстанс веб приложения. Цель: Запустить еще один инстанс веб приложения. Создать балансировщик нагрузки и закрепить его за двумя созданными инстансами, где работает веб приложение.
11	**Кластер на Kubernetes**	когда нужен свой кластер. Пример создания кластера. Развертка простого приложения. Kubernetes и container registry.
12	**Мониторинг**	зачем нужен мониторинг. Какие метрики. Обзор CloudWatch. Домашние задания 1 Переместить веб приложение в кластер на Kubernetes. Цель: Переместить веб приложение в кластер на Kubernetes. Настроить cloudwatch таким образом, чтобы можно было получать логи веб приложения.
13	**Networking. VPC**	зачем нужна приватная сеть. Организация своей приватной сети на примере взаимодействия между контейнерами EC2. Создание своего домена и привязка к ресурсу.

14 **CDN**

что из себя представляет сервис доставки контента в глобальной сети. Обзор технологии Cloudfront. Пример использования CDN для организации кеша веб-сервиса.

Домашние задания

1 Создать приватную сеть. Подключить VPN, чтобы можно было работать с ней удаленно.

  Цель: Создать приватную сеть. Подключить VPN, чтобы можно было работать с ней удаленно. Далее расширить разработанное веб приложение таким образом, чтобы там появился статический контент (картинки, js, css). Далее настроить CDN таким образом, чтобы он кешировал данный статический контент.

3 GCP

1	**Введение**	основные компоненты платформы GCE (которые будут изучены в рамках курса). Обзор системы управления аккаунтами и доступами Google IAM.
2	**Compute engine и app engine**	типы сервисов. Отличие compute от app engine. Типы инстансов, их возможности и конфигурирование. Работа с инстансами (включение / выключение, подключение по ssh). Развертка простого веб-сервера на compute engine. Развертка простого веб-сервера на app engine. Домашние задания 1 Перенести разработанный веб сервис из кластера AWS на compute engine. Цель: Перенести разработанный веб сервис из кластера AWS на compute engine. Привязку к SQS, БД и lambda функцию нужно оставить на AWS.
3	**Работа с хранилищами Cloud storage**	принцип работы хранилища. Пример заливки простого файла. Интеграции Cloud Storage с инстансом на compute engine на примере сайта с медиа контентом.

4 **Базы данных**

развертка базы данных на compute engine (на выбор MongoDb / Postgres). Бэкапы и работа с дисками. Проблемы репликации.

Домашние задания

1 Развернуть БД на compute engine.

Цель: Развернуть БД на compute engine. Перевести веб сервис на работу с новой развернутой БД.
Данные необходимо смигрировать.

---

5 **База данных как сервис**

развертка БД на основе Cloud SQL / Big Table сервисов. Миграция данных (на примере развернутой базы на compute engine из предыдущего урока).

Домашние задания

1 Развертка без сервера

Цель: Домашняя работа: будет дан код сайта, который нужно будет развернуть без сервера. Сайт будет отображать список загруженных картинок как лента инстаграма. Будет кнопка - загрузить новую картинку. После загрузки, картинку нужно будет обработать (сделать thumb) и залить в s3 и базу данных. После загрузки картинки она должна появиться на сайте. И никаких серверов.

6	**Очереди сообщений**	обзор подходов к организации обмена сообщениями между сервисами. Развертка очереди (apache kafka, RabbitMQ, ActiveMQ) на compute engine инстансе. Домашние задания 1 Поднять новый инстанс на compute engine и перенести на него lambda функцию из AWS. Цель: Поднять новый инстанс на compute engine и перенести на него lambda функцию из AWS. Далее поднять еще один инстанс и установить на него любую очередь. Связать эту очередь с веб сервисом и перенесенной lambda функцией.
7	**Очереди сообщений как сервис**	обзор сервиса PUB/SUB. Замена очереди (развернутой на compute engine в рамках предыдущего урока).
8	**Serverless applications**	обзор сервиса. Его отличия от AWS lambda. Создание функций. Запуск функции на примере загрузки картинки в cloud storage. Запуск функции на примере работы с PUB/SUB. Домашние задания 1 Замена очереди сообщений (развернутой в рамках предыдущего ДЗ) на PUB/SUB. Цель: Замена очереди сообщений (развернутой в рамках предыдущего ДЗ) на PUB/SUB. Перенос lambda функции на платформу google functions.

9	**Балансировка нагрузки**	обзор GC LB. Создание двух инстансов compute engine с балансировщиком. Другие примеры использования.
10	**Container registry**	обзор container registry сервиса. Его отличие от container registry AWS. Пример использования. Создание, деплой и развертка образов. Автоматизация. Домашние задания 1 Запустить еще один инстанс веб приложения. Цель: Запустить еще один инстанс веб приложения. Создать балансировщик нагрузки и закрепить его за двумя созданными инстансами, где работает веб приложение.
11	**Кластер на Kubernetes**	обзор реализации кластера на google cloud. Отличия от AWS Kubernetes. Пример создания кластера. Развертка простого приложения. Kubernetes и container registry.

12	**Мониторинг**	обзор stackDriver. Пример использования stackDriver на развернутом приложении в compute engine. Пример использования на развернутом приложении в Kubernetes. Домашние задания 1 Переместить веб приложение в кластер на Kubernetes. Цель: Переместить веб приложение в кластер на Kubernetes. Настроить cloudwatch таким образом, чтобы можно было получать логи веб приложения.
13	**Networking. VPC**	обзор VPC на google cloud. Организация своей приватной сети на примере взаимодействия между инстансами на compute engine. Создание своего домена и привязка к ресурсу.
14	**Cloud CDN**	обзор CDN на google cloud. Пример использования CDN для организации кеша веб-сервиса. Домашние задания 1 Создать приватную сеть. Подключить VPN, чтобы можно было работать с ней удаленно. Цель: Создать приватную сеть. Подключить VPN, чтобы можно было работать с ней удаленно. Далее настроить CDN таким образом, чтобы он кешировал статический контент разработанного веб приложения.

4 Risks and costs

1	**Риски при проектировании (технические и экономические)**	технические риски при проектировании архитектуры разрабатываемой ИС (отказоустойчивость, скалируемость, защищенность). Экономические риски при проектировании инфраструктуры в разрезе предложенной архитектуры (внеплановое увеличение стоимости, привязка к зависимым сервисам). Типы производств и их потребности (средняя компания, стартап и т.д.). Стадии развития компании и переоценка ценностей.
2	**Планирование бюджета**	оценка бюджета и стоимости архитектуры. Подходы к сокращению бюджета и последующее влияние на архитектуру системы.

5 Итоговый проект

1 **Итоговый проект**

в качестве финального проекта будет дана бизнес задача (или согласована с преподавателем). Нужно разработать план архитектуры (физическую и логическую топологию), рассчитать примерный бюджет и риски. Далее, необходимо будет реализовать согласованную архитектуру. Задача должна включать в себя все аспекты: выбор и развертка CVS, настройка CI/CD, авто развертка приложений, авто скалирование, защита информации.

Домашние задания

1 Проект