Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Пермский национальный** **исследовательский политехнический университет»**

Факультет: Прикладной математики и механики

Кафедра: Вычислительной математики и механики

Направление: 09.04.02 «Информационные системы и технологии»

Программа: «Информационные технологии и системная инженерия»

Отчёт по лабораторным работам

по дисциплине

«Программирование с использованием

облачных платформ GOOGLE

APPS/AMAZON/AZURE»

Выполнила:

студент гр. ИТСИ-17-1м

Боталова Юлия Евгеньевна

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*(подписи)*

Принял:

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(должность, ФИО руководителя)*

*\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(оценка) (подпись)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(дата)*

**Пермь 2018**

Оглавление

[Глава 1. Лабораторные работы 3](#_Toc532907103)

[Лабораторная работа №1. Docker, Docker Swarm, Docker Compose, HAproxy 3](#_Toc532907104)

[Лабораторная работа №2. Работа с протоколом S3 9](#_Toc532907105)

[Лабораторная работа №3 14](#_Toc532907106)

[Глава 2. Amazon Web Services Database 17](#_Toc532907107)

[1. Amazon Aurora 17](#_Toc532907108)

[2. Amazon RDS 18](#_Toc532907109)

[3. Amazon DynamoDB 20](#_Toc532907110)

[4. Amazon Redshift 21](#_Toc532907111)

[5. Amazon ElastiCache 23](#_Toc532907112)

[6. Amazon Neptune 24](#_Toc532907113)

[7. AWS Database Migration Service 25](#_Toc532907114)

[Использованные источники 28](#_Toc532907115)

# Глава 1. Лабораторные работы

## Лабораторная работа №1. Docker, Docker Swarm, Docker Compose, HAproxy

**Docker** – это программная платформа для быстрой сборки, отладки и развертывания приложений. Docker упаковывает ПО в стандартизованные блоки, которые называются [контейнерами](https://aws.amazon.com/ru/containers/). Каждый контейнер включает все необходимое для работы приложения: библиотеки, системные инструменты, код и среду исполнения. Благодаря Docker пользователи могут быстро развертывать и масштабировать свои приложения в любой среде и сохранять уверенность в том, что код будет работать.

Использование контейнеров – это стандартный способ упаковки кода приложения, конфигураций и зависимостей в один объект. Контейнеры работают на операционной системе, установленной на сервере, и запускаются как изолированные от ресурсов процессы. Это обеспечивает быстрое, надежное и согласованное развертывание, независимо от среды.

**Docker Swarm** – это средство для развертывания кластера хостов Docker. Вы можете использовать его для быстрого развертывания кластера хостов на вашем локальном компьютере, либо на поддерживаемых облачных платформах.

Docker Swarm служит для объединения множества Docker хостов в один виртуальный хост. Docker Swarm предоставляет REST API интерфейс, совместимый с Docker API. Таким образом, все инструменты, которые работают с API Docker'a смогут работать с Docker Swarm, не подозревая о том, что за ним стоит кластер Docker'ов, а не одна машина.

**Docker-compose** это простой инструмент, который позволяет настроить и запустить несколько контейнеров одной командой. Вся конфигурация для docker-compose описывается в файле docker-compose.yml, который обычно лежит в корне проекта.

По умолчанию, когда вы запускаете docker-compose up, docker-compose ищет конфигурационный файл docker-compose.yml в текущей директории. Их может быть несколько для разных вариантов использования. Первый вариант использования — это разбиение большого проекта на несколько более мелких. Даже если вы запускаете несколько отдельных docker-compose, сервисы все равно смогут общаться друг с другом по имени из docker-compose.

**HAProxy** — серверное программное обеспечение для обеспечения высокой доступности и балансировки нагрузки для TCP и HTTP-приложений, посредством распределения входящих запросов на несколько обслуживающих серверов.

Его основная задача - повышение производительности серверной среды путем распределения рабочей нагрузки среди нескольких серверов (web, приложения, базы данных). Им пользуются такие известные проекты как GitHub, Imgur, Instagram и Twitter.

**Алгоритмы распределения нагрузки**

Алгоритм распределения нагрузки отвечает за выбор сервера для обработки запроса пользователя. HAProxy предлагает нам несколько алгоритмов. В дополнение к алгоритму, мы можем указать вес каждого сервера, в соответствии с которым он будет выбран чаще или реже.

**Приведем основные алгоритмы:**

* **Roundrobin -** алгоритм, применяемый по умолчанию. Перебирает сервера по очереди.
* **Leastconn -** выбирает сервер с наименьшим числом активных соединений. Рекомендуется к использованию для длинных сессий. Сервера в одном бекэнде варьируются циклически.
* **Hash -** выбирает сервер исходя из хеша, построенного на основе IP пользователя. Таким образом, пользователь всегда обращается к одному и тому же серверу.

**Проверка работоспособности**

HAProxy применяет проверку работоспособности сервера, чтобы определить способен ли он обработать запрос. Так нам не приходится вручную исключать сервер из настройки бекэнда. По умолчанию HAProxy пытается установить TCP соединение с сервером и проверяет, производит ли тот обработку данных поступающих на 80 порт.

Если сервер не проходит проверку, то он автоматически исключается из бекэнда и трафик на него не перенаправляется до тех пор, пока сервер снова не станет доступен. Если же падают все сервера, то приложение становится недоступно пока не возобновит свою работу хотя бы один сервер.

**Цель работы:** получить навыки работы с Docker, Docker Swarm, Docker Compose, HAproxy.

1. Создаём новое приложение Node.js – в файле index.js прописываем следующее (рисунок 1):

* **nano index.js**

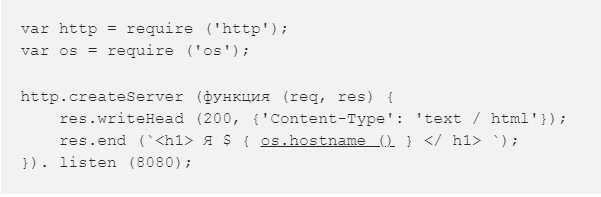


Рисунок 1. Текст index.js

1. Docker - изуем приложение, для этого создаём файл Dockerfile со следующим кодом (рисунок 2):



Рисунок 2. Текст Dockerfile

1. Создаем docker-образ нашего приложения, вводим в консоли следующую команду (рисунок 3):

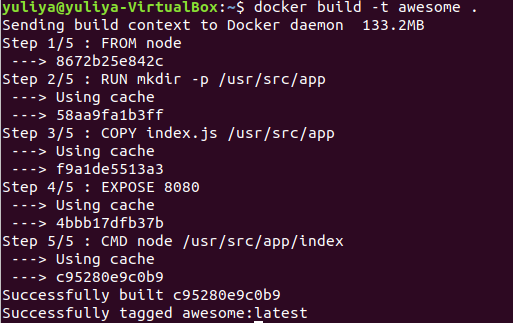
* **docker build -t awesome . **

Рисунок 3. Вызов команды docker build –t awesome .

1. Для нашего HTTP-сервера мы будем использовать HAProxy , это означает, что нам нужно создать контейнер с HAProxy, который будет прослушивать порт 80 и загружать баланс для разных контейнеров Node.js на порт 8080. Чтобы создать наши контейнеры (Node.js приложения и HAProxy), мы будем использовать Docker Compose, напишем наш docker-compose.yml файл (рисунок 4):

* **nano docker-compose.yml**

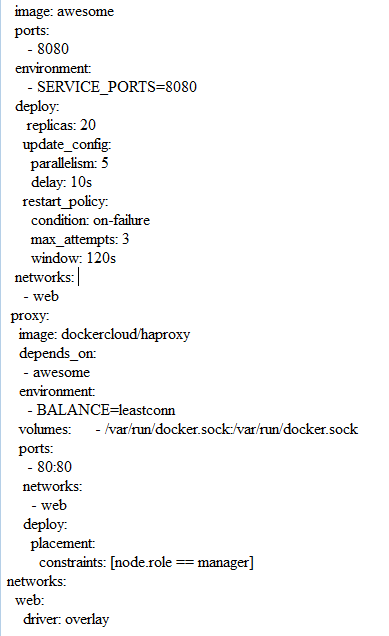


Рисунок 4. Текст docker-compose.yml

Чтоб включить docker swarm:

* **docker swarm init**

1. Cоздаём stack с названием prod из файла **docker-compose.yml:**

* **docker stack deploy --compose-file=docker-compose.yml prod**

1. Следующей командой увидим запущенные сервисы (рисунок 5)

* **docker service ls**

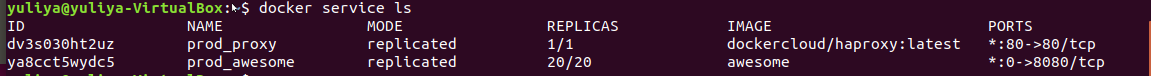


Рисунок 5. Запущенные сервисы

Когда мы откроем браузер и впишем <http://localhost> , то у нас выйдет такой результат – контейнер со своим идентификатором (рисунок 6).

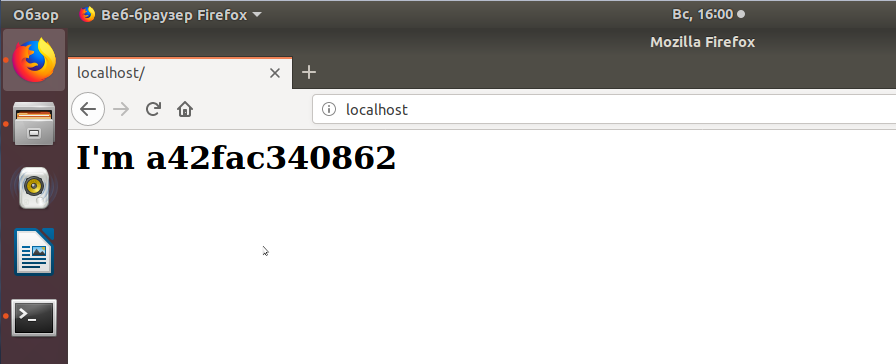


Рисунок 6. Страница в браузере localhost с идентификатором контейнера

Чтобы наверняка убедиться в существовании контейнера с идентификатором a42fac340862, в консоли сделаем проверку (рисунок 7) и, да, такой контейнер существует.

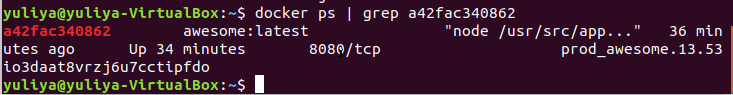


Рисунок 7. Проверка существования контейнера по идентификатору

## Лабораторная работа №2. Работа с протоколом S3

Amazon Simple Storage Service (Amazon S3) — онлайновая веб-служба, предлагаемая Amazon Web Services, предоставляющая возможность для хранения и получения любого объёма данных, в любое время из любой точки сети, так называемый файловый хостинг.

Amazon S3 – это объектное хранилище, предназначенное для хранения и извлечения любых объемов данных из любых источников: веб‑сайтов и мобильных приложений, корпоративных приложений, а также данных с датчиков или устройств IoT. S3 открывает возможности для обеспечения безопасности и соответствия согласно самым строгим нормативным требованиям. Сервис предоставляет клиентам гибкость в управлении данными, обеспечивая оптимизацию затрат, контроль доступа и соответствие требованиям. Такой веб-сервис позволяет просто создавать приложения, использующие интернет-хранилище.

Сервис Amazon S3 спроектирован для максимальной гибкой работы. Вы можете хранить данные любого типа и в любом количестве; создавать простое FTP-приложение или сложное веб-приложение.

Основной протокол доступа к объектному хранилищу – это протокол, созданный Amazon Web Services для сервиса AWS S3 (Simple Storage Service). S3 API — набор команд, которые «понимает» хранилище и выполняет в ответ некие действия. Полнота поддержки S3 API и определяет функциональность объектного хранилища.

Протокол S3 позволяет взаимодействовать с данными удалённо с помощью стандартных запросов и иметь к ним постоянный доступ.

Объектное хранилище на базе платформы Mail.Ru Cloud Solutions создано для быстрой работы с большими объемами данных и миллионами объектов. Поддерживает S3-совместимый API (REST API) и легко интегрируется в приложения, написанные на любом языке.

**Цель работы:** получить навыки работы с протоколом S3.

1. Регистрируемся в MAIL.RU CLOUD SOLUTIONS.
2. Создаём аккаунт для старта работы с Объектным хранилищем. Мой аккаунт – Jula.

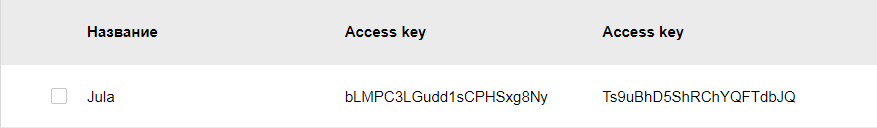


Рисунок 8. Зарегистрированный аккаунт Jula

1. Создаём бакет. В Объектном хранилище все данные хранятся в отдельных бакетах, в которых можно организовывать иерархию своих файлов. Первый созданный бакет называется firstbaket.

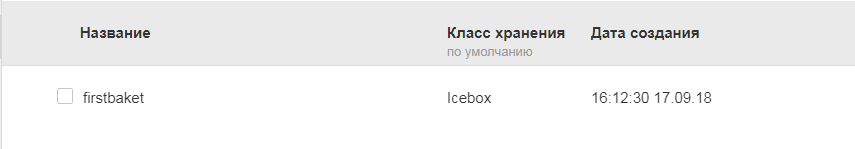


Рисунок 9. Первый созданный бакет firstbaket

1. Для прохождения аутентификации к сервисам Cloud Storage вам понадобятся ключи доступа известные как Access key ID и Secret access key – на рисунке 1 изображён только Access key ID, второй секретный ключ показывают лишь при создании нового Access key, потом он становится не доступен.

* Далее создаём файл учётных данных в ~/.aws /credentials в директории C:\Users\Boss\
* Добавляем ключи доступа в файл, выглядит файл так (рисунок 3). Для безопасности второй секретный ключ частично замазан.

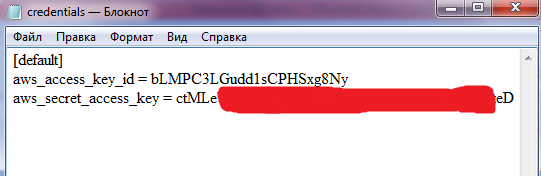


Рисунок 10. Файл credentials

1. После установки SDK загружаем пакет AWS, используя require (рисунок 4).

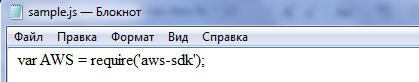


Рисунок 11. Загрузка пакета AWS

1. Загружаем учетные данные (credentials) и создаём клиента (рисунок 5).

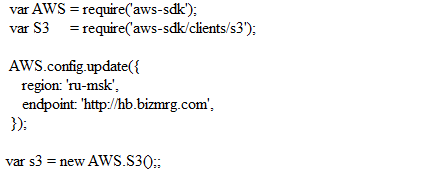


Рисунок 12. Загрузка учётных данных и создание клиента

1. Создав клиента S3, можем выполнять API-запросы:

* Получить тело объекта (рисунок 6)

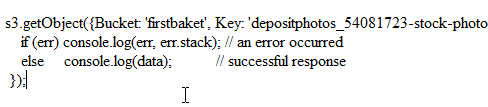


Рисунок 13. Код для получения тела объекта

* Положить объект (рисунок 7)

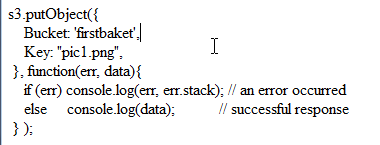


Рисунок 14. Код для загрузки объекта

На рисунке 8 видно, что без проблем загрузились объекты – pic1.png и example.js.

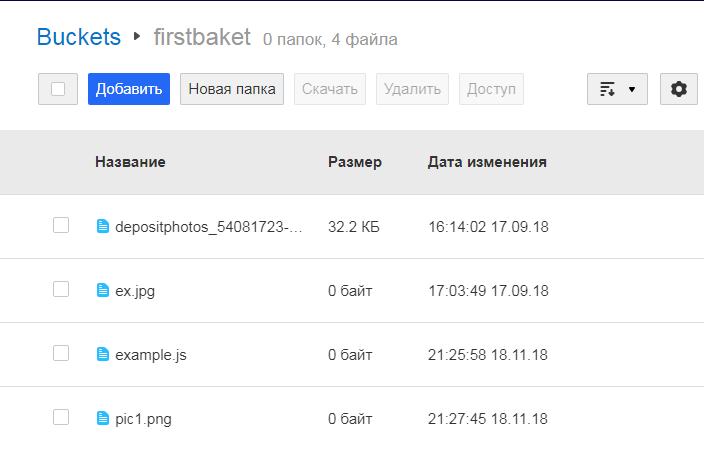


Рисунок 15. Результат работы sample.js – загрузка файла на mail.ru cloud solutions

На рисунке 9 можно увидеть результат работы sample.js. Заметим, что в sample.js в получении был указан файл «depositphotos\_540817…», на консоли видим Body этого объекта.

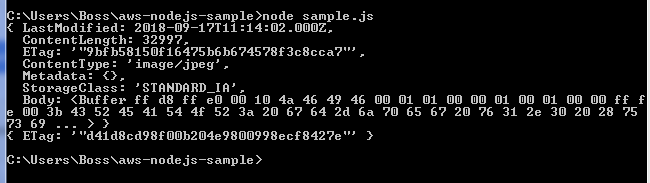


Рисунок 16. Результат работы sample.js – получение тела объекта с на mail.ru cloud solutions

## Лабораторная работа №3

**Google Cloud Platform Cloud IAM**

Google Cloud Platform (GCP) предлагает Cloud IAM, который позволяет вам управлять контролем доступа, определяя, кто (identity) имеет какой доступ (role) к какому ресурсу. Cloud IAM позволяет предоставлять детальный доступ к определенным ресурсам GCP и предотвратить нежелательный доступ к другим ресурсам. Облачный IAM позволяет применять принцип безопасности с наименьшими привилегиями, поэтому пользователь может предоставлять только необходимый доступ к своим ресурсам. Концепции, связанные с идентичностью В Cloud IAM, вы предоставляете доступ к членам (members). Участники могут быть следующих типов:

* Google аккаунт
* Сервисный аккаунт
* Группа Google
* Домен G Suite
* Облачный домен
* Google аккаунт

***Учетная запись Google*** представляет разработчика, администратора или любого другого человека, который взаимодействует с GCP. Любой адрес электронной почты, связанный с учетной записью Google, может быть личностью, включая gmail.com или другие домены. Новые пользователи могут зарегистрировать учетную запись Google, перейдя на страницу регистрации учетной записи Google.

***Сервисный аккаунт***

Учетная запись службы - это учетная запись, которая принадлежит приложению, а не отдельному конечному пользователю. Когда запускаете код, размещенный на GCP, вы указываете учетную запись, под которой должен запускаться код. Есть возможность создать столько учетных записей служб, сколько необходимо для представления различных логических компонентов для приложения.

***Группа Google***

Группа Google - это именованная коллекция учетных записей Google и учетных записей служб. Каждая группа имеет уникальный адрес электронной почты, связанный с группой. Группы Google - это удобный способ применить политику доступа к группе пользователей. Есть возможность предоставлять и изменять элементы управления доступом сразу для всей группы, а не предоставлять или изменять элементы управления доступом по одному для отдельных пользователей или учетных записей служб. Группы Google не имеют учетных данных для входа в систему, и нет возможности использовать группы Google для установления личности для отправки запроса на доступ к ресурсу.

**Концепции, связанные с управлением доступом**

Когда аутентифицированный участник пытается сделать запрос, Cloud IAM принимает решение о том, разрешено ли члену выполнять операцию с ресурсом.

***Ресурс***

Некоторые службы, такие как Cloud Pub / Sub и Compute Engine, поддерживают предоставление разрешений Cloud IAM с более высокой степенью детализации, чем на уровне проекта. В других случаях можно предоставить Cloud IAM разрешения на уровне проекта. Разрешения затем наследуются всеми ресурсами в этом проекте.

***Права доступа***

Разрешения определяют, какие операции разрешены для ресурса. В мире Cloud IAM разрешения представлены, например, в виде pubsub.subscriptions.consume. Каждая служба GCP имеет связанный набор разрешений для каждого метода REST, который она предоставляет. Вызывающая сторона этого метода нуждается в этих разрешениях для вызова этого метода. Например, вызывающему абоненту Publisher.Publish() необходимо pubsub.topics.publish разрешение. Разрешения не назначаются пользователям напрямую. Вместо этого назначается им роль, которая содержит одно или несколько разрешений.

**Роли**

Роль - это набор разрешений. Нельзя назначить разрешение пользователю напрямую; вместо этого вы даете им роль. Когда вы предоставляете роль пользователю, вы предоставляете ему все разрешения, которые содержит эта роль.

В Cloud IAM есть три вида ролей:

***Примитивные роли.*** Роли, исторически доступные в консоли Google Cloud Platform, будут продолжать работать. Это роли владельца, редактора и средства просмотра.

***Предопределенные роли***. Предопределенные роли - это роли Cloud IAM, которые обеспечивают более детальный контроль доступа, чем примитивные роли. Например, предопределенная роль Pub / Sub Publisher (role / pubsub.publisher) предоставляет доступ только к публикации сообщений в разделе Cloud Pub / Sub.

***Пользовательские роли.*** Роли, которые вы создаете, чтобы адаптировать разрешения к потребностям вашей организации, когда предопределенные роли не соответствуют вашим потребностям.

# Глава 2. Amazon Web Services Database

Amazon Web Services обеспечивает полностью управляемый сервис реляционной базы данных и базы данных NoSQL, а также полностью управляемый сервис кэширования в памяти и полностью управляемый сервис петабайтового хранилища данных.

Рассмотрим сервисы AWS Database на рисунке 17.

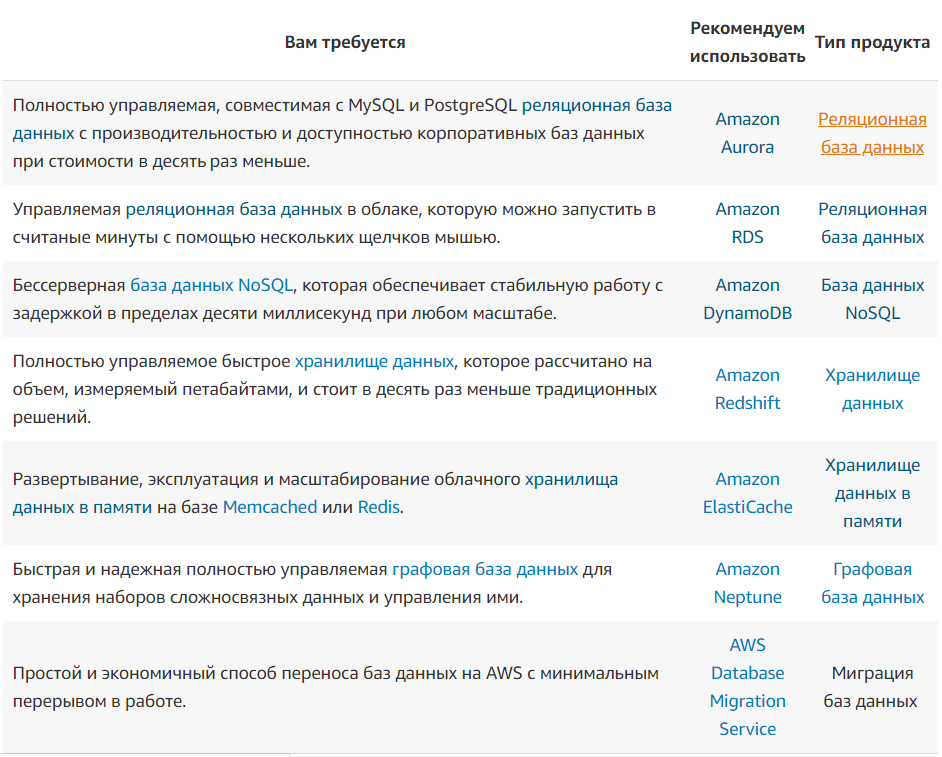


Рисунок 17. AWS Database Services

1. Amazon Aurora – это совместимая с MySQL и PostgreSQL реляционная база данных, созданная для облака и совмещающая в себе скорость и доступность самых современных коммерческих баз данных с простотой и экономичностью баз данных с открытым исходным кодом.

**Преимущества:**

1. Высокая производительность и масштабируемость

Aurora работает в 5 раз быстрее стандартной БД MySQL, в 3 раза быстрее БД PostgreSQL. Производительность соответствует уровню коммерческих баз данных, а стоимость – в десять раз меньше. По мере изменения потребностей можно легко масштабировать развертывание базы данных.

1. Высокая доступность и надежность

Aurora обеспечивает доступность на уровне более 99,99 %. Она использует отказоустойчивое самовосстанавливающееся хранилище, созданное для облака. Amazon Aurora непрерывно выполняет резервное копирование данных, незаметно восстанавливается при сбоях в работе физического хранилища.

1. Высокий уровень безопасности

Amazon Aurora обеспечивает несколько уровней защиты базы данных. Среди них сетевая изоляция, шифрование данных при хранении ключами шифрования, создаваемыми и управляемыми с помощью AWS Key Management Service (KMS), и шифрование данных при передаче с использованием SSL

1. Совместимость с MySQL и PostgreSQL

Ядро базы данных Amazon Aurora полностью совместимо с MySQL и PostgreSQL; совместимость с новыми выпусками регулярно добавляется.

1. Полностью управляемая БД

Amazon Aurora работает под полным управлением сервиса Amazon Relational Database Service (RDS). Aurora автоматически и непрерывно выполняет мониторинг базы данных и делает резервные копии, что обеспечивает точное восстановление на заданный момент времени.

1. Поддержка миграции

Базы данных MySQL и PostgreSQL можно переносить в Aurora с помощью AWS Database Migration Service, который обеспечивает безопасную миграцию с минимальным временем простоя.

1. Amazon RDS (Relational Database Service) обеспечивает легкую настройку, работу и масштабирование баз данных MySQL, Oracle, SQL Server или PostgreSQL в облаке.

Сервис обеспечивает экономичное и масштабируемое использование ресурсов при одновременной автоматизации трудоемких задач администрирования, таких как выделение аппаратного обеспечения, настройка базы данных, установка исправлений и резервное копирование. Это позволяет сосредоточить внимание на приложениях, чтобы обеспечить для них высокую производительность, высокую доступность, безопасность и совместимость.

**Преимущества:**

1. Простота администрирования

Сервис Amazon RDS позволяет просто перейти от первичного проекта к развертыванию. При этом не нужно беспокоиться о выделении инфраструктуры или установке и обслуживании программного обеспечения для баз данных.

1. Высокая масштабируемость

Можно масштабировать вычислительные ресурсы и ресурсы памяти, обеспечивающие работу системы, уменьшая или увеличивая их объем до максимально возможных 32 виртуальных ЦП и 244 ГиБ оперативной памяти. Масштабирование вычислительных ресурсов, как правило, занимает всего несколько минут.

1. Доступность и надежность

Amazon RDS обеспечивает автоматическое резервное копирование, снимки состояния БД, В любой момент из снимка состояния БД можно создать новый инстанс. В то время как снимки состояния БД могут служить полноценными резервными копиями.

1. Скорость

Доступное в сервисе Amazon RDS универсальное хранилище на базе SSD обеспечивает не менее трех операций ввода-вывода в секунду (IOPS) на каждый выделенный гигабайт и позволяет добиться пикового значения в 3000 IOPS ( объемом операций ввода-вывода в секунду ).

1. Безопасность

Сервис Amazon RDS упрощает управление сетевым доступом к базе данных. Amazon RDS также позволяет запускать инстансы баз данных в [облаке](https://aws.amazon.com/ru/vpc/)  (Amazon VPC), чтобы изолировать их и подключаться к существующей ИТ‑инфраструктуре с использованием VPN со стандартным шифрованием IPsec.

1. Экономичность

Плата начисляется по очень низким тарифам и только за ресурсы, которые реально используются.

1. Amazon DynamoDB – хорошо масштабируемый и полностью управляемый сервис базы данных NoSQL.

Этот сервис благодаря своей бесперебойной пропускной способности, масштабированию хранилища, а также автоматической трехпотоковой репликации освобождает вас от задач администрирования баз данных, требующих много времени, и позволяет сосредоточиться на приложении и бизнесе.

Amazon DynamoDB – это **нереляционная** база данных, обеспечивающая надежную производительность при любом масштабе. База данных является полностью управляемой и работает в нескольких регионах с несколькими ведущими серверами. Она обеспечивает устойчивую задержку в пределах нескольких миллисекунд и обладает встроенными средствами безопасности, резервного копирования и восстановления, а также кэширования в памяти.

**Преимущества:**

1. Производительность в любом масштабе
2. Полностью управляемая БД
3. Решение корпоративного уровня
4. DynamoDB обеспечивает устойчивый отклик на уровне нескольких миллисекунд при любом масштабе. Это позволяет создавать приложения с практически неограниченными пропускной способностью и хранилищем. Можно добавить кэширование в памяти, которое снижает время отклика с миллисекунд до микросекунд, без изменений в приложении.
5. DynamoDB – это бессерверная база данных, которая автоматически масштабирует пропускную способность и непрерывно создает резервные копии в целях защиты данных. DynamoDB предоставляет глобально распределенным приложениям быстрый доступ к локальным данным.

Создано для критически важных рабочих нагрузок. Безопасность данных гарантируется шифрованием, а доступность – соглашением об уровне обслуживания. Сервис обеспечивает полное управление таблицами с тщательным контролем доступа, встроенными средствами мониторинга и поддержкой частных подключений по VPN

1. Amazon Redshift – это быстрое масштабируемое хранилище данных, которое позволяет проще и экономичнее анализировать все данные в хранилище и в озере данных

Благодаря технологиям машинного обучения, массово‑параллельному выполнению запросов и использованию столбчатых хранилищ на высокопроизводительных дисках. Производительность Redshift оказывается в десять раз выше, чем производительность других хранилищ данных. Сервис позволяет за считанные минуты настроить новое хранилище данных и выполнить его развертывание, а затем сразу выполнять запросы к петабайтам данных хранилища Redshift ..

**Преимущества:**

1. Повышенная производительность

Производительность Amazon Redshift в десять раз выше, чем производительность других хранилищ данных. В этом сервисе используется машинное обучение, массово‑параллельная архитектура, оптимизированное для вычислений аппаратное обеспечение и кэширование результирующих наборов данных для обеспечения высокой пропускной способности и времени ответа на уровне долей секунды.

1. Простая настройка, развертывание и управление

Сервис Amazon Redshift прост в использовании и позволяет выполнить развертывание нового хранилища данных за считанные минуты. Redshift обеспечивает автоматизацию большинства стандартных административных задач, связанных с мониторингом и масштабированием хранилища данных, а также с управлением.

1. Экономичность

Amazon Redshift является наиболее экономичным облачным хранилищем данных: его использование обходится в десять раз дешевле традиционных локальных хранилищ данных. Redshift не предусматривает авансовых платежей: оплата начисляется только по факту использования. Работу с сервисом можно начать по тарифу 0,25 USD в час без каких‑либо обязательств, а затем выйти на уровень всего 250 USD за терабайт данных в год.

1. Быстрое масштабирование в зависимости потребностей

Amazon Redshift обеспечивает масштабирование и позволяет выполнять запросы к гигабайтам и эксабайтам данных в хранилище данных Redshift и озере данных на базе Amazon S3.В Redshift, по мере изменения потребностей можно осуществлять масштабирование в любом направлении.

1. Запросы к озеру данных

Amazon Redshift расширяет возможности хранилища данных до озера данных и позволяет получать уникальную аналитическую информацию, которую невозможно получить из отдельных хранилищ данных.

1. Безопасность

На базе Amazon Redshift выполняются критически важные рабочие нагрузки крупных финансовых сервисов, учреждений здравоохранения, организаций розничной торговли и правительственных организаций. Amazon Virtual Private Cloud (VPC) помогает изолировать кластеры.

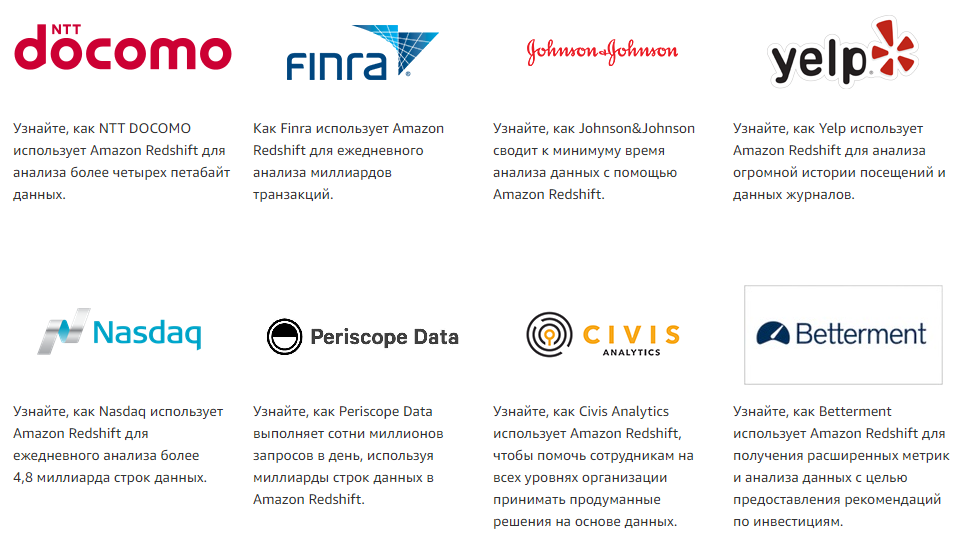


Рисунок 18. Компании, использующие Amazon Redshift

1. Amazon ElastiCache – это веб-сервис, который упрощает развертывание и масштабирование кэша в памяти в облаке, а также управление им.

Amazon ElastiCache предлагает полностью управляемые среды Redis и Memcached. Эффективное развертывание и масштабирование популярных хранилищ данных в памяти, совместимых с решениями с открытым исходным кодом, а также управление ими. Amazon ElastiCache – популярное решение для игровых и рекламных приложений, финансовых сервисов, приложений в сфере здравоохранения и Интернета вещей.\

**Преимущества:**

1. Высочайшая производительность

Amazon ElastiCache работает как хранилище данных в памяти и система кэширования, обеспечивая поддержку самых ресурсоемких приложений. С помощью оптимизированного комплексного стека, работающего на выделенных клиенту узлах, Amazon ElastiCache обеспечивает безопасность и высочайшую производительность.

1. Полностью управляемая БД

Он полностью освобождает пользователя от задач управления оборудованием. ElastiCache непрерывно выполняет мониторинг кластеров, чтобы обеспечить бесперебойное функционирование рабочих нагрузок и позволить пользователям сосредоточиться на более важных задачах по разработке приложений.

1. Масштабируемость

Amazon ElastiCache поддерживает горизонтальное масштабирование. Масштабирование операций записи и масштабирование памяти поддерживаются за счет сегментирования.

1. Amazon Neptune – это быстрый, надежный, полностью управляемый сервис графовой базы данных, который упрощает создание и запуск приложений, работающих с наборами сложносвязанных данных.

В основе Amazon Neptune лежит специально созданное высокопроизводительное ядро графовой базы данных, оптимизированное для хранения миллиардов взаимосвязей и выполнения запросов к графу с задержками на уровне миллисекунд. Amazon Neptune поддерживает популярные модели графов, а также их соответствующие языки запросов, что позволяет легко создавать запросы для эффективной навигации по наборам сложносвязанных данных. Neptune лежит в основе таких примеров использования графов, как сервисы рекомендаций, системы выявления мошенничества, графы знаний, разработка лекарственных препаратов и сетевая безопасность.

**Преимущества:**

1. Поддержка API Open Graph

Amazon Neptune поддерживает API Open Graph для Gremlin и SPARQL и обеспечивает высокую производительность для каждой из этих графовых моделей и их языков запросов.

1. Высокая производительность и масштабируемость

Amazon Neptune – это специально созданная высокопроизводительная графовая база данных. Она оптимизирована для обработки запросов графов. Neptune поддерживает выполнение более ста тысяч запросов графов в секунду. По мере изменения потребностей можно легко масштабировать развертывание базы данных в нужном направлении.

1. Высокая доступность и надежность

Amazon Neptune обладает высокой доступностью, надежностью и совместимостью с принципом ACID (атомарность, непротиворечивость, изолированность, долговечность). В нем используется отказоустойчивое самовосстанавливающееся хранилище, созданное для облака. Neptune непрерывно выполняет резервное копирование данных в Amazon S3 и незаметно восстанавливается при сбоях в работе физического хранилища.

1. Высокий уровень безопасности

Amazon Neptune обеспечивает несколько уровней безопасности для базы данных, включая сетевую изоляцию с помощью Amazon VPC и шифрование данных при хранении с помощью ключей, создаваемых и управляемых через AWS Key Management Service (KMS). В зашифрованном инстансе Neptune шифруются данные базового хранилища, а также автоматически создаваемые резервные копии, снимки состояния и реплики, находящиеся в том же кластере.

Neptune автоматически и непрерывно выполняет мониторинг базы данных и делает резервные копии в Amazon S3, что обеспечивает точное восстановление на заданный момент времени.

1. Полностью управляемый
2. AWS Database Migration Service помогает просто и безопасно выполнить миграцию баз данных на платформу AWS.

Исходная база данных во время миграции остается в полностью рабочем состоянии, что сводит к минимуму время простоя использующих ее приложений. AWS Database Migration Service позволяет выполнить миграцию данных.

**Преимущества**

1. Простота использования

Сервис AWS Database Migration Service прост в использовании. Ничего не требуется для установки и настройки. Чтобы начать миграцию базы данных, достаточно нескольких щелчков мышью в Консоли управления AWS.

1. Минимальные простои

Migration Service позволяет осуществить миграцию баз данных на платформу AWS практически без простоев. Все изменения, вносимые в исходную базу данных во время миграции, непрерывно реплицируются в целевую базу данных, при этом исходная база данных остается в полностью рабочем состоянии.

1. Поддержка большинства распространенных баз данных

AWS Database Migration Service позволяет выполнять миграцию данных, используя в качестве базы-источника и целевой базы большинство распространенных баз данных, как коммерческих, так и с открытым исходным кодом. Сервис поддерживает как однородные миграции, например из Oracle в Oracle, так и разнородные миграции между различными платформами баз данных, например из Oracle в Amazon Aurora. Кроме того, сервис может переносить данные между SQL, NoSQL и текстовыми целевыми базами.

1. Низкая стоимость

AWS Database Migration Service является экономичным сервисом. Вы платите только за вычислительные ресурсы, используемые в процессе миграции, и за хранение любых дополнительных журналов. Миграция базы данных объемом в один терабайт обойдется вам всего в 3 USD.

1. Быстрая и простая настройка

С помощью Консоли управления AWS можно всего за несколько минут настроить задание миграции. В задании миграции определяются параметры, которые будет использовать AWS Database Migration Service во время миграции.

1. Надежность

Сервис обладает высокой отказоустойчивостью и способностью к самовосстановлению. Он осуществляет постоянный мониторинг исходной и целевой баз данных, сетевого подключения и инстанса репликации. В случае прерывания процесса сервис автоматически перезапустит его и продолжит миграцию с того места, где она остановилась.

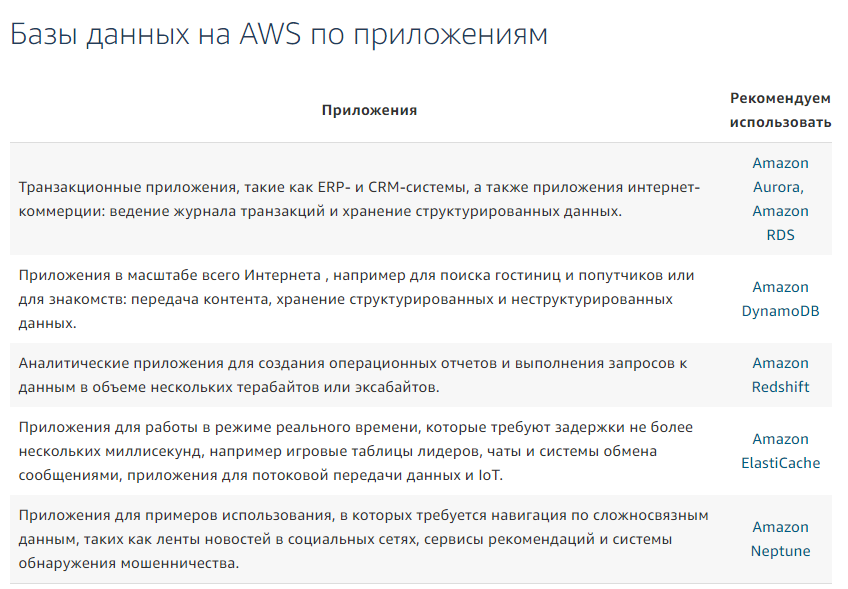


Рисунок 19. Базы данных на AWS по приложениям

# Использованные источники

1. <https://aws.amazon.com/ru/docker/>
2. <https://aws.amazon.com/ru/containers/>
3. <https://www.8host.com/blog/sozdanie-klastera-kontejnerov-docker-s-pomoshhyu-docker-swarm-v-ubuntu-16-04/>
4. <https://habr.com/post/261415/>
5. <https://docs.docker.com/compose/>
6. <https://habr.com/post/322440/>
7. <https://severalnines.com/resources/tutorials/mysql-load-balancing-haproxy-tutorial#what>
8. <http://devacademy.ru/posts/haproxy/>
9. <https://aws.amazon.com/ru/s3/>
10. <https://aws.amazon.com/ru/s3/faqs/>
11. <https://habr.com/company/technoserv/blog/329338/>
12. <https://mcs.mail.ru/storage/>
13. <https://aws.amazon.com/ru/products/databases/>