1. Технологические и иные предпосылки появления облачных вычислений

**Облачные вычисления** представляют собой динамически масштабируемый способ доступа к внешним вычислительным ресурсам в виде сервиса, предоставляемого посредством Интернета, при этом пользователю не требуется никаких особых знаний об инфраструктуре "облака" или навыков управления этой "облачной" технологией​.

Предпосылки:

**Рост:**

* производительности процессоров (CPU, GPU);
* Пропускной способности компьютерных сетей;
* Ёмкости оперативной и дисковой памяти.

**Виртуализация**

•VmWare, Microsoft, Citrix

**Интернет-сервисы**

•Amazon, SalesForce, Google

**Инструменты управления вычислительными ресурсами**

* Управление мощностями (перераспределение и выделение
* ресурсов
* **Сервисный подход к управлению ИТ**
* •ITIL, SLA
* **Сервисный подход к созданию ИС**
* •SOA, web services
* **Технологическое развитие**
* •Каналы связи
* •Интеллектуальные: сервера, сетевые устройства, устройства хранения данных

1. Основные характеристики (признаки) облачных вычислений по NIST.

Облачные вычисления – это модель предоставления повсеместного, удобного  и доступного по запросу доступа к разделяемому пулу конфигурируемых компьютерных ресурсов (виртуальных машин, систем хранения данных, приложений и сервисов), которые  могут быть быстро подготовлены и предоставлены  пользователю с минимальными усилиями управления  со стороны поставщика облачных услуг или взаимодействия с ним.​

* **Самообслуживание по требованию.**​

Потребитель по мере необходимости автоматически, без взаимодействия с поставщиком услуг, может самостоятельно определять и изменять вычислительные мощности, такие как серверное время, объем хранилища данных;​

* **Широкий (универсальный) сетевой доступ.**​

Вычислительные возможности доступны по сети через стандартные механизмы, что способствует широкому использованию разнородных (тонких или толстых) платформ клиента (терминальных устройств);​

* **Объединение ресурсов в пулы**. Конфигурируемые вычислительные и иные ресурсы поставщика объединены в единый пул для совместного использования распределенных ресурсов большим количеством потребителей;​
* **Мгновенная эластичность ресурсов**(мгновенная масштабируемость). Облачные услуги могут быстро предоставляться, расширяться, сжиматься и освобождаться исходя из потребностей потребителя; ​
* **Измеряемый сервис**(учет потребляемого сервиса и возможность оплаты реально использованных услуг. ​

1. Модели развертывания облаков: частное облако, публичное облако, гибридное облако, общественное облако.

Private cloud (частное облако) - инфраструктура, предназначенная для использования облачных вычислений в масштабе одной организации;

Community cloud (облако сообщества) - облачная инфраструктура, которая предназначена для исключительного использования облачных вычислений определенным сообществом потребителей от организаций, которые решают общие проблемы;

* Public cloud (публичное облако) - инфраструктура, предназначенная для свободного использования облачных вычислений широкой публикой;
* Hybrid cloud (гибридное облако) - это комбинация различных облачных инфраструктур (частных, публичных или сообществ), остающихся уникальными объектами, но связанных между собой стандартизованными или частными технологиями, которые обеспечивают возможность обмена данными и приложениями.

|  |  |  |
| --- | --- | --- |
| **Private**​ | **Частное**​ | 1. Реализация модели облачных вычислений на ресурсах, имеющихся в распоряжении одной компании, для обслуживания внутренних потребителей ​  2. Облачная инфраструктура функционирует целиком в целях обслуживания одной организации. Инфраструктура может управляться самой организацией или третьей стороной и может существовать как на стороне потребителя так и у внешнего провайдера. ​ |
| **Community**​  ​ | **Комму-нальное**​ | Облачная инфраструктура используется совместно несколькими организациями и поддерживает ограниченное сообщество, разделяющими общие принципы (например, миссию, требования к безопасности, политики, требования к соответствию <регламентам и руководящим документам>). Такая облачная инфраструктура может управляться самими организациями или третьей стороной и может существовать как на стороне потребителя так и у внешнего провайдера.​ |
| **Hybrid**​ | **Гибридное**​ | Облачная инфраструктура является композицией (сочетанием) двух и более облаков (частных, общих или публичных), остающихся уникальными сущностями, но объединенных вместе стандартизированными или частными (проприетарными) технологиями, обеспечивающими портируемость данных и приложений между такими облаками (например, такими технологиями, как пакетная передача данных для баланса загрузки между облаками). ​ |
| **Public**​ | **Публич-ное**​ | Облачная инфраструктура создана в качестве общедоступной или доступной для большой группы потребителей не связанной общими интересами, но, например, принадлежащими к одной области деятельности\*. Такая инфраструктура находится во владении организации, продающей соответствующие облачные услуги/ предоставляющей облачные сервисы. ​ \*) принадлежность к одной области деятельности/ индустрии может предполагать специфичные для этой индустрии приложения, потребность в которых испытывают организации, ведущие аналогичную деятельность или работающие на одном рынке. ​ |

1. Модели предоставления услуг облачных вычислений: Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS).

IaaS - это предоставление компьютерной инфраструктуры как услуги на основе концепции облачных вычислений.

IaaS состоит из трех основных компонентов:

* Аппаратные средства (серверы, системы хранения данных, клиентские системы, сетевое оборудование)
* Операционные системы и системное ПО (средства виртуализации, автоматизации, основные средства управления ресурсами)
* Связующее ПО (например, для управления системами)

IaaS основана на технологии виртуализации, позволяющей пользователю оборудования делить его на части, которые соответствуют текущим потребностям бизнеса, тем самым увеличивая эффективность использования имеющихся вычислительных машин.

IaaS избавляет предприятия от необходимости поддержки сложных инфраструктур центров обработки данных, клиентских и сетевых инфраструктур, а также позволяет уменьшить связанные с этим капитальные затраты и текущие расходы.

Примеры. Первопроходцами в IaaS считается компания Amazon, которые на сегодняшний день предлагают два основных IaaS-продукта: EC2 ( **Elastic Compute Cloud** ) и S3 ( **Simple Storage Service** ).

**Salesforce.com**​

**Microsoft Office 365**​

**Box**​

**Google Apps**​

**Amazon Web Services**​

**Concur**​

**Zendesk**​

**Zendesk**​

**DocuSign**​

**Dropbox**​

**Slack**​

*PaaS - это предоставление интегрированной платформы для разработки, тестирования, развертывания и поддержки веб-приложений как услуги*.

Такой подход имеет следующие достоинства:

* масштабируемость;
* отказоустойчивость;
* виртуализация;
* безопасность.

Примеры.

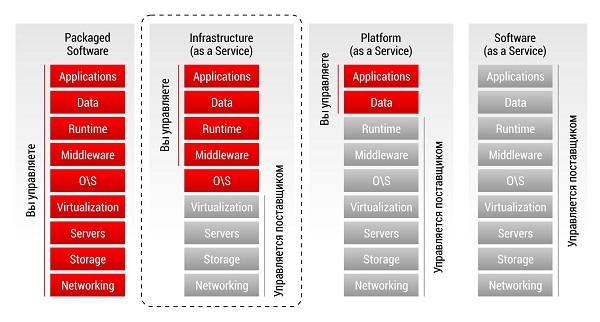
Самым известным примером такой платформы является AppEngine от Google​

* Продукты компании Mosso:​
* Cloud Sites — веб-хостинг (Linux, Windows, Mail) для нагрузочных веб-проектов с возможностью расширять базовые бесплатные — возможности за дополнительную плату (трафик, хранилище данных, вычислительная мощность).​
* Cloud Files — файловый cloud-хостинг с ежемесячной погигабайтной оплатой за объем хранимых файлов.

**SaaS** – модель развертывания приложения, которая подразумевает предоставление приложения конечному пользователю как услуги по требованию (on demand).

В модели SaaS:

* приложение приспособлено для удаленного использования;
* одним приложением могут пользоваться несколько клиентов;
* оплата за услугу взимается либо как ежемесячная абонентская плата, либо на основе суммарного объема транзакций;
* поддержка приложения входит уже в состав оплаты;



1. Основные преимущества и недостатки моделей развертывания и предоставления облачных вычислений и предлагаемых на их основе решений.

Основные преимущества и достоинства «облачных вычислений»:

* доступность вычислений, ресурсов;
* отказоустойчивость среды;
* «непритязательность» к ресурсам компьютера («не более, чем для работы в Интернет»);
* устойчивость к потере данных;
* надежность;
* экономичность;
* эффективность;
* простота;
* организация совместной работы;
* открытый интерфейс;
* гибкость и масштабируемость;
* производительные вычисления;
* развитые средства хранения данных;
* «инструментальность» и другие.

Недостатки «облачных вычислений»:

* постоянная потребность в сетевом (Интернет) соединении;
* недостаточная безопасность при работе;
* недоступность некоторых приложений («малая функциональность»);
* зависимость от облачного провайдера;
* недостаточность профессионалов в области «облачных вычислений»;
* отсутствие надежных ЦОД (центров обработки данных) и другие.

1. Системы и сети хранения данных. Основные виды топологий сетей хранения данных.

*SAN - это высокоскоростная коммутируемая сеть передачи данных, объединяющая серверы, рабочие станции, дисковые хранилища и ленточные библиотеки. Обмен данными происходит по протоколу Fibre Channel, оптимизированному для быстрой гарантированной передачи сообщений и позволяющему передавать информацию на расстояние от нескольких метров до сотен километров*.

1. Однокоммутаторная структура (англ. single-switch fabric) состоит из одного коммутатора Fibre Channel, сервера и системы хранения данных. Обычно эта топология является базовой для всех стандартных решений — другие топологии создаются объединением однокоммутаторных ячеек.
2. Каскадная структура— набор ячеек, коммутаторы которых соединены в дерево с помощью межкоммутаторных соединений.
3. Решетка — набор ячеек, коммутатор каждой из которых соединен со всеми другими. При отказе одного (а в ряде сочетаний — и более) соединения связность сети не нарушается. Недостаток — большая избыточность соединений
4. Кольцо— практически повторяет схему топологии решётка. Среди преимуществ — использование меньшего количества соединений.
5. Развитие процессорных и сетевых технологий за последние 25 лет и их влияние на технологии облачных вычислений.
6. Программное обеспечение облачных вычислений с открытым кодом. Достоинства и недостатки.

Достоинства ПО с открытым кодом​

Благодаря наличию открытых стандартов и большому числу программных интерфейсов к приложениям с открытым кодам эти приложения могут легко взаимодействовать друг с другом, что создает основу для выбора приложений и построению на их основе устойчивых решений для облачных вычислений на предприятиях.

Причины использования ПО с открытым кодом для облачных вычислений:​

* с целью экономии средств на приобретение ПО​
* проекты облачных вычислений с открытым кодом имеют широкую аудиторию пользователей​
* проекты с открытым кодом, как правило, всегда являются инновационными с агрессивным циклом выпуска релизов (версий), которые связаны с продвижением новых технологий

Примером приложений  СОК и решений на их основе являются:​

* Salesforce.com;​
* GoogleDocs;​
* RedHatNetwork;​
* VMWare Cloud Foundrg;​
* Google AppEngine;​
* Windows Azure;​
* Rackspace Sites;​
* RedHat OpenShift;​
* Active StateStakato;​
* AppFog;​
* EC2 (Amazon);​
* Rackspase Cloud Files;​
* OpenStack;​
* Eucalip tus;​
* OpenNebula и мн. др.

1. Системы облачных вычислений с открытым кодом OpenStack и OpenNebula. Сравнение функциональных возможностей этих систем.

Среди всех предлагаемых реализаций IaaS, OpenStack является одним из двух продуктов, которые способны поддерживать несколько моделей облачных вычислений (IaaS, PaaS или SaaS).

Глобальный проект OpenStack был основан компанией Rackspace и NASA , т.е. теми, кто создал широко масштабируемую систему облачных вычислений, свободно доступную по лицензии Apache 2.0.

* OpenStack не требует никакого проприетарного (коммерческого) аппаратного и программного обеспечения. ​
* Это ПО было создано для работы, как с полностью виртуализированными системами, так и системами, развернутыми  на "голом железе".​
* OpenStack поддерживает несколько гипервизоров, включая KVM и XenServer, а также технологию контейнеров, вк лючая LXC. ​

Основные компоненты OpenStack:

* Nova - контроллер вычислительных ресурсов (аналог Amazon EC2);
* Glance - библиотека образов виртуальных машин, обычно с бэкендом в Swift;
* Swift - облачное файловое хранилище (аналог Amazon S3);
* Cinder  - служба работы с блочными устройствами хранения данных (выведена из Nova в отдельный проект);
* Keystone - сервис идентификации;
* Ceilometer - средства сбора, нормализации и трансформации данных, предоставляемых сервисами OpenStack. Собираемые данные используются для реализации различных сценариев реагирования на события;
* Trove - база данных;
* Sahara - Elastic Map Reduce;
* Ironic - средства управления и провижининга физическими серверами (Bare Metal Provisioning);

OpenNebula

Это ПО является комбинацией функционального и исследовательского проектов.

Функциональность продукта:

* интуитивно самообслуживаемый портал,
* автоматизируемый каталог управления сервисами и интерфейсы супер пользователя,
* управление производительностью и объемом данных,
* реализация функций:
* высокой готовности,
* непрерывности бизнеса,
* управление виртуальной инфраструктурой,
* безопасность уровня предприятия,
* интеграция с инструментами других производителей и
* превосходная поддержка продукта (коммерческая).
* Поддерживает гипервизоры KVM и ESXi.

1. Понятие виртуализации. Типы виртуализации. Виртуализация физического сервера.

*В компьютерных технологиях под термином "виртуализация" обычно понимается абстракция вычислительных ресурсов и предоставление пользователю системы, которая "инкапсулирует" (скрывает в себе) собственную реализацию*.

Проще говоря, пользователь работает с удобным для себя представлением объекта, и для него не имеет значения, как объект устроен в действительности.

**Преимущества виртуализации:**

1. **Эффективное использование вычислительных ресурсов**
2. **Сокращение расходов на инфраструктуру**
3. **Снижение затрат на программное обеспечение**.
4. **Повышение гибкости и скорости реагирования системы**.
5. **Несовместимые приложения могут работать на одном компьютере**
6. **Повышение доступности приложений и обеспечение непрерывности работы предприятия**
7. **Возможности легкой архивации**
8. **Повышение управляемости инфраструктуры**

Существуют следующие виды (способы) виртуализации, такие как:

* виртуализация серверов:
* виртуализация на уровне операционных систем,
* виртуализация приложений,
* виртуализация представлений.

Виртуализация серверов может быть двух видов:

* + полная виртуализация
  + паравиртуализация

Виртуализация серверов подразумевает запуск на одном физическом сервере нескольких виртуальных серверов. Виртуальные машины или сервера представляют собой приложения, запущенные на хостовой операционной системе, которые эмулируют физические устройства сервера. На каждой виртуальной машине может быть установлена операционная система, на которую могут быть установлены приложения и службы.

Имеется несколько способов реализации виртуализации  в зависимости от типа интерфейса компьютерной системы, используемого для этого:​

* Виртуализация на уровне набора команд (ISA). Используется интерфейс между аппаратными средствами и ПО, который представляет собой набор машинных команд;​
* Виртуализация на уровне системных вызовов (эмуляция системных вызовов библиотек функций ОС);​
* Виртуализация на уровне библиотечных вызовов (на уровне API) приложений.​

1. История развития технологий виртуализации.

Компания IBM была первой. В 60-х годах прошлого века виртуализация представляла чисто научный интерес и была оригинальным решением для изоляции компьютерных систем в рамках одного физического компьютера.

В 1999 г. компания VMware представила технологию виртуализации систем на базе x86 (Vmware WorkStstion)

Позднее в "битву“ включились такие компании как Parallels (ранее SWsoft), Oracle (Sun Microsystems), Citrix Systems (XenSourse)

Корпорация Microsoft вышла на рынок средств виртуализации в 2003 г. с приобретением компании Connectiх, выпустив свой первый продукт Virtual PC для настольных ПК.

1. Требования к архитектуре ЭВМ для поддержки виртуализации. Проблемы виртуализации платформы Intel x86

В 1974 году двое ученых  из Калифорнийского университета (Лос-Анджелес), работающих  в компьютерной сфере, Геральд  Попек  (Gerald Popek) и Роберт  Голдберг (Robert  Goldberg), опубликовали основополагающую статью («Formal Requirements for Virtualizable Third Generation Architectures»), в которой  дали точный перечень тех условий,  которым  должна отвечать компьютерная архитектура, чтобы иметь возможность эффективно поддерживать виртуализацию.​

* Такими требованиями являются:​
* 1.  **Безопасность**— у гипервизора должно  быть полное  управление виртуализированными ресурсами.​
* 2.  **Эквивалентность**— поведение программы на виртуальной машине  должно  быть идентичным поведению этой же программы, запущенной на реальном оборудовании. ​
* 3.  **Эффективность**— основная часть кода в виртуальной машине должна выполняться без вмешательства гипервизора.

Проблемы вмртуализации платформы Intel x86. Имеются команды способные изменить состояние процессора исполняемые в пользовательском режиме.

1. Принципы виртуализации. Способы виртуализации. Основные типы виртуализации.

Существуют следующие виды (способы) виртуализации, такие как:

* виртуализация серверов:
* виртуализация на уровне операционных систем,
* виртуализация приложений,
* виртуализация представлений.

Виртуализация серверов может быть двух видов:

* + полная виртуализация
  + паравиртуализация

Виртуализация серверов подразумевает запуск на одном физическом сервере нескольких виртуальных серверов. Виртуальные машины или сервера представляют собой приложения, запущенные на хостовой операционной системе, которые эмулируют физические устройства сервера. На каждой виртуальной машине может быть установлена операционная система, на которую могут быть установлены приложения и службы.

Виртуализация приложений подразумевает применение модели сильной изоляции прикладных программ с управляемым взаимодействием с ОС, при которой виртуализируется каждый экземпляр приложений, все его основные компоненты: файлы (включая системные), реестр, шрифты, INI-файлы, COM-объекты, службы.

Приложение исполняется без процедуры инсталляции в традиционном ее понимании и может запускаться прямо с внешних носителей (например, с флэш-карт или из сетевых папок).

* Виртуализация представлений подразумевает эмуляцию интерфейса пользователя. Т.е. пользователь видит приложение и работает с ним на своём терминале, хотя на самом деле приложение выполняется на удалённом сервере, а пользователю передаётся лишь картинка удалённого приложения. В зависимости от режима работы пользователь может видеть удалённый рабочий стол и запущенное на нём приложение, либо только само окно приложения.

Имеется несколько способов реализации виртуализации  в зависимости от типа интерфейса компьютерной системы, используемого для этого:​

* Виртуализация на уровне набора команд (ISA). Используется интерфейс между аппаратными средствами и ПО, который представляет собой набор машинных команд;​
* Виртуализация на уровне системных вызовов (эмуляция системных вызовов библиотек функций ОС);​
* Виртуализация на уровне библиотечных вызовов (на уровне API) приложений.​

Суть виртуализации – имитация поведения интерфейса компьютерной системы. Существуют два способа реализации такой имитации:​

* Средствами среды исполнения  приложения.​
* Средствами экранирования аппаратных средств машины  с помощью ПО преобразования в набор команд той же или другой архитектуры, используемого в качестве интерфейса. Это ПО получило название **VMM – Virtual Machine Monitor** (Монитор виртуальных машин).

1. Монитор виртуальных машин. Два подхода к реализации монитора виртуальных машин.

**Виртуальная машина** *— это полностью изолированный программный контейнер, который работает с собственной ОС и приложениями, подобно физическому компьютеру. Виртуальная машина действует так же, как физический компьютер, и содержит собственные виртуальные (т.е. программные) ОЗУ, жесткий диск и сетевой адаптер*.

* Идея монитора виртуальных машин не нова. Она была предложена еще в 1974 г. в работе Г. Popek, Р. Goldberg, «Formal Requirements for Virtualizable Third Generation Architectures».​
* Монитор виртуальных машин обеспечивает интерфейс между гостевой ОС и хозяйской ОС. При этом подавляющая часть кода приложения, гостевой ОС, а также хозяйской ОС выполняется напрямую на аппаратных средствах физической машины (все общие команды и часть привилегированных). ​
* И лишь небольшая часть кода, которую составляют некоторые привилегированные  команды перехватываются монитором виртуальных машин, модифицируются им, и затем передаются на исполнение аппаратным средствам.

Монитор виртуальных ​машин получил название​ Гипервизор.​ Различают: ​

гипервизоры I и II типов.​

Имеются два подхода к построению мониторов виртуальных машин:​

* **Эмуляция** всех инструкций, что влечет за собой резкое падение производительности. Этот подход в измененном виде используется совремными **мониторами виртуальных машин**.​
* Использование **паравиртуализации**, которая предполагает модификацию гостевых ОС, таким образом, чтобы нейтрализовать, либо полностью исключить влияние “чувствительных” команд, средствами хозяйской ОС.​

1. Гипервизор первого типа достоинства и недостатки.

На аппаратные средства устанавливается специализированная ОС (гипервизор) предназначенная для виртуализации на уровне набора команд аппаратной платформы (Vmware ESX).​ Такой монитор получил название “родного” (naitive) монитора виртуальных машин.​

Достоинства данной технологии заключаются в:​

* отсутствии потребности в хостовой ОС – ВМ, устанавливаются фактически на "голое железо", а аппаратные ресурсы используются более эффективно. ​

Недостатки:​

* необходимость поддержки в гипервизоре собственных драйверов внешних устройств, из-за чего возникают высокие дополнительные накладные расходы на используемые аппаратные ресурсы, ​
* отсутствие учета особенностей гостевых ОС, меньшая, чем нужно, гибкость в использовании аппаратных средств​

Используются не модифицированные экземпляры гостевых операционных систем, а для поддержки работы этих ОС служит общий слой эмуляции их исполнения поверх хостовой ОС, в роли которой выступает обычная операционная система.

Такая технология применяется, в частности, в VMware Workstation, VMware Server (бывший GSX Server), Parallels Desktop, Parallels Server, MS Virtual PC, MS Virtual Server (Hyper-V), Virtual Iron.​

Достоинства: ​

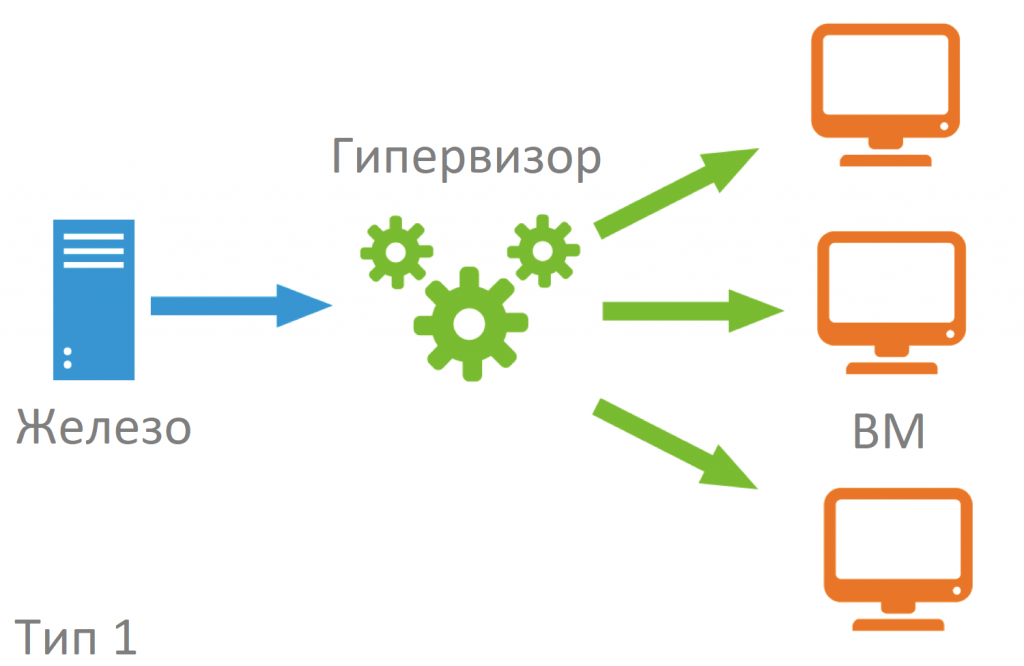
-  не требуется модификации гостевой ОС ;​

- «прозрачность» виртуализации.​

Недостатки :​

— высокие дополнительные накладные расходы на используемые аппаратные ресурсы, ​

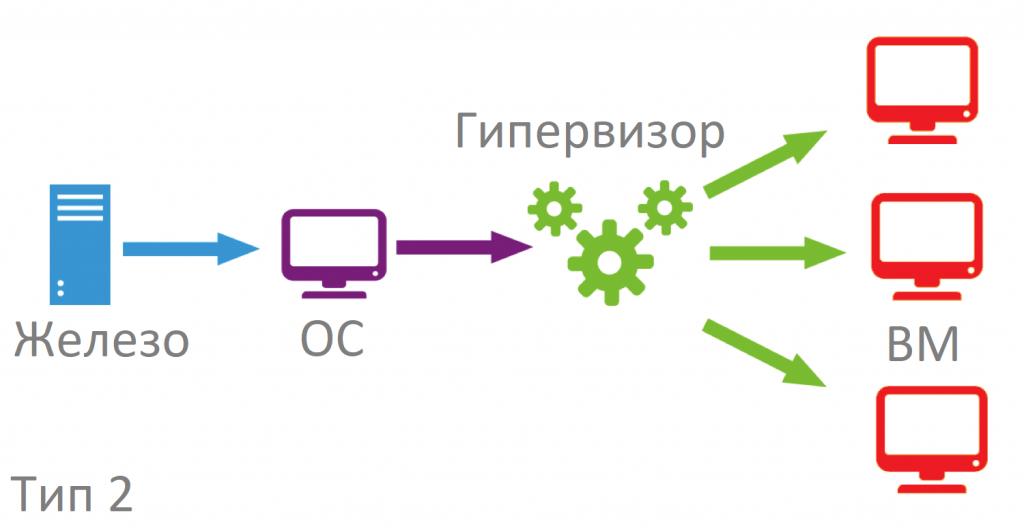
* отсутствие учета особенностей гостевых ОС, ​
* - меньшая, чем нужно, гибкость в использовании аппаратных средств​



1. Гипервизор 2-го типа достоинства и недостатки.

* Монитор виртуальных машин работает в рамках хозяйской ОС.​
* В процессе своей работы он модифицирует код гостевых ОС с целью обеспечения их виртуализации на уровне набора команд.​
* При этом код содержащий “чувствительные” команды переписывается на лету (Vmware WorkStation).​
* **Накладные расходы составляют 10-15%**​

Больше подходят для индивидуального использования, чем в качестве решений уровня предприятия.



1. Паравиртуализация. Достоинства и недостатки.

Используются не модифицированные экземпляры гостевых операционных систем, а для поддержки работы этих ОС служит общий слой эмуляции их исполнения поверх хостовой ОС, в роли которой выступает обычная операционная система.

* Модификация ядра гостевой ОС выполняется таким образом, что в нее включается новый набор API, через который она может напрямую работать с аппаратурой, не конфликтуя с другими виртуальными машинами. При этом нет необходимости задействовать полноценную ОС в качестве хостового ПО, функции которого в данном случае исполняет специальная система, получившая название гипервизора (hypervisor).

Достоинства данной технологии заключаются в отсутствии потребности в хостовой ОС – ВМ, устанавливаются фактически на "голое железо", а аппаратные ресурсы используются эффективно. ​

Недостатки — в сложности реализации подхода и необходимости создания специализированной ОС-гипервизора.

1. Виртуализация серверов. Виртуализация приложений. Виртуализация представлений. Виртуализация рабочих столов.

Существуют следующие виды (способы) виртуализации, такие как:

* виртуализация серверов:
* виртуализация на уровне операционных систем,
* виртуализация приложений,
* виртуализация представлений.

Виртуализация серверов может быть двух видов:

* + полная виртуализация
  + паравиртуализация

Виртуализация серверов подразумевает запуск на одном физическом сервере нескольких виртуальных серверов. Виртуальные машины или сервера представляют собой приложения, запущенные на хостовой операционной системе, которые эмулируют физические устройства сервера. На каждой виртуальной машине может быть установлена операционная система, на которую могут быть установлены приложения и службы.

Виртуализация приложений подразумевает применение модели сильной изоляции прикладных программ с управляемым взаимодействием с ОС, при которой виртуализируется каждый экземпляр приложений, все его основные компоненты: файлы (включая системные), реестр, шрифты, INI-файлы, COM-объекты, службы.

Приложение исполняется без процедуры инсталляции в традиционном ее понимании и может запускаться прямо с внешних носителей (например, с флэш-карт или из сетевых папок).

* Виртуализация представлений подразумевает эмуляцию интерфейса пользователя. Т.е. пользователь видит приложение и работает с ним на своём терминале, хотя на самом деле приложение выполняется на удалённом сервере, а пользователю передаётся лишь картинка удалённого приложения. В зависимости от режима работы пользователь может видеть удалённый рабочий стол и запущенное на нём приложение, либо только само окно приложения.

1. Система виртуализации Hyper-V. Осбенности архитектуры Hyper-V.

Hyper-v представляет собой гипервизор, т.е. прослойку между оборудованием и виртуальными машинами уровнем ниже операционной системы. Эта архитектура была первоначально разработана IBM в 1960-е годы для мэйнфреймов и недавно стала доступной на платформах x86/x64, как часть ряда решений, включая Windows Server 2008 Hyper-V и Vmware ESX.

**Hyper-V** поддерживает разграничение согласно понятию *раздел*. **Раздел** — логическая единица разграничения, поддерживаемая гипервизором, в котором работают ОС. Каждый экземпляр гипервизора должен иметь один *родительский раздел*, с запущенной Windows Server 2008. Стек виртуализации запускается на родительском разделе и обладает прямым доступом к аппаратным устройствам. Затем родительский раздел порождает *дочерние разделы*, на которых и располагаются гостевые ОС. Дочерний раздел также может породить собственные дочерние разделы. Родительский раздел создает дочерние при помощи API-*гипервизора*, представленного в Hyper-V.

Hyper-V обеспечивает высокую производительность, практически равную производительности одной операционной системы, работающей на выделенном сервере.

Преимущества:

Низкие затраты на оборудование и программное обеспечение. Благодаря технологии виртуализации серверов Microsoft Hyper-V, на каждом физическом сервере может быть запущено несколько виртуальных серверов, используя одну лицензию на Windows Server Enterprise или Datacenter.

Высокая доступность и отказоустойчивость (новый уровень SLA). Кластер Hyper-V автоматически восстанавливает свою работоспособность менее чем за 5 минут даже в случае полного отказа одного из серверов.

Масштабируемость (готовность к росту бизнеса). Если возникает задача развёртывания нового сервиса или приложения, вы просто за считанные минуты создаёте новую виртуальную машину и разворачиваете необходимые приложения.

Снижение затрат на обслуживание

1. Система виртуализации VMware.

В 1998 году VMware запатентовала свои программные техники виртуализации и с тех пор выпустила немало эффективных и профессиональных продуктов для виртуализации различного уровня: от VMware Workstation, предназначенного для настольных ПК, до VMware ESX Server, позволяющего консолидировать физические серверы предприятия в виртуальной инфраструктуре.

В процессорах на базе x86 содержатся 17 особых инструкций, создающих проблемы при виртуализации, из-за которых операционная система отображает предупреждающее сообщение, прерывает работу приложения или просто выдает общий сбой.

Для преодоления этого препятствия компания VMware разработала адаптивную технологию виртуализации, которая "перехватывает" данные инструкции на этапе создания и преобразует их в безопасные инструкции, пригодные для виртуализации

* **VMware Workstation** – платформа, ориентированная на desktop-пользователей и предназначенная для использования разработчиками ПО, а также проффесинальными пользователями.
* **VMware Player** – бесплатный "проигрыватель" виртуальных машин на основе виртуальной машины.
* **VMware ESX Server** – это гипервизор, разбивающий физические серверы на множество виртуальных машин. VMware ESX является основой пакета VMware vSphere и входит во все выпуски VMware vSphere.
* **VMware Fusion** – настольный продукт для виртуализации на платформе Mac от компании Apple.
* **VMware vSphere** – комплекс продуктов, представляющий надежную платформу для виртуализации ЦОД.
* VMware vSphere включает ряд компонентов, преобразующих стандартное оборудование в общую устойчивую среду,:
* Службы инфраструктуры — это компоненты, обеспечивающие всестороннюю виртуализацию ресурсов серверов, хранилищ и сетей, их объединение и точное выделение приложениям.
* Службы приложений — это компоненты, предоставляющие встроенные элементы управления независимо от их типа или ОС.
* VMware vCenter Server предоставляет центральную консоль , которая поддерживает масштабируемость для управления крупными средами ЦОД.

1. Система виртуализации Xen.

* Разработка некоммерческого гипервизора Xen начиналась как исследовательский проект компьютерной лаборатории Кембриджского университета. Основателем проекта и его лидером был Иан Пратт (Ian Pratt) сотрудник университета.​
* Изначально Xen представлял собой самую развитую платформу, поддерживающую технологию паравиртуализации. ​

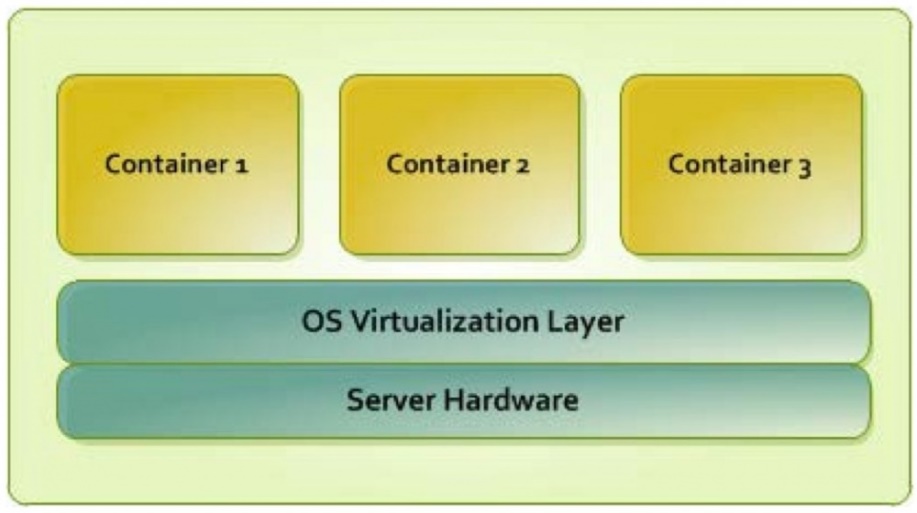
Эта технология позволяет гипервизору в хостовой системе управлять гостевой ОС посредством гипервызовов VMI (Virtual Machine Interface), что требует модификации ядра гостевой системы. На данный момент бесплатная версия Xen включена в дистрибутивы нескольких ОС, таких как Red Hat, Novell SUSE, Debian, Fedora Core, Sun Solaris. В середине августа 2007 года компания XenSource была поглощена компанией Citrix Systems.​

* Open Source версия платформы Xen применяется в основном в образовательных и исследовательских целях.​
* Версии Xen включаются в дистрибутивы многих Linux-систем, что позволяет их пользователям применять виртуальные машины для изоляции программного обеспечения в гостевых ОС с целью его тестирования и изучения проблем безопасности, без необходимости установки платформы виртуализации​

Xen идеально подходит для поддержки старого программного обеспечения в виртуальной машине. Для более же серьезных целей в производственной среде предприятия необходимо использовать коммерческие платформы компании Citrix.

1. Виртуализация на уровне ОС достоинства и недостатки.

 метод виртуализации, при котором ядро операционной системы поддерживает несколько изолированных экземпляров пространства пользователя, вместо одного. Эти экземпляры (часто называемые контейнерами или зонами) с точки зрения пользователя полностью идентичны реальному серверу.

Виртуализация на уровне ОС позволяет виртуализовывать физические серверы на уровне ядра операционной системы.   
  
  
  
Слой виртуализации ОС обеспечивает изоляцию и безопасность ресурсов между различными контейнерами. Слой виртуализации делает каждый контейнер похожим на физический сервер. Каждый контейнер обслуживает приложения в нем и рабочую нагрузку. 

**Основные преимущества контейнерной виртуализации**

* Контейнеры выполняются на одном уровне с физическими серверами. Отсутствие виртуализованного оборудования и использование реального оборудования и драйверов позволяют получить непревзойденную производительность.
* Каждый контейнер может масштабироваться до ресурсов целого физического сервера.
* Технология виртуализации на уровне ОС позволяет добиться высочайшей плотности, среди доступных среди решений виртуализации. Возможно создание и запуск сотен контейнеров на одном обычном физическом сервере.
* Контейнеры используют единую ОС, что делает их поддержку и обновление очень простым. Приложения могут быть также развернуты в отдельном окружении.

1. Эволюция распределенных систем и возникновение облачных сред.

Типы современных РС сформировались в процессе эволюции двух классов систем:

Систем высопроизводительных вычислений (HPC – High Performance Computing);

Систем высокой пропускной способности (HTC – High Throughput Computing);

Цели создания платформ облачных вычислений​

* Предоставление ресурсов по запросам пользователей обеспечивая:​
* массштабируемость;​
* виртуализацию ресурсов;​
* эффективность  использования ресурсов;​
* надежность предоставления ресурсов;​
* поддержку исполнения приложений Web 2.0;​
* безопасность предоставления разделяемых ресурсов и разделяемый доступ к ЦОДам.​

Впервые идею «облачных вычислений» озвучил Д. Ликлайдер в 1960 году. Его идея заключалась в том, что каждый человек на планете будет подключен к сети, из которой он будет получать не только данные, но и программы. Другой ученый, Джон Маккарти высказал идею о том, что вычислительные мощности будут предоставляться пользователям как услуга (сервис) [2].

В 90-е гг. происходит быстрое развитие глобальной сети — Интернет, оказывающее косвенное влияние на развитие облачных технологий. Значительно увеличилась пропускная способность сетей, расширилась география охвата. Наряду с развитием компьютерных сетей усовершенствовались аппаратные технологии, появились многоядерные процессоры, значительно увеличился объем хранилищ информации.

Появление первой технологии, близкой к современному пониманию термина «cloud computing», приписывается компании Salesforce.com, основанной в 1999 году. Данная компания стала первой компанией, предоставившей доступ к своему приложению через сайт, по сути, данная компания стала первой компанией, предоставившей свое программное обеспечение по принципу – программное обеспечение как сервис (SaaS). Следующим шагом стала разработка облачного веб-сервиса компанией Amazon в 2002 году. Данный сервис позволял хранить информацию и производить вычисления. В 2006 Amazon запустила сервис под названием Elastic Compute cloud (EC2) как веб-сервис который позволял его пользователям запускать свои собственные приложения. Следующим свою технологию постепенно ввела Google, начав с 2006 года предложение SaaS сервисов под названием «Google Apps», а затем и модели предоставления платформы как сервиса (PaaS) под названием “Google App Engine”. И, наконец, свое предложение анонсировала компания Microsoft, презентовав ее на конференции PDC в 2008 году под названием «Azure Services Platform»

1. GRID – как прообраз облачных систем. Концепция GRID.

GRID – Распределенная вычислительная среда, реализованная посредством развертывания основанной на стандартах инфраструктуры, которая поддерживает разделяемое использование ресурсов и координированное решение задач в рамках динамичных виртуальных организаций

Концепция

* Концепция грид-вычислений идея подобна  концепции электросети (англ. Power Grid):  нам не важно, откуда к нам в розетку приходит электричество. Независимо от этого мы можем подключить к электросети утюг, компьютер или стиральную машину. ​
* Аналогично и в идеологии грид: мы можем запустить любую задачу с любого компьютера или мобильного устройств на вычисление, ресурсы же для этого вычисления должны быть автоматически предоставлены на удаленных высокопроизводительных серверах, независимо от типа нашей задачи.​
  + Повсеместный, стандартный, надежный и дешевый доступ к ресурсам
  + Сложность инфраструктуры скрыта от пользователя (прозрачность)
  + Легкость интеграции новых ресурсов
  + Динамическое перераспределение нагрузки и отказоустойчивость
  + Управление сложным ансамблем ресурсов
* Три критерия определяющих что распределенная система является GRID:​
* Реализована координация использования ресурсов системы, при отсутствии централизованного управления этими ресурсами.​
* Построена на стандартных открытых протоколах и интерфейсах.​
* Обеспечивает показатели:​
* Высокая надежность и доступность.​
* Малое время отклика.​
* Большая пропускная способность.​

Проблемы решаемые с помощью GRID​

* Разделение ресурсов​
* Компьютеры, хранилища данных, приложения, приборы, сети, …​
* Гетерогенность ресурсов, механизмов безопасности и политик доступа​
* Условия предоставления ресурсов: согласование, учет, оплата, …​
* Координированное решение задач​
* Интеграция распределенных ресурсов​
* Обеспечение интегрального качества обслуживания​
* Динамические виртуальные организации​
* Новые структуры поверх обычных организационных структур​
* Отображение на локальные механизмы управления​

**Грид – это система, которая координирует распределенные ре**сурсы посредством стандартных, открытых, универсальных протоколов и интерфейсов для обеспечения нетривиального качества обслуживания (QoS – Quality of Service).

1. Область применения и спектр приложений GRID.

Приложения

* Высокопроизводительные вычисления
  + Интерактивное моделирование (климат)
  + Имитационные эксперименты с высоким разрешением / большим числом объектов (формирование галактик, гравитационные волны, имитация боевых действий)
  + Проектирование (прогон вариантов, объединение моделей компонентов)
* Работа с большими массивами данных
  + Анализ экспериментальных данных (физика высоких энергий)
  + Анализ изображений и данных зондирования (астрономия, изучение климата, экология)
* Совместная работа в рамках распределенных коллективов
  + Удаленное управление приборами (микроскопы, рентгеноскопия , …)
  + Удаленная визуализация (изучение климата, биология)
  + Engineering (large-scale structural testing, chemical engineering)
* Сложность задач требует совместной работы людей из различных организаций и разделяемого использования ресурсов, данных и приборов.

Классы приложений

* Распределенный суперкомпьютинг
  + очень большие задачи, требующие существенных вычислительных мощностей, процессорного времени, памяти и т.д.
* Высокопоточные вычисления
  + решение большого числа слабо связанных подзадач при помощи простаивающих вычислительных ресурсов
* Вычисления “по требованию”
  + удаленные ресурсы, временно интегрированные с локальными вычислениями
* Вычисления с привлечением больших объемов данных
  + вычисления с привлечением больших объемов распределенных данных
* Коллективные вычисления
  + поддержка взаимодействия и совместной работы нескольких удаленных участников.

1. Архитектура вычислительного кластера.

В общем случае, вычислительный кластер - это набор компьютеров (вычислительных узлов), объединенных некоторой коммуникационной сетью. Каждый вычислительный узел имеет свою оперативную память и работает под управлением своей операционной системы. Наиболее распространенным является использование однородных кластеров, то есть таких, где все узлы абсолютно одинаковы по своей архитектуре и производительности.

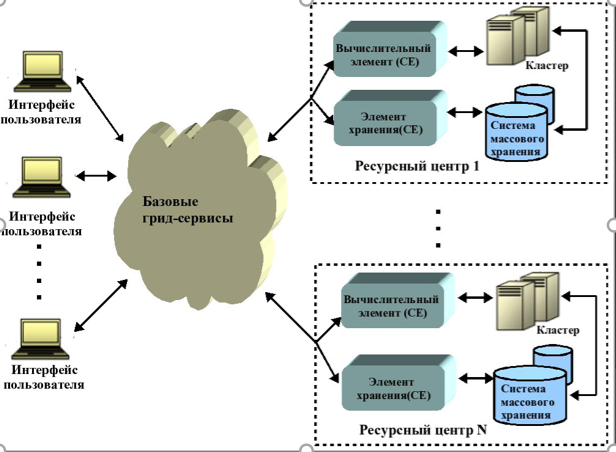


1. Архитектура GRID. SOA и Web-сервисы в современной архитектуре GRID. Виды Web-сервисов.

Ключевой проблемой grid является объединение ресурсов из различных организаций для совместного использования. В результате формируется федерация систем, которая влечет за собой образование виртуальных организаций.​

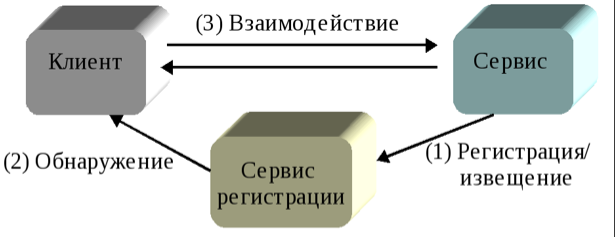
Для построения Grid была разработана и принята в качестве стандарта *OGSA (Open*Grid Services Architecture – Открытая архитектура грид-сервисов), основанная на архитектуре SOA – Service Oriented Architecture.

Основная  часть компонент этого промежуточного ПО построена на основе Веб-сервисных технологий и принципов OGSA.



**Сервис ориентированная архитектура (service-oriented architecture, SOA) – основа GRID**​.

SOA - является основой построения надежных распределенных систем, которые в качестве услуг предоставляют функциональные возможности, с дополнительным акцентом на слабые связи между взаимодействующими сервисами.​



* В определение SOA входит понятие слабой связи сервисов. Этот термин подразумевает, что взаимодействующие программные компоненты имеет минимальное знание друг о друге: ​
* они находят информацию, которая им нужна для взаимодействия непосредственно перед взаимодействием.​

**Веб-сервис** – веб-приложение, которое предоставляет программный интерфейс и пользователем является прогрмма, описание осущетсвяется с помощью wsdl. По сути, другая программа вызывает методы веб-сервиса.

*Эта программа может быть написана на любом языке и платформе.*

* SOA и веб-сервисы являются ≪ортогональными≫ понятиями:​
* сервисная ориентация – это архитектурный стиль построения системы, ​
* веб-сервисы – технология реализации сервиса. ​
* Они, конечно, могут использоваться совместно – как это часто и случается, но они взаимно независимы.​
* Веб-сервисы хорошо подходят в качестве строительных блоков SOA-среды​

Легкость, с которой веб-сервисы могут быть реализованы и возможность обращаться к ним локально или удаленно и независимо от платформы на которой работает клиент, привела к тому, что они были широко приняты администраторами систем как ≪агенты виртуализации≫, которые обеспечивают общие интерфейсы управления к различным ресурсам.​

На первом этапе своего существования грид-системы строились или на основе специально разработанных общедоступных компонент или на основе закрытых (проприетарных) технологий.

В дальнейшем стало ясно, что есть значительное перекрытие между целями​

вычислительного грида и преимуществ сред, основанных на SOA и веб-сервисах. ​

Быстрый прогресс в технологии веб-сервисов и разработке соответствующих стандартов обеспечили эволюционный путь от жесткой и узко-направленной архитектуры грид- систем первого поколения к стандартизированным, сервис-ориентированным гридам, гарантирующим стабильно-высокое качество обслуживания  пользователей (грид промышленного уровня)​

1. Микросервисная архитектура. Бессерверная технология создания и запуска приложений в облаке.

Микросервисы противопоставляются традиционной монолитной архитектуре. Монолит означает, что компоненты продукта взаимосвязаны и взаимозависимы.   Если перестает работать один - все остальные тоже «отваливаются». ​

* Предшественником микросервисной архитектуры является [сервис-ориентированная архитектура (SOA)](https://en.wikipedia.org/wiki/Service-oriented_architecture), которая также разделяет бизнес-логику на компоненты. По сути, микросервисная архитектура - частный случай SOA c набором более строгих правил.​
* У микросервисов есть особые свойства, они же преимущества: ​  
  ​
* **Гетерогенность**: возможность построить систему с помощью разных языков программирования и технологий;​
* **Децентрализованное управление данными**: каждый микросервис содержит свой набор данных, доступный другим микросервисам только через соответствующий интерфейс;​
* **Независимость инфраструктуры**: каждый микросервис - независимая единица, поэтому вносить изменения и разворачивать его можно независимо от других;​
* **Масштабируемость**: чтобы увеличить производительность системы, нужно расширить только те сервисы, которые в этом нуждаются.​
* ​

Микросервисная архитектура.

* Множество языков, множество возможностей​
* Децентрализованное управление данными​
* Использование открытых стандартов типа HTTP ​
* Автоматизация инфраструктуры​
* Проектирование под отказ (с учетом отказов отдельных микросервисов​
* Синхронные вызовы считаются опасными​
* Каждый сервис работает в отдельном процессе​

Бессерверная архитектура – это способ создания и запуска приложений и сервисов без необходимости управления инфраструктурой. ​ Приложение по-прежнему работает на серверах, но управление этими серверами AWS полностью берет на себя. ​

* Это избавляет от необходимости заниматься выделением ресурсов, масштабированием и обслуживанием серверов для запуска приложений, баз данных и систем хранения данных.​
* для реализации бессерверных архитектур можно использовать облачные сервисы, такие как:​
* AWS Lambda (основан на микросервисах); ​
* Amazon API Gateway;​
* Amazon DynamoDB (NoSQL-СУБД);​
* реляционная СУБД Aurora.​
* Другие облака:​
* Microsoft Azure Functions​
* Google Functions​
* IBM OpenWhisk​

1. Общая архитектура облачных систем. Проблемы архитектурного дизайна систем облачных вычислений.

Включает следующие компоненты:

* 1. ЦОДы
  2. Платформы снабжения виртуализированными вычислительными, сетевыми ресурсами и ресурсами СХД и СУБД из нескольких ЦОДов для обеспечения запросов от мультиарендных приложений;
  3. Управление снабжением ресурсами;
  4. Управление доступом к облаку;
  5. Каталоги предоставляемых сервисов;
  6. Мониторинг производительности и безопасности облака.

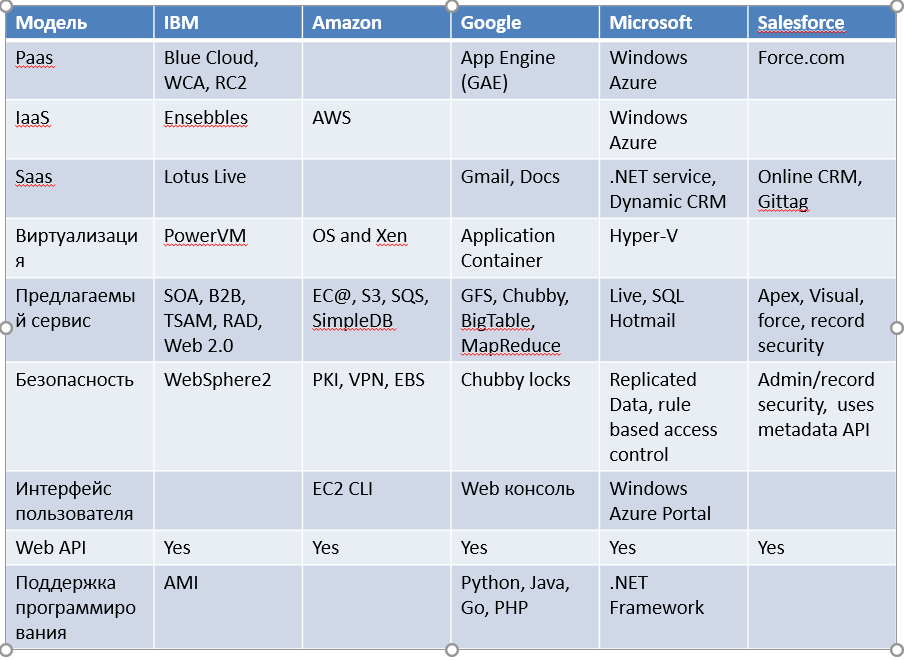
Проблемы архитектурного дизайна систем облачных вычислений.

* Проблема доступности и блокировки данных.
* Конфиденциальность данных и их защита.
* Непредсказуемая производительность
* Распределенные хранилища данных и широко распространенные ошибки программного обеспечения.
* Масштабируемость, Интероперабельность и стандартизация.
* Лицензирование ПО и разделение репутации.

1. Какие приложения не следует располагать в системах облачных вычислений.
2. Обзор основных публичных облачных платформ и предлагаемых ими сервисов.

Пользователями услуг, предоставляемых публичными облаками являются:

* 1. Администраторы ИТ;
  2. Производители ПО;
  3. Конечные пользователи:
     1. Индивидуальные пользователи;
     2. Пользователи организаций (корпоративных ИС).



1. Облачная платформа Goggle – GAE. Инфраструктура и функциональная модель. Основные сервисы. Достоинства и недостатки GAE.

Облачная платформа Google была создана по результатам экспертного анализа запросов к поисковой системе (2008).

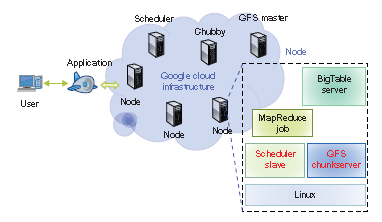
Google имеет несколько сотен ЦОД, развернутых по всему миру, в которых установлено более 460000 серверов (2012).

Основной вид сервиса – PaaS, хотя имеются и облачные приложения-сервисы:

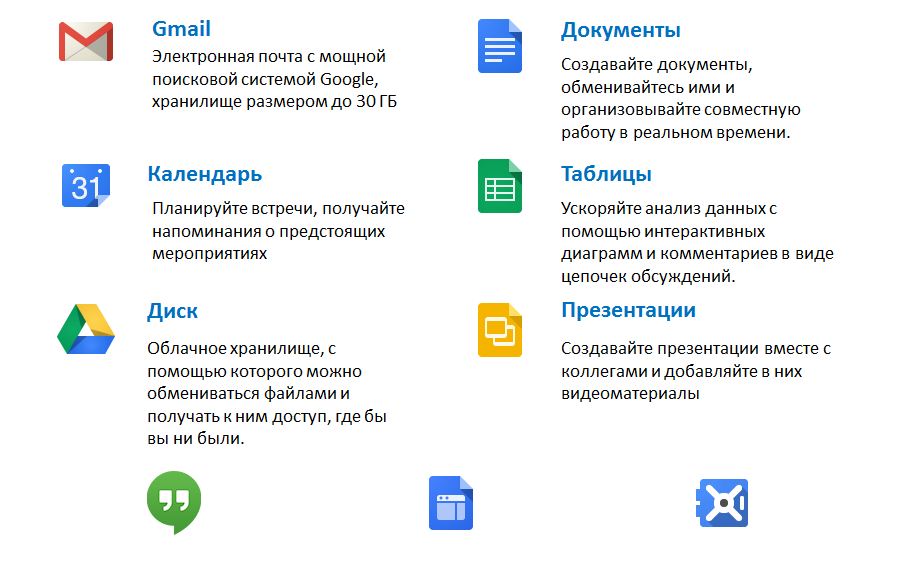
* 1. GMAIL, GoogleDocs, Google Earth;
  2. Служба хостинга приложений;
  3. Служба аккаунтов.

Инфраструктура GAE включает в себя крмпоненты:

* 1. GFS – Google File System;
  2. MapReduce – модель и сервис распределенных вычислений
  3. BigTable - проприетарная высокопроизводительная база данных, построенная на основе Google File System (GFS);
  4. Chubby - Бесплатный онлайн словарь;
  5. Sheduler – планировщик на основе cron



* Компоненты GAE обеспечивают следующую функциональность:
  + Datastore - Хранилище данных. Объектно-ориентированная, распределенная служба хранения структурированных данных на основе технологий BigTable.
  + Среда исполнения приложений - образует платформу для разработки и исполнения масшабируемых веб приложений. Языки: Java, Python.
  + SDK – инструменты разработки приложений. Позволяет тестировать приложения локально и загружать их в облако GAE4;
  + Административная консоль – используется для управления приложениями пользователей в цикле разработки, вместо использования физических ресурсов облака.
  + GAE web service infrastructure – предоставляет специальный интерфейс , обеспечивающий гибкость использования и управление ресурсами системы хранения и сетевыми ресурсами.

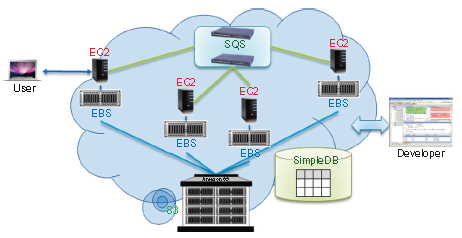


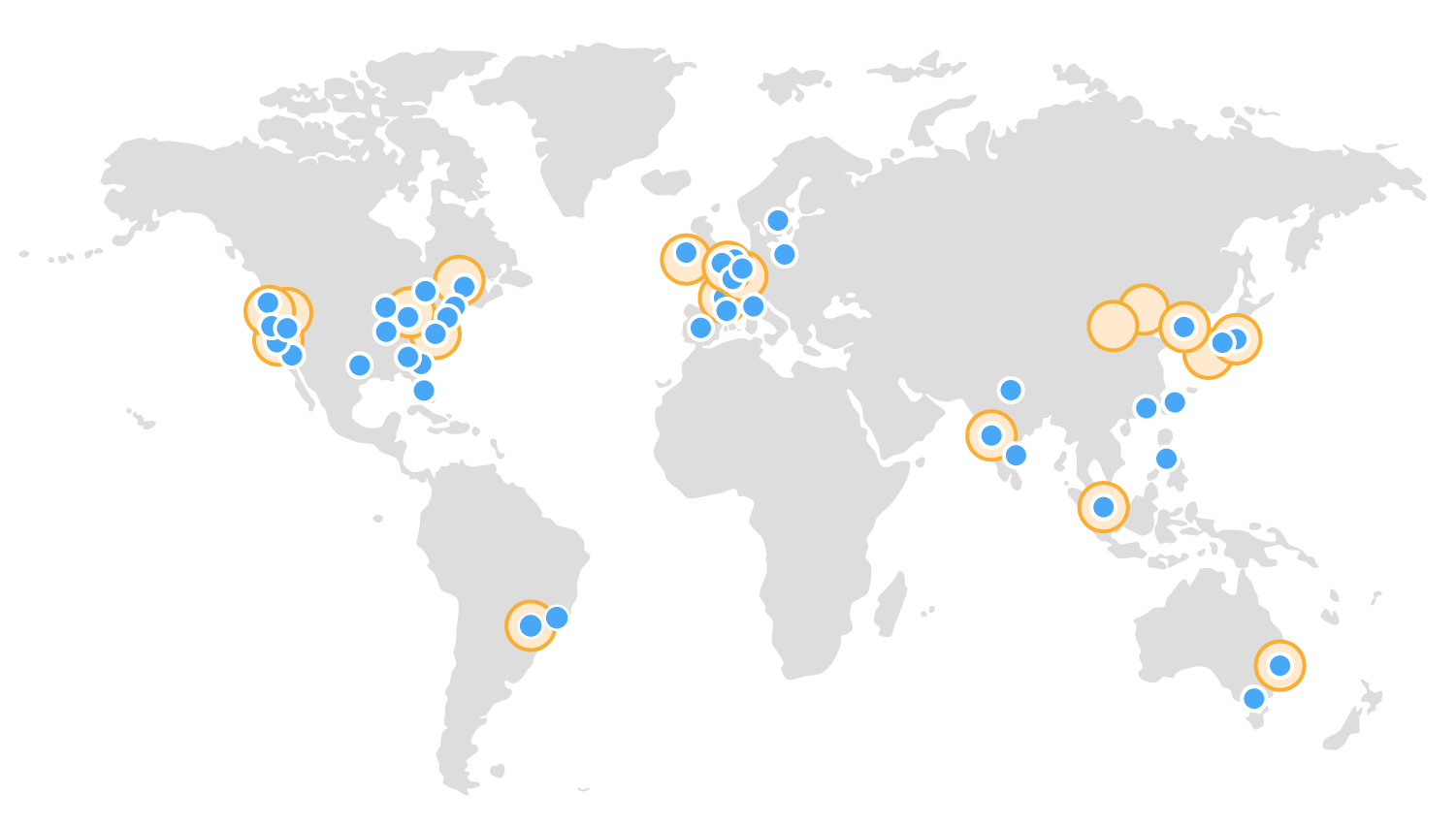
* ***Достоинства***
  + Для GAE быстро и легко разрабатывать
  + Не требует усилий по обслуживанию
  + Очень щедро выделены ежедневные бесплатные ресурсы
  + Вполне разумные цены при превышении квот
  + Лёгкое и быстрое выкладывание
* ***Недостатки***
  + Некоторые приложения невозможно реализовать из-за ограниченности функциональных возможностей
  + Контролируемое окружение не позволяет использовать некоторые стандартные библиотеки/фреймворки
  + Не поддерживаются реляционные базы данных
  + Может потребоваться дополнительная работа при миграции с GAE (если понадобится)

1. Облачная платформа Amazon Web Services - AWS. Архитектура, зоны доступности. Бесплатные сервисы AWS. Способы управления сервисами. Инструменты AWS для разработчиков.

(AWS) — инфраструктура платформ облачных веб-сервисов, представленная компанией Amazon в начале 2006 года;

В 2015 г. AWS имеет более миллиона активных клиентов каждый месяц из 190 стран, в том числе около 2000 правительственных учреждений, 5000 учебных заведений и более 17500 некоммерческих организаций.





В инфраструктуре AWS представлено много сервисов для предоставления различных услуг, таких как:

* 1. хранение данных (файловый хостинг, распределённые хранилища данных),
  2. аренда виртуальных серверов,
  3. предоставление вычислительных мощностей и др.



Бесплатные сервисы

* Виртуальные машины только на базе Amazon EC2 Linux Micro Instance. Бесплатно предоставляется 750 часов использования, которых, в течении первого года с момента регистрации учетной записи, будет достаточно для использования одного EC2 экземпляра в режиме 24/7;​
* 750 часов использования сервиса [Elastic Load Balancer](http://aws.amazon.com/elasticloadbalancing/) + 15 гигабайт обработанных данных;​
* 10 гигабайт в сервисе [Amazon Elastic Block Storage](http://aws.amazon.com/ebs) (EBS). В бесплатный лимит EBS включены 1 миллион операций ввода/ввывода, 1 гигабайт для хранения мгновенных снимков, 10000 get запросов и 1000 put запросов к этим мгновенным снимкам;​
* 5 гигабайт в системе хранения [Amazon Simple Storage Service](http://aws.amazon.com/s3) (S3). В бесплатный лимит S3 включены 20000 Get запросов и 2000 Put запросов;​
* 30 гигабайт входящего и исходящего трафика в равных пропорциях;​
* 25 часов работы с [Amazon SimpleDB](http://aws.amazon.com/simpledb) и 1 гигабайт для данных;​
* 100000 запросов впри использовании сервиса [Amazon Simple Queue](http://aws.amazon.com/sqs);​
* 100000 запросов, 100000 HTTP уведомлений и 1000 уведомлений по электронной почте при использовании сервиса [Amazon Simple Notification](http://aws.amazon.com/sns);​
* 10 метрик, 10 предупреждений и 1000000 запросов к API в сервисе [Amazon Cloudwatch](http://aws.amazon.com/cloudwatch).​

Инструменты для разработчиков

* [AWS CodeStar Разрабатывайте и развертывайте приложения на AWS](https://aws.amazon.com/ru/codestar/?p=tile)​
* [AWS CodeCommit Хранение кода в частных репозиториях Git](https://aws.amazon.com/ru/codecommit/?p=tile)​
* [AWS CodeBuild Сборка и тестирование кода](https://aws.amazon.com/ru/codebuild/?p=tile)​
* [AWS CodeDeploy Автоматизированное развертывание кода](https://aws.amazon.com/ru/codedeploy/?p=tile)​
* [AWS CodePipeline Выпуск ПО с использованием непрерывной доставки](https://aws.amazon.com/ru/codepipeline/?hp=tile&so-exp=below)​
* [AWS Cloud9 Создание, запуск и отладка кода в облачной IDE](https://aws.amazon.com/ru/cloud9/?p=tile)​
* [AWS X-Ray Анализ и отладка приложений](https://aws.amazon.com/ru/xray/?p=tile)​
* [Интерфейс командной строки AWS Унифицированное средство управления сервисами AWS](https://aws.amazon.com/ru/cli/?p=tile)

1. Сервис Amazon EC2. Функциональность, типы инстансов, запуск инстансов.

* **Amazon Elastic Compute Cloud (EC2)**— это облачный сервис, предоставляющий:​
* виртуальные сервера​
* 2 вида хранилищ данных:​
* краткосрочное (исчезает вместе с ВМ)​
* блочное (остается при удалении ВМ).​
* балансировщик нагрузки.​
* С помощью EC2 можно:​
* создать Amazon Machine Image (AMI), который будет содержать ваши приложения, библиотеки, данные и связанные с ними конфигурационные параметры. Или использовать заранее настроенные шаблоны образов для работы;​
* загрузить AMI в [Amazon S3](https://ru.wikipedia.org/wiki/Amazon_S3). Amazon EC2 предоставляет инструменты для хранения AMI. [Amazon S3](https://ru.wikipedia.org/wiki/Amazon_S3) обеспечивает безопасное, надёжное и быстрое хранилище для хранения образов; ​
* использовать Amazon EC2 [Веб-сервис](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B8%D1%81) для настройки безопасности и сетевого доступа;​
* выбирать тип(ы) [операционной системы](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0), какой вам необходим, запустить, завершить, или контролировать несколько AMI по мере необходимости, используя [API](https://ru.wikipedia.org/wiki/API) [Веб-сервиса](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B8%D1%81), или различных инструментов управления, которые предусмотрены;​
* Балансировка нагрузки и автомасштабирование. Можно создать правила при которых станет возможно автоматически увеличить количество серверов, если один или несколько серверов несправляются с нагрузкой.​
* определить необходимость работать в нескольких местах, использовать статический [IP](https://ru.wikipedia.org/wiki/IP-%D0%B0%D0%B4%D1%80%D0%B5%D1%81) или другие варианты; ​
* платить только за ресурсы, которые вы собираетесь потреблять, такие как время или передача данных.​

Запуск инстанса.

* **ерез AWS Management Console**: ​

- Зайти в консоль​

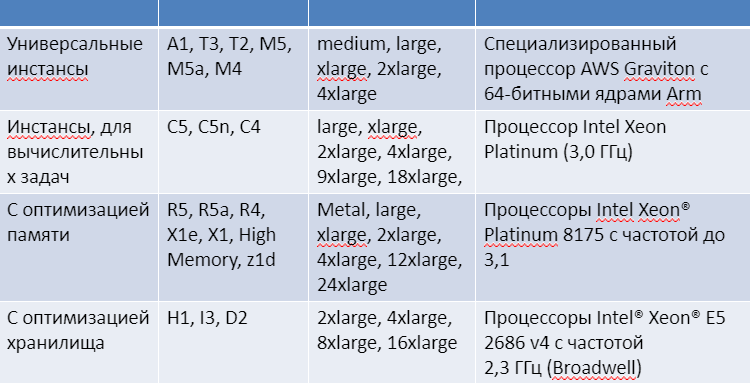
<https://console.aws.amazon.com/ec2/>​

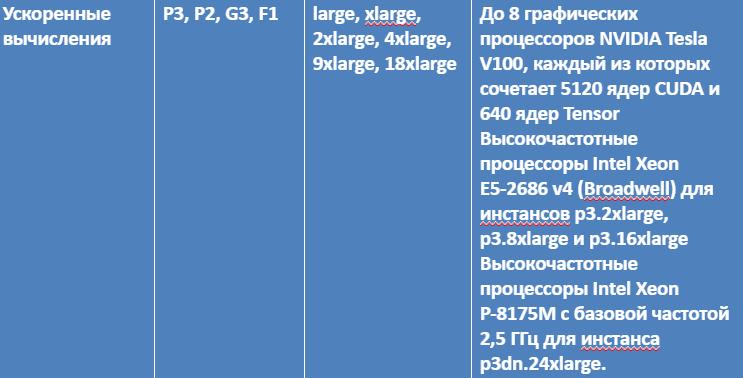
- В панели консоли выбрать **Launch Instance**​

• **Через командную строку**:​

**$ ec2-run-instances ami\_id --availability-zone zone**​

Типы инстансов





1. Сервис Amazon S3. Функциональность, виды услуг, пространство имен корзин S3. Поддержка версион ности.

**Amazon Simple Storage Service** (Amazon S3)

* онлайновая [веб-служба](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%BB%D1%83%D0%B6%D0%B1%D0%B0), предлагаемая [Amazon Web Services](https://ru.wikipedia.org/wiki/Amazon_Web_Services), предоставляющая возможность для хранения и получения любого объёма данных, в любое время из любой точки сети, так называемый файловый [хостинг](https://ru.wikipedia.org/wiki/%D0%A5%D0%BE%D1%81%D1%82%D0%B8%D0%BD%D0%B3). ​
* С помощью Amazon S3 достигается высокая масштабируемость, надёжность, высокая скорость и недорогая инфраструктура хранения данных. ​ Файлы хранятся в отдельных бакетах, в которых можно создавать директории и поддиректории. Бакеты хранятся в разных регионах.
* Впервые появилась в марте 2006 года в США[[1]](https://ru.wikipedia.org/wiki/Amazon_S3#cite_note-1) и в ноябре 2007 года в Европе.[[2]](https://ru.wikipedia.org/wiki/Amazon_S3#cite_note-2) ​
* Amazon S3 используется многими другими сервисами для хранения и хостинга файлов. Например, сервисы хранения и обмена файлов [Dropbox](https://ru.wikipedia.org/wiki/Dropbox) и [Ubuntu One](https://ru.wikipedia.org/wiki/Ubuntu_One), веб-сайты [Twitter](https://ru.wikipedia.org/wiki/Twitter) и [Woot.com](https://ru.wikipedia.org/w/index.php?title=Woot.com&action=edit&redlink=1), загрузчик игры [Minecraft](https://ru.wikipedia.org/wiki/Minecraft). ​

**К бакетам можно применять разного рода политики безопасности**: делать их приватными, публичными, а так же разделять права между пользователями.​

• **S3 может логгировать запросы и складывать отчёты в отдельный бакет**. Это удобно при расследовании, когда много пользователей или приложений имеют доступ к сервису.​

• **Загрузка, удаление и другие операции доступны по REST или SOAP**, так же возможно шифрование канала передачи данных с S3.​

* **Можно встроить BitTorrent протокол c**заменой http, как основного протокола скачивания файлов.​
* **Предоставляется 99.999999999% гарантия целостности**и 99.99% гарантия доступности файлов в год.​
* **S3 так же предполагает версионность файлов**.​

Всегда можно восстановить файл предыдущей версии, т.е. откатиться до нужного состояния.​

* **Пространство имен бакетов одно на всех пользователей**, поэтому названия бакетов должны быть уникальными.​

**Использование  Amazon  S3**​

Можно при помощи консоли, а можно при помощи инструментов командной строки.

* Создание корзины (bucket), где будут​

храниться файлы:​

s3cmd  mb  s3://*BUCKET*​

* Заполнение корзины файлами:​

s3cmd  put  *LOCAL\_FILE*s3://*BUCKET/S3FILE*​

* Извлечение файла из облака:​

s3cmd  get s3://*BUCKET/S3FILE  LOCAL\_FILE*​

​

1. Сервис Amazon Route 53. Назначение и область применения.

**Route53**— это облачный DNS сервис от​ Amazon.​

• Помогает улучшить производительность​ приложений в глобальном масштабе.​

• Amazon постоянно собирают анонимную информацию о задержках до конечных пользователей и проводят сравнительные анализы этих самых задержек.​

• Amazon Route 53 автоматически определит ближайший к клиенту датацентр и отдаст IP вашего приложения в нем.​

Отличительная особенность интеграция с другими сервисами AWS.

[DNS](https://aws.amazon.com/route53/what-is-dns/) – это глобальный распределенный сервис, преобразующий доменные имена, удобные для человеческого восприятия (например, www.example.com), в числовые IP-адреса (например, 192.0.2.1), используемые для взаимодействия компьютеров между собой. Система DNS в Интернете очень похожа на телефонную книгу, которая устанавливает привязку имен абонентов к их телефонным номерам.

С помощью Amazon Route 53 можно создавать публичные записи DNS и управлять ими. Он позволяет управлять IP-адресами, которые соответствуют вашим доменным именам в «телефонной книге» сервиса DNS Интернета. Сервис Route 53 по запросу также преобразует конкретные доменные имена в соответствующие IP-адреса (например, 192.0.2.1). Route 53 можно использовать для создания записей DNS для нового домена или для передачи записей DNS существующего домена. Простые API сервиса Route 53, соответствующие стандарту REST, позволяют легко создавать и обновлять записи DNS и управлять ими. С помощью функции проверки работоспособности Route 53 можно контролировать состояние и производительность приложений, веб-серверов и других ресурсов. Кроме того, вы можете зарегистрировать новые доменные имена или передать существующие доменные имена под управление Route 53.

Управление осуществляется через консоль или через инструменты командной строки.

1. Сервисы AWS уровня построения платформы.

Содержит Web-сервисы для работы с:​

• **Параллельной обработкой**– Amazon Elastic MapReduce.​

• **Платежными системами**– Amazon DevPay, Amazon FPS.​

• **Доставкой контента**– Amazon CloudFront.​

• **Workforce**– Amazon Mechanical Turk.​

• **Обработкой сообщений**– Amazon SNS, Amazon SQS​

• **Электронной почтой**– Amazon SES.​

#### Simple Queue Service (SQS)

**SQS** — сервис для построения очередей событий. Требуется такая очередь, например, когда разделены приложения создания имейла и его посылка. Тогда создаётся элемент очереди c телом письма, хедерами и т.п, а приложение отправляющее почту считывает элементы из очереди и рассылает их.  
  
Мы используем очереди SQS для создания и отправки Push сообщений. А так же для отправки электронной почты.  
  
Лимитов по количеству очередей и по количеству элементов в очередях Amazon не предоставляет.

#### Simple Email Service (SES)

##### Описание SES используется для отправки почты, а точнее рассылок. Высокая репутация IP адресов, высокая производительность серверов, позволяющая слать десятки-сотни тысяч писем в день даёт возможность осуществлять рассылку сообщений от малого до огромного корпоративного размера предприятия.  Особенностью можно считать автоматическое увеличение лимита писем посланных в сутки. С 10 тысяч до миллиона лимит поднимается атоматически в зависимости от ваших нужд посылки. Так же увеличивается лимит количества писем, посланных в секунду. В начале “прокачки” аккаунта этот лимит стоит на 5 штуках в секунду.

##### Функционал

SES позволяет слать письма через API — непосредственно из приложения. Существуют десятки библиотек, плагинов дающих возможность слать письма обходя SMTP методы. Для тех приложений, которые не могут быть интегрированы с SES через API — существует опция включения SMTP сервера с авторизацией по связке логин-пароль.

1. Сервисы уровня кросс-сервисных услуг и интеграции.

Содержит Web-сервисы для работы с:​

 • **Аутентификацией и авторизацией**– AWS IAM, MFA.​

 • **Мониторингом**– Amazon CloudWatch.​

 • **Развертыванием и автоматизацией**– AWS​

Elastic Beanstalk, AWS Cloud Formation.​

* Сервис IAM позволяет контролировать права доступа ко всем остальным сервисам AWS.​
* Каждому IAM пользователю можно присвоить:​
* пару ключей​
* логин и пароль​
* пару сертификатов​
* С ключами и сертификатами пользователи могут иметь доступ к API и утилитам командной строки.​
* С логином и паролем — в консоль, которая доступна лишь членам организации​

AWS Elastic Beanstalk​

* Это простой в использовании сервис для развертывания и масштабирования интернет-приложений и сервисов, разработанных с помощью Java, .NET, PHP, Node.js, Python, Ruby, Go и Docker, на серверах Apache, Nginx, Passenger и IIS.​
* Просто загрузите код, а Elastic Beanstalk автоматически выполнит развертывание: выделит ресурсы, займется балансировкой нагрузки, автоматическим масштабированием и мониторингом работоспособности приложения. ​
* При этом пользователь сохраняет полный контроль над ресурсами AWS, используемыми для приложения, и в любое время может получить к ним доступ.​

AWS Cloud Formation

* AWS CloudFormation предоставляет универсальный язык для описания и выделения всех ресурсов инфраструктуры в облачной среде. ​
* CloudFormation позволяет использовать простой текстовый файл для автоматического безопасного моделирования и выделения всех ресурсов, необходимых для приложений по всем регионам и аккаунтам пользователя. ​
* Этот файл служит единым источником достоверной информации для создаваемой облачной среды.​

Amazon Cloud Watch​

* **loud Watch**используется  для  мониторинга здоровья и состояния всех сервисов AWS, включая стандартный мониторинг здоровья серверов, доступность тех или иных портов, систем хранения, работу СУБД, место на S3 и очень много всяких других проверок.​
* Автомасштабирование построено на показателях счётчиков CloudWatch.​
* По политикам CloudWatch могут сработать триггеры, которые запускают новые копии серверов для увеличения мощности приложения, и так же при снижении нагрузки потушить ненужные серверы​
* Cloud Watch существует 3 типа состояний:​
* •  **OK**(нормальное состояние)​
* • **ALARM**(ошибка или тревога)​
* • **UNSUFFICIENT  DATA**(неизвестное состояние)​
* На все состояния  можно настроить  триггеры, которые будут срабатывать  во  время изменения состояния.

1. AWS. Сервис Хранилище, состав основных услуг.

* [Amazon S3Масштабируемое хранилище данных в облаке](https://aws.amazon.com/ru/s3/?p=tile)​
* [Amazon EBSБлочное хранилище для EC2](https://aws.amazon.com/ru/ebs/?p=tile)​
* [Amazon Elastic File SystemУправляемое файловое хранилище для EC2](https://aws.amazon.com/ru/efs/?p=tile)​
* [Amazon GlacierНедорогое архивное хранилище в облаке](https://aws.amazon.com/glacier1/?p=tile)​
* [AWS Storage GatewayИнтеграция гибридных хранилищ](https://aws.amazon.com/ru/storagegateway/?p=tile)​
* [AWS SnowballПередача петабайтов данных](https://aws.amazon.com/ru/snowball/?p=tile)​
* [AWS Snowball EdgeПередача петабайтов данных с использованием встроенных вычислительных ресурсов](https://aws.amazon.com/ru/snowball-edge/?p=tile)​
* [AWS SnowmobileПередача эксабайтов данных](https://aws.amazon.com/ru/snowmobile/?p=tile)

1. AWS. Сервис Базы данных, состав основных услуг.

[Amazon AuroraУправляемая реляционная база данных с высокой производительностью](https://aws.amazon.com/ru/rds/aurora/?p=tile)​

[Amazon RDSУправляемый сервис реляционных БД для MySQL, PostgreSQL, Oracle, SQL Server и MariaDB](https://aws.amazon.com/ru/rds/?p=tile)​

[Amazon DynamoDBУправляемая база данных NoSQL](https://aws.amazon.com/ru/dynamodb/?hp=tile&so-exp=below)​

[Amazon ElastiCacheСистема кэширования в памяти](https://aws.amazon.com/ru/elasticache/?p=tile)​

[Amazon RedshiftБыстрое, простое и экономичное хранение данных](https://aws.amazon.com/ru/redshift/?p=tile)​

[Amazon NeptuneПолностью управляемый сервис графической базы данных](https://aws.amazon.com/ru/neptune/?p=tile)​

[AWS Database Migration ServiceПроведите миграцию баз данных с минимальным временем простоя](https://aws.amazon.com/ru/dms/?p=tile)​

1. AWS. Сервис Сетевая конфигурация и доставка контента, состав основных услуг.

* [Amazon VPC Изолированные облачные ресурсы](https://aws.amazon.com/ru/vpc/?p=tile)​
* [Amazon CloudFront Глобальная сеть доставки контента](https://aws.amazon.com/ru/cloudfront/?p=tile)​
* [Amazon Route 53 Масштабируемая система доменных имен](https://aws.amazon.com/ru/route53/?p=tile)​
* [Amazon API GatewayРазработка, развертывание и управление API](https://aws.amazon.com/ru/api-gateway/?p=tile)​
* [AWS Direct Connect Выделенное сетевое подключение к AWS](https://aws.amazon.com/ru/directconnect/?p=tile)​
* [Elastic Load Balancing Балансировка нагрузки при больших масштабах](https://aws.amazon.com/ru/elasticloadbalancing/?p=tile)

1. Платформа Microsoft Windows Azure. Архитектура, зоны доступности. Бесплатные сервисы Azure. Способы управления сервисами

Windows Azure — это открытая и гибкая облачная платформа, позволяющая быстро создавать, развертывать и управлять приложениями в глобальной сети центров обработки данных под управлением Майкрософт.

Приложения можно разрабатывать с помощью любого языка, инструмента или платформы.

Кроме того, приложения в общедоступном облаке можно интегрировать с имеющейся ИТ-средой предприятия.

* Сейчас службы Windows Azure размещены в нескольких центрах обработки данных в США, Европе и Азии. ​
* Эти центры управляются корпорацией Майкрософт и обеспечивают кругло суточную профессиональную поддержку.
* Платформа Azure располагает большим количеством регионов по всему миру, чем любой другой поставщик облачных служб. Это обеспечивает необходимый масштаб для размещения приложений ближе к пользователям в разных уголках мира и хранения данных в локальных расположениях, а также гарантирует максимальную устойчивость и соответствие требованиям.​
* **54  региона (весь мир)**​
* **доступно в 140 странах**​
* Windows Azure предоставляет компаниям облачные службы четырех основных типов:
* вычислительные службы;
* сетевые службы;
* службы обработки данных;
* службы приложений.
* Для того, чтобы быстро развернуть сложное решение на платформе Azure можно воспользоваться преимуществом Azure Marketplace. Для этого выберите решение в магазине Azure.​
* Переход по ссылке “Создание виртуальной машины” перенесет вас на портал Azure к мастеру создания кластера на базе решения от DataStax.​
* После заполнения необходимых параметров вы получите возможность задать размер кластера с числом узлов.​
* Указав еще несколько параметров вы получите развернутый в Azure кластер объединяющий множество узлов:​
* Таким образом, Azure Marketplace позволил в несколько кликов и за несколько минут развернуть сложное инфраструктурное решение состоящие из десятков элементов.

1. Обзор служб Windows Azure.

* Windows Azure предоставляет компаниям облачные службы четырех основных типов:
* вычислительные службы;
* сетевые службы;
* службы обработки данных;
* службы приложений.

1. Вычислительные службы Windows Azure.

Актуальная версия Windows Azure поддерживает четыре вычислительные службы:

* 1. **Виртуальные машины**. Предоставляет универсальную вычислительную среду, в которой можно создавать, развертывать и управлять виртуальными машинами в облаке Windows Azure.
  2. **Веб-сайты**. Предоставляет управляемую облачную веб-среду, в которой можно как создавать новые веб-сайты, так и переносить в нее существующие.
  3. **Облачные службы**. Позволяет создавать и развертывать почти неограниченно масштабируемые приложения высокой доступности практически на любом языке программирования и с минимальными расходами на администрирование.
  4. **Мобильные службы**. Является готовым к использованию решением для создания и развертывания приложений, а также для хранения данных для мобильных устройств.

1. Сетевые службы и Web приложения Windows Azure.

Windows Azure поддерживает две сетевые службы

**Виртуальная сеть**. Позволяет использовать общедоступное облако Windows Azure в качестве расширения локального центра обработки данных.

**Диспетчер трафика.** Позволяет маршрутизировать трафик приложений для пользователей центров обработки данных Windows Azure тремя способами:

* 1. с максимальной производительностью,
  2. по принципу циклического обслуживания и
  3. на основе отказоустойчивой активно-пассивной конфигурации.

Azure Marketplace предлагает [десятки готовых шаблонов популярных CMS](http://azure.microsoft.com/ru-ru/marketplace/web-applications/) и веб-приложений, таких как Drupal, Wordpress, Django, CakePHP, Flask, Joomla, Express, проектов ASP.NET и многих других.

Отличительная черта шаблонов веб-приложений в том, что они предназначены для развертывания в среде Azure Web App – в виде сервисов, а не виртуальных машин, благодаря чему упрощается и ускоряется их развертывания, обслуживание и снижается цена.

1. Службы обработки данных Windows Azure

Windows Azure поддерживает пять служб обработки данных.

* 1. **Управление данными.** Позволяет хранить бизнес-информацию в базах данных SQL различными способами — на выделенных виртуальных машинах Microsoft SQL Server, в базе данных Windows Azure SQL, в таблицах NoSQL через REST или в BLOB-хранилище.
  2. **Бизнес-аналитика.** Облегчает изучение и повышает информативность данных с помощью служб Microsoft SQL Server Reporting and Analysis или Microsoft SharePoint Server, работающих на виртуальной машине, Windows Azure SQL Reporting, Windows Azure Marketplace или HDInsight — реализации «больших данных» на базе Hadoop.
  3. **HDInsight.** Разработанная Майкрософт на основе Hadoop, эта служба полностью повторяет функциональность Apache Hadoop в облаке.
  4. **Кэш.** Включает в себя распределенное решение для кэширования, ускоряющее работу облачных приложений и снижающее нагрузку на базу данных.
  5. **Резервное копирование.** Помогает автономно защитить данные на сервере, позволяя создавать как автоматические, так и ручные копии в Windows Azure.

1. Службы приложений Windows Azure.

Windows Azure поддерживает семь служб обработки данных.

**Мультимедийные службы.** Позволяет формировать рабочие процессы для создания, управления и распространения медиаконтента с помощью общедоступного облака Windows Azure.

**Обмен сообщениями.** Сюда входят две службы (шина обслуживания Windows Azure и очередь Windows Azure), обеспечивающие связь приложений в вашем частном облаке и общедоступном облаке Windows Azure.

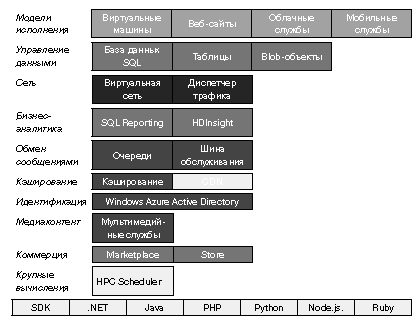
**Узлы уведомлений**. Предоставляет хорошо масштабируемую **кросс**платформенную инфраструктуру push-уведомлений для приложений, работающих на мобильных устройствах.

**Службы BizTalk.** Предоставляет функции B2B (Business-to-Business) и EAI (Enterprise Application Integration) для облачных и гибридных решений по интеграции.

**Active Directory.** Отвечает за управление идентификацией и контроль доступа для облачных приложений.

**Многофакторная аутентификация.** Представляет собой дополнительный уровень аутентификации наряду с учетными данными пользователей, повышая защищенность доступа к локальным и облачным приложениям.

1. Компоненты Windows Azure.



1. Windows Azure решение инфраструктура.

Позволяет быстро развертывать виртуальные машины.

Предоставляется галерея образов предварительно настроенных ВМ, что позволяет упростить процесс развертывания ОС и ПО на ВМ.

Имеется возможность загружать или скачивать диски ВМ.

«Облачные ВМ» можно интегрировать с локальным окружением на основе виртуальных сетей.

1. Windows Azure решение управление доступом.

* Windows Azure AD предоставляет облачные службы идентификации, с помощью которых можно:​
* управлять правами доступа сотрудников, партнеров и клиентов к корпоративным ресурсам — как локальным, так и облачным.​
* синхронизировать локальную инфраструктуру Active Directory с Windows Azure AD, чтобы реализовать для пользователей единый вход (single sign-on, SSO) в облачные приложения​
* использовать многофакторную аутентификацию Windows Azure в качестве дополнительного уровня защиты для особо важных бизнес-данных и приложений.​

1. Windows Azure решения интернет, интеграция и «Большие данные»

Это решения для хостинга веб-сайта предприятия, на основе:

* 1. SNI (Sever Name Indication) и SSL-сертификатов на базе IP;
  2. наличию центров обработки данных по всему миру с гарантированным уровнем услуг (SLA);
  3. круглосуточной поддержкой Windows Azure;
  4. аутентификацию и контроль доступа в Windows Azure Active Directory
  5. хранить бизнес-информацию своего веб-сайта в базе
  6. данных Windows Azure SQL, таблицах NoSQL или BLOB-хранилище
  7. Разрабатывать веб-сайт можно на любом языке программирования, включая ASP.NET, PHP, Node.js, Python и даже Classic ASP;
  8. можно сделать сайт на базе популярной платформы или одного из готовых шаблонов в галерее приложений Windows Azure, в том числе WordPress, Umbraco, DotNetNuke, Drupal, Django, CakePHP и Express.
* Windows Azure позволяет быстро создавать Hadoop-кластеры на базе полнофункционального Apache Hadoop и затем:
  + с помощью Windows Azure PowerShell и интерфейса командной строки Windows Azure можно без труда интегрировать HDInsight в существующие рабочие процессы аналитики, чтобы получать действенные прогнозы от HDInsight путем интеллектуального анализа данных в Microsoft Excel.
* Windows Azure поддерживает несколько вариантов интеграции локальной инфраструктуры с приложениями, работающими в общедоступном облаке Windows Azure:
  + на основе использования шины обслуживания Windows Azure;
  + за счет использования Службы Windows Azure BizTalk для встраивания приложений business-to-business (B2B) в PaaS;
  + собственных решений для интеграции с помощью инструментов .NET и Visual Studio.

1. Windows Azure сервисы разработки и тестирования.

Windows Azure ускоряет разработку приложений за счет:

* 1. можно просто создать в среде Windows Azure столько виртуальных машин, сколько вам сейчас необходимо
  2. разработку и тестирование выполнить прямо в облаке
  3. когда приложение будет отлажено, можно
  4. развернуть его в рабочей среде (облаке)

1. Windows Azure сервисы PaaS. AppFabric – назначение и ее применение в качестве сервисной шины.

Windows Azure предоставляет облачные службы по созданию приложений (Windows Services as Platform API). Этот сервис использует:​

* AppFabric;​
* SQL Azure;​
* SharePoint Services;​
* Dynamic CRM Services;​
* Live Services;​
* BizTalk Services.

Представляет собой средство обработки запросов пользователей к веб-сервисам Azure, основанное на использовании сервисной шины и средства контроля доступа на базе технологии .NET

поддерживает протоколы SOA – SOAP, REST, протоколы семейства WS-.

Действует как сервисная шина на основе SOA.

AppFabric позволяет работать через меэсетавые экраны и NAT на основе использования адресов рандеву (встречи), которые включают в себя:

- URI приложения;

- пространство имен сервисной шины.

Если оба приложения соответствуют требованиям .NET сервисов, то между ними может быть установлено прямое соединение минуя сервисную шину.

Шаг 1. Клиент запрашивает аутентификацию от службы контроля доступа.

Шаг 2. Контроль доступа создает токен на основе хранимых правил для серверного приложения.

Шаг 3. Токен подписывается и возвращается клиенту.

Шаг 4. Клиент представляет токен службе приложений.

Шаг 5. Серверное приложение проверяет ситгатуру и использует токен для принятия решения какие приложения пользователю можно делать.

1. Windows Azure MarketPlace – назначение и реализация.

Представляет собой канал продаж.

Канал Azure Markertplace с присутствием на более чем сотни рынков по всему миру становится отличным способом для любого разработчика ПО выйти на мировую сцену и начать продавать себя в любой точке мира без лишних хлопот с юридическими и налоговыми вопросами.

Представляет собой каталог решений, сервисов, образов окружений разнообразной сложности, готовых для быстрого автоматического развертывания на платформе Azure.

Основная идея Azure Marketplace – дать независимому разработчику канал продаж своих решений по всему миру, а клиентам Azure – легкий способ развернуть функциональные решения в собственной инфраструктуре.

1. Сервисы SaaS Windows Azure.

Windows Live Services. Office365, Microsoft 365,

M-files – система управления контентом предприятия;

Umbraco – популярная CMS UaaS (Umbraco as a Service);

Snelstat – популярное ПО для финансового бизнеса.

Gep – тоже деловое ПО;

Uber – По для связывания паасжира с водителем такси.

1. Облачные хранилища данных назначение, основные характеристики, преимущества и недостатки.

Облачное хранилище данных (англ. cloud storage) — модель онлайн-хранилища, в котором данные хранятся на многочисленных распределённых в сети серверах, предоставляемых в пользование клиентам, в основном, третьей стороной.

В отличие от модели хранения данных на собственных выделенных серверах, приобретаемых или арендуемых специально для подобных целей, количество или какая-либо внутренняя структура серверов клиенту, в общем случае, не видна.

* Данные хранятся и обрабатываются в так называемом «облаке», которое представляет собой, с точки зрения клиента, один большой виртуальный сервер.

Физически же такие серверы могут располагаться удалённо друг от друга географически

Преимущества облачных хранилищ данных

1. Возможность бесплатно хранить определённый объём файлов, у каждого провайдера есть свои условия использования и ограниченное свободное место;
2. Оплата услуг идёт только за фактическое использование хранилища, а не за аренду всего сервера;
3. Уменьшение общих издержек на ИТ инфраструктуру организации, т.е. клиентам нет необходимости заниматься приобретением, поддержкой и обслуживанием собственной ресурсов по хранению данных;
4. Неограниченный доступ к данным отовсюду;
5. резервирование и все процедуры, связанные с сохранностью и целостностью данных, производятся самим провайдером облачного центра, клиент в этот процесс никак не вовлекается;
6. гибкость облачных хранилищ, сервис подходит практически, под все операционные системы, как персональные компьютеры, так и мобильные;
7. синхронизация файлов и доступ к облачным файлам, через операционную систему пользователя, зависит от самого клиента облака, поэтому нужно обращать внимание при выборе облака, на его готовность к работе с десктопным клиентом.

Недостатки ОХД

1. Если нет интернета, то нет доступа к данным в облаке.
2. Конфиденциальность информации - не все сервисы шифруют данные в облаке, что создает потенциальную угрозу перехвата информации.

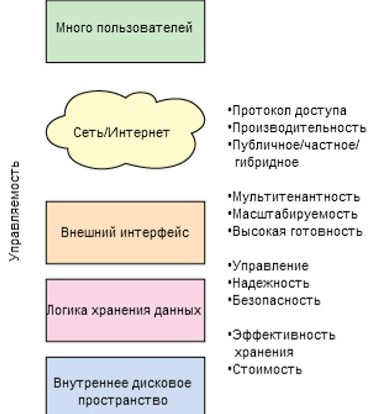
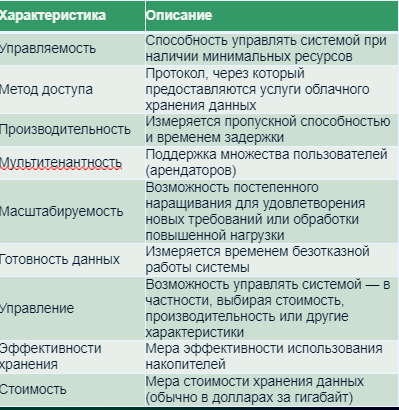
Характеристики общедоступных ОХД

* Стоимость в зависимости от объема;
* Возможность синхронизации данных с другими устройствами хранения пользователя.
* Надежность хранения – обеспечение процедурами сохранения/восстановления;
* Конфиденциальность данных;
* Дополнительные бонусы.

1. Общая архитектура облачных хранилищ данных. Управляемость и методы доступа к ОХД.

Облачная архитектура хранения данных ― это прежде всего доставка ресурсов хранения данных по требованию в высокомасштабируемой и мультитенантной (мультиарендной) среде. ​

Обобщенно облачная архитектура хранения данных представляет собой внешний интерфейс, который предоставляет API для доступа к накопителям​

Управляемость

* Расходы клиента можно разделить на две общие категории: ​
* стоимость самой физической экосистемы хранения ​
* расходы на управление ею. ​
* Стоимость управления скрыта, но представляет собой долгосрочный компонент общей стоимости. ​
* По этой причине облачная система хранения должна быть в значительной степени самоуправляемой. ​
* Решающее значение имеет возможность добавлять новые накопители, когда система автоматически реконфигурируется для их размещения, и способность системы находить и автоматически исправлять ошибки.

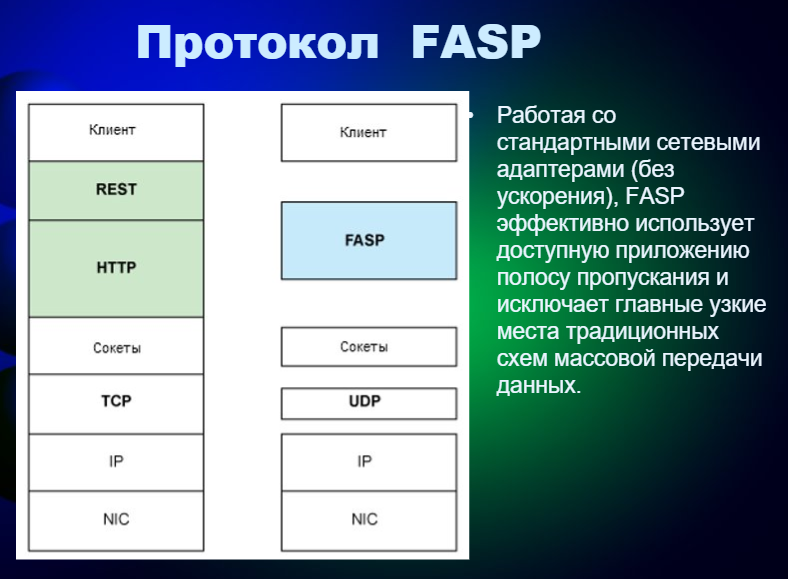
Методы доступа

* Для доступа к ОХД используются следующие методы доступа: ​
* API Web-сервисов в том числе RESTfull-API (Amazon S3), Windows Azure™ , Mezeo Cloud Storage Platform;​
* Использование API Web-сервиса, требует интеграции с приложенем, поэтому используются и общие методы доступа:​
* На основе файлов:​
* NFS;​
* Common Internet File System(CIFS);​
* FTP;​
* На основе блоков:​
* iSCSI\​
* Другие:​
* WebDAV (Web-based Distributed Authoring and Versioning)​

1. Масштабируемость, производительность, мультитенантность, эффективность и высокая готовность ОХД. Протокол FASP. Алгоритм рассредоточения данных IDA.

Производительность

* Главная задача ОСХД ― это перемещение данных между пользователем и удаленным поставщиком облачных услуг. Проблема кроется в TCP, который управляет потоком данных на основе подтверждения приема пакетов из удаленного узла. ​
* Потеря или задержка пакетов приводит к применению мер по ограничению заторов пакетов с дополнительным ограничением производительности во избежание глобальных сетевых проблем. ​
* TCP идеально подходит для перемещения небольших объемов данных через глобальную сеть Интернет, но не для доставки больших объемов данных – в этом случае время обмена данными (RTT) увеличивается. ​
* Amazon с помощью Aspera Software решила эту проблему, исключив из уравнения TCP. Для ускорения массового перемещения данных во избежание больших RTT и крупных потерь пакетов разработан новый протокол *Fast and Secure Protocol* (FASP™).​
* Главная задача облачной системы хранения данных ― это перемещение данных между пользователем и удаленным поставщиком облачных услуг. ​
* Проблема кроется в том, что TCP управляет потоком данных на основе подтверждения приема пакетов из удаленного узла. Потеря или задержка пакетов приводит к применению мер по ограничению скоплений пакетов с дополнительным ограничением производительности во избежание глобальных сетевых проблем. ​
* Amazon с помощью Aspera Software решила эту проблему, исключив TCP при передаче данных. ​
* Был разработан новый протокол *Fast and Secure Protocol* (FASP™) на базе UDP. UDP позволяет управлять заторами протоколу прикладного уровня FASP​



Мультитенантность

* Это означает, что хранилище используется многими пользователями (или «арендаторами» [*англ.* - tenants]). ​
* Мультитенантность затрагивает многие уровни облачной системы хранения, от уровня приложения, где пользователям выделяются изолированные пространства имен, до уровня хранения, где отдельным пользователям или категориям пользователей могут выделяться отдельные физические накопители. ​
* Мультитенантность распространяется даже на сетевую инфраструктуру, которая соединяет пользователей с накопителями, обеспечивая гарантированное качество обслуживания и выделенную полосу пропускания для конкретного пользователя.​

Масштабируемость

* Возможность наращивать ресурсы хранения (как вверх, так и вниз) означает улучшенную экономическую эффективность для пользователя и повышенную сложность для поставщика облачных услуг.​
* Серверы и система хранения должны допускать изменение размера без всяких последствий для пользователей.​

Облачная инфраструктура хранения должна обеспечивать:​

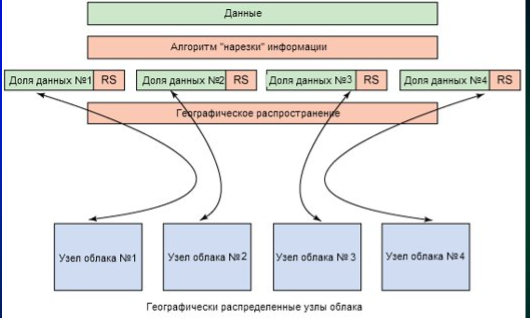
* функциональное масштабирование;​
* масштабирование нагрузки;​
* географическая масштабируемость;​
* и внутреннее масштабирование. Серверы и система хранения должны допускать изменение размера без всяких последствий для пользователей. ​

Высокая готовность

* Когда поставщик облачных услуг хранит данные пользователя, он должен иметь возможность возвратить эти данные пользователю по требованию.​
* Существуют интересные новые схемы обеспечения высокой готовности, такие как рассредоточение информации.​
* Компания Cleversafe, использует алгоритм рассредоточения информации (Information Dispersal Algorithm - IDA) для повышения доступности данных перед лицом физических отказов и простоев сети. ​
* Он позволяет "нарезать" данные с помощью кодов Рида-Соломона для их восстановления в случае потери части данных.​

Алгоритм рассредоточения данных IDA

* IDA позволяет настраивать количество долей данных, так чтобы заданный объект данных можно было разрезать на **четыре** доли при одном допустимом сбое или **на 20 долей**при восьми допустимых сбоях. ​
* Как и RAID, IDA позволяет восстанавливать данные из подмножества исходных данных при некоторых накладных расходах на коды ошибок​

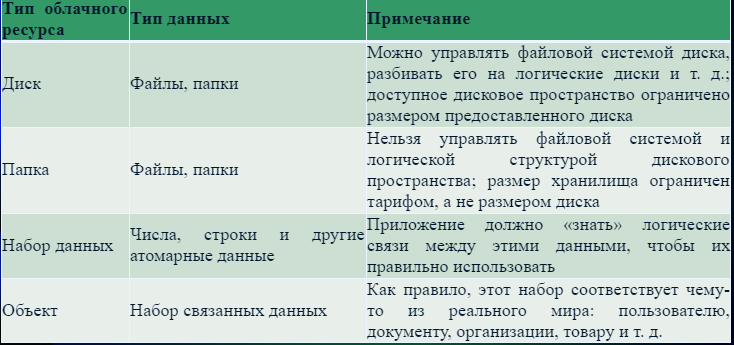


1. Типы облачных хранилищ данных.

Невозможно заранее сказать, какой из названных типов хранилищ данных является наилучшим — его выбор зависит от характера решаемых прикладных задач и применяемого программного обеспечения.

Все типы облачных хранилищ делятся на:​

* 1) блочные (диски);​
* 2) файловые;​
* 3) «базовые» (СУБД); ​
* 4) объектные. ​
* Между собой они различаются степенью агрегации данных — если дисковые и файловые хранилища содержат в себе атомарные данные, то в СУБД и, тем более, в объектных хранилищах данные собраны в связанные структуры, которыми можно оперировать на более высоком, прикладном, уровне.​



Облачные диски

* Сейчас очень часто облачными дисками называют сервисы типа Dropbox, OneDrive или Яндекс.Диск, что неправильно.​
* Для предоставления дискового пространства через интернет существуют специальные протоколы, например, Internet Small Computer Systems Interface (iSCSI), iFCP (Internet Fibre Channel Protocol) или Fibre Channel over IP (FCIP). На их основе создаются, так называемые, сети хранения данных (SAN, Storage Area Network). По этим протоколам дисковые устройства представляются серверу, к которому они подключаются, на самом низком — блочном — уровне, и поэтому являются универсальными.​
* Однако взаимодействие на таком уровне требует интенсивного обмена данными между сервером и дисковым устройством. Если такой обмен идёт по недостаточно быстрым каналам связи, эффективность дискового устройства падает. Если связь высокоскоростная и высоконадёжная, облачные диски могут стать удобной и универсальной частью корпоративной информационной системы.​

Облачные папки

* Именно их сейчас часто именуют облачными дисками. О том, что это неверно, уже было сказано. Тем не менее, облачная папка — полезный и широко востребованный ресурс, особенно среди частных лиц. Например, он даёт возможность использовать одни и те же файлы с разных устройств, подключенных к интернету. Также возможно совместное использование папки для групповой работы с файлами.​
* Положить файл в облачную папку можно либо через браузер и веб-интерфейс, либо через специальную локальную папку, которая автоматически синхронизируется с облачным хранилищем. Во втором случае на компьютер пользователя требуется установить соответствующую утилиту.​
* Помимо доступа к файлам с разных устройств, облачные папки обеспечивают хранение резервных копий этих файлов. Как правило, облачная инфраструктура строится на оборудовании с очень высокой надёжностью.​

СУБД

* Большое распространение получили СУБД на основе SQL-запросов к реляционным базам данных. Помимо них в последнее время стали появляться и находить применение базы неструктурированных или плохо структурированных данных.​
* В ответ на запросы СУБД обоих типов возвращают, так называемые, наборы данных (datasets) — выборки данных, собранных в таблицы или некие списки.​
* Принципиальным является то, что приложение, запросившее данные из хранилища, должно само уметь сложить эти данные в некоторую прикладную сущность, например, «пользователь», «задача», «растение», «транспортное средство», «объект недвижимости» и т. д.​

Объектные хранилища

* Идея объектного хранения данных возникла и прорабатывалась ещё в 70-е и 80-е годы. Но тогда она развивалась в рамках отдельных замкнутых проектов, а сейчас это — широко востребованная услуга, используемая в самых разнообразных приложениях.​
* Для взаимодействия с облачным объектным хранилищем используется программный интерфейс (API). То есть запись объекта в хранилище происходит не перетаскиванием файла из одной папки в другую, а с помощью специальных команд, выдаваемых приложениями.​
* Файлы в таком хранилище сопровождаются дополнительными сведениями — мета-данными — которые позволяют обрабатывать эти файлы как прикладные объекты: бухгалтерские документы, видеофильмы, проекты, товары, фотографии и т. п.​

1. Платформа работы с "большими данными" Apache Hadoop. Назначение, состав модулей. Распределенная файловая система HDFS. Механизм репликации HDFS. Платформа распределенных вычислений MapReduce.

Apache Hadoop

* Является одной из основополагающих технологий «[больших данных](https://ru.wikipedia.org/wiki/%D0%91%D0%BE%D0%BB%D1%8C%D1%88%D0%B8%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5)». ​
* Разработан на [Java](https://ru.wikipedia.org/wiki/Java) в рамках вычислительной парадигмы [MapReduce](https://ru.wikipedia.org/wiki/MapReduce), согласно которой приложение разделяется на большое количество одинаковых элементарных заданий, выполнимых на узлах кластера и естественным образом сводимых в конечный результат.​
* По состоянию на 2014 год, проект состоит из четырёх модулей — ​
* Hadoop Common ([связующее программное обеспечение](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D1%8F%D0%B7%D1%83%D1%8E%D1%89%D0%B5%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) — набор инфраструктурных программных библиотек и утилит, используемых для других модулей и родственных проектов), ​
* HDFS ([распределённая файловая система](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D1%91%D0%BD%D0%BD%D0%B0%D1%8F_%D1%84%D0%B0%D0%B9%D0%BB%D0%BE%D0%B2%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0)), ​
* YARN (система для планирования заданий и управления кластером) ​
* Hadoop MapReduce (платформа программирования и выполнения распределённых MapReduce-вычислений).​
* Интересный факт. *Хадуп — именно так сын одного из изобретателей фреймворка назвал желтого игрушечного слона.*​

Распределенная файловая система HDFS

* В HDFS вместо таблицы используется специальный сервер — сервер имён (NameNode), а данные разбросаны по серверам данных (DataNode). ​
* данные разбиты на блоки (обычно по 64Мб или 128Мб)​
* HDFS имеет классическую Unix-овскую древовидную структуру директорий, пользователей с триплетом прав, и даже схожий набор консольных команд: ​

# оценить размер директории ​

du -sh mydata ​

hadoop fs -du -s -h mydata ​

# вывести на экран содержимое всех файлов в директории ​

cat mydata/\* ​

hadoop fs -cat mydata/\* ​

# просмотреть корневую директорию: локально и на HDFS​

 ls / ​

hadoop fs –ls / ​

* Выполняет 2 главные задачи: ​
* запись метаданных ​
* хранение собственно данных. ​
* Принцип работы HDFS состоит в распределении файлов по нескольким узлам в кластере. ​
* Так, с метаданными работает сервер NameNode, с собственно данными — DataNode.​
* NameNode-сервер управляет пространством имен файловой системы и доступом клиентов к данным.​
* Чтобы разгрузить NameNode-сервер, передача данных осуществляется только между клиентом и DataNode-сервером.​
* HDFS является весьма надежной системой, ведь в случае уничтожения половины узлов кластера, теряется всего 3% информации.​

MapReduce

* MapReduce  обеспечивает параллельные  локальные вычисления на машинах кластера Hadoop с блоками данных, что обеспечивает быструю обработку больших объемов данных. ​
* Принцип работы MapReduce состоит из двух шагов: map и reduce. Вот как происходит процесс:​
* **Map**: получение входных данных главным узлом (master node) -> разделение информации на части -> передача данных рабочим узлам (worker node).​
* **Reduce**: получение главным узлом ответа от рабочих узлов -> формирование результата.​

Механизм репликации HDFS

При обнаружении NameNode-сервером отказа одного из DataNode-серверов (отсутствие heartbeat-сообщений от оного), запускается механизм репликации данных:​  
— выбор новых DataNode-серверов для новых реплик;​  
— балансировка размещения данных по DataNode-серверам.​  
Аналогичные действия производятся в случае повреждении реплик или в случае увеличения количества реплик присущих каждому блоку.​

1. Инструменты для работы с Hadoop. База данных Hbase. Apache Spark.

* Cуществует большое количество проектов непосредственно связанных с Hadoop, но не входящих в Hadoop core:​
* [**Hive**](https://hive.apache.org/) – инструмент для SQL-like запросов над большими данными (превращает SQL-запросы в серию MapReduce–задач);​
* [**Pig**](https://pig.apache.org/) – язык программирования для анализа данных на высоком уровне. Одна строчка кода на этом языке может превратиться в последовательность MapReduce-задач;​
* [**Hbase**](http://hbase.apache.org/) – колоночная база данных, реализующая парадигму [BigTable](https://ru.wikipedia.org/wiki/BigTable);​
* [**Cassandra**](http://cassandra.apache.org/) – высокопроизводительная распределенная key-value база данных; ​
* [**ZooKeeper**](https://zookeeper.apache.org/) – сервис для распределённого хранения конфигурации и синхронизации изменений этой конфигурации; ​
* [**Mahout**](http://mahout.apache.org/) – библиотека и движок машинного обучения на больших данных.​

. База данных Hbase

Hbase — это распределенная, колоночно-ориентированная, мультиверсионная база типа «ключ-значение».​  
​

Данные организованы в **таблицы**, проиндексированные первичным ключом, который в Hbase называется **RowKey**. ​  
Для каждого **RowKey** ключа может храниться неограниченны набор **атрибутов (или колонок)**. ​  
​  
Колонки организованны в **группы колонок**, называемые **Column Family**. Как правило в одну **Column Family** объединяют колонки, для которых одинаковы паттерн использования и хранения. ​  
​  
Для каждого **аттрибута** может храниться несколько различных **версий**. Разные версии имеют разный **timestamp**.​  
​ Поддерживаются 4 основные операции:​  
​  
**— Put**: добавить новую запись в hbase. ​  
**— Get**: получить данные по определенному RowKey. ​  
**— Scan**: читать записи по очереди. Можно указать запись с которой начинаем читать, запись до которой читать, количество записей которые необходимо считать, ​  
**— Delete**: пометить определенную версию к удалению. Физического удаления при этом не произойдет, оно будет отложено до следующего Major Compaction​

Использование Hbase оправдано когда: ​  
​  
— Данных много и они не влезают на один компьютер​  
— Данные часто обновляются и удаляются​  
— В данных присутствует явный «ключ» по к которому удобно привязывать все остальное​  
— Нужна пакетная обработка данных​  
— Нужен произвольный доступ к данным по определенным ключам​

Apache Spark альтернатива хадуп

* [Apache Spark](http://spark.apache.org/) представляет собой движок для распределённой обработки данных. ​
* Apache Spark обычно использует компоненты Hadoop, такие как HDFS и YARN для своей работы, при этом сам в последнее время стал популярнее, чем Hadoop​
* Spark требует наличия менеджера кластера и распределенной системы хранения данных. Если вопрос управления кластерами решается путем использования нативных средств, Hadoop YARN или Apache Mesos (при работе с многоузловыми кластерами), то распределенная система хранения данных может быть исключительно посторонней.​
* Именно по этой причине в большинство проектов с использованием технологий Big Data, Spark устанавливается поверх «слона»: связка из передовых приложений для аналитики от Spark и Hadoop Distributed File System позволяет программам на кластерах Hadoop выполняться до **100 раз** быстрее в RAM и до **10 раз**быстрее на ROM.​
* Spark предлагает ряд операций  по обработке данных. Поэтому при разработке приложений с технологиями Больших Данных, Spark чаще всего заменяет именно Hadoop MapReduce, а не весь Hadoop.

1. Использование технологий "больших данных" в Azure и AWS.

* Windows Azure позволяет быстро создавать Hadoop-кластеры на базе полнофункционального Apache Hadoop и затем:​
* с помощью Windows Azure PowerShell и интерфейса командной строки Windows Azure можно без труда интегрировать HDInsight в существующие рабочие процессы аналитики, чтобы получать действенные прогнозы от HDInsight путем интеллектуального анализа данных в Microsoft Excel.​
* При помощи HDInsight вы можете выполнять интерактивные запросы структурированных и неструктурированных данных в любом формате и объемом в несколько петабайт. ​
* Также можно создавать модели и подключать их к средствам бизнес-аналитики.​
* Azure HDInsight можно применять в различных сценариях обработки больших данных. Это могут быть исторические данные (т. е. собранные и сохраненные) или данные в реальном времени (потоковые данные, передаваемые прямо из источника).​

HDInsight можно использовать для обработки потоковых данных, получаемых в режиме реального времени с различных устройств.​

1. Обзор возможностей, преимущества и недостатки файловых облачных хранилищ OneDrive, Dropbox,

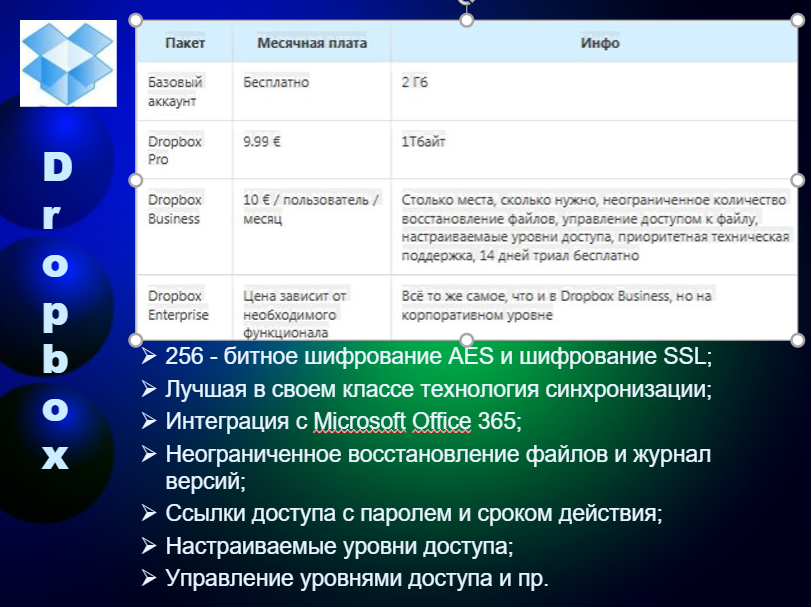


Позитив:

* Освобождение памяти компьютера для ускорения его работы и повышения производительности;
* Возможность не хранить на ПК информацию, являющуюся конфиденциальной;
* Надежное хранение информации, инет вероятности, что она потеряется, например, в результате сбоя в операционной системе или поломки телефона или компьютера;
* Возможность упорядочить материалы, хранить в «облаке» лишь определенный их тип для быстрого доступа;
* Также можно предоставить другим пользователям доступ к определенным материалам или папке по ссылке – это позволяет не пересылать файлы, а просматривать их совместно в режиме онлайн;
* Можно также разрешить редактирование (например, текстового документа) и скачивание файлов нескольким пользователям по ссылке;
* Возможность получить доступ к своим файлам с любого устройства, работающего на основе данной операционной системы, а просмотреть их с любого устройства и вовсе можно по ссылке.
* Для упрощения и систематизации хранения файлов возможно создать несколько аккаунтов с одного или с разных устройств.

Негатив:

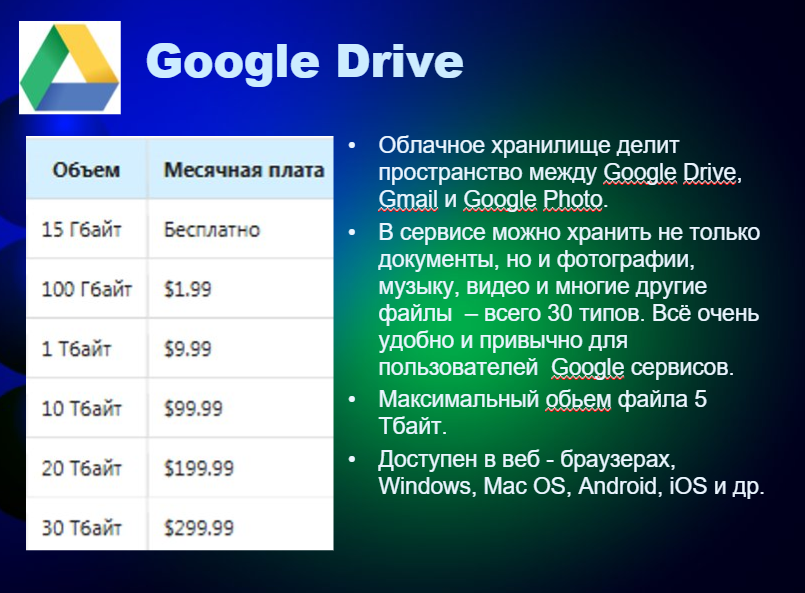
* Отсутствие доступа к сервису из-за технических неполадок на серверах;
* Общий открытый доступ к материалам из-за неправильных настроек доступа, установленных пользователем;
* Взлом пароля и попадание конфиденциальной информации из облака в чужие руки;
* Отсутствие доступа к файлам в случае, когда отсутствует подключение к интернету;
* На диске полностью отсутствует функция поиска, что очень усложняет обращение с программой, когда в нее загружено много документов.
* Кроме того, размер памяти «облака» ограничен в зависимости от типа аккаунта. То есть, со временем место на нем может закончиться, дополнительное же будет предоставляться платно.



Недостатки

* Главным недостатком Dropbox можно считать подход к выбору папок для  синхронизации. Фактически приложение следит за содержимым только одной папки Dropbox.​
* Кроме того, в июле 2014 в своем интервью изданию The Guardian известный Эдвард Сноуден сделал заявление, в котором сообщил, что Dropbox не в полной  мере заботится о конфиденциальности данных пользователя и даже напрямую участвует в глобальной системе слежке PRISM.​
* PRISM —государственная программа США — комплекс мероприятий, осуществляемых с  целью массового негласного сбора информации, передаваемой по сетям электросвязи, ​
* Однако, не все согласны с этим. Использование Dropbox в связке с  BoxCryptor, который надежно шифрует файлы перед синхронизацией их с облаком, обеспечивает  конфидициальность данных в Dropbox.​

1. Обзор возможностей, преимущества и недостатки облачного хранилища Google Drive, Яндекс.Диск, Mega.



Mega



* шифрует весь контент прямо в браузере с помощью алгоритма AES; ​
* пользователи могут передавать друг другу файлы в зашифрованном виде, при этом все данные хранятся в «облаке»; ​
* ключи доступа к файлам не публикуются в открытом доступе, а распространяются по схеме Friend​
* -to-Friend, между доверяющими друг другу пользователями.​
* одним из самых выгодных облачных сервисов;​
* кроме того, важное отличие Mega от других подобных сервисовэто то, что он защищает личные данные пользователя. ​

Яндекс Диск



* Позволяет пользователям хранить свои данные на серверах в блаке и передавать их другим пользователям в интернете. ​
* Работа построена на синхронизации данных между различными устройствами. ​
* В настоящее время регистрация пользователей доступна всем.​
* Яндекс.Диск может выступать в качестве службы облачного сервиса, интегрируясь в офисный пакет Microsoft Office, а недавно появилась возможность  автоматической загрузки фото и видеофайлов с цифровых камер и внешних  носителей информации на Яндекс. Диск. При этом пользователю предоставляются дополнительно 32 ГБ пространства на полгода.​

1. Контейнеры. История возникновение контейнеров. Контейнерная виртуализация в Linux.

Лидером в продвижении контейнерных технологий является компания Parallels (до 2008 г. Swsoft), представив продукт Virtuozzo.

Virtuozzo был создан изначально для Linux, в 2005 г. появилась версия для Windows , примерно тогда же компания Sun реализовала Solaris Containers для своей ОС (и появился термин «контейнеры»).

Название Virtuozzo порой бывает синонимом всего этого направления виртуализации, поскольку остальные средства намного уступают ему в популярности.

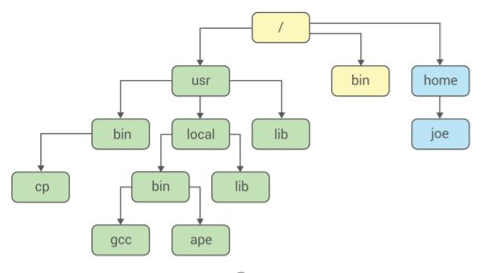
* **Виртуализация на уровне операционной системы — виртуализирует физический сервер на уровне ОС, позволяя запускать изолированные виртуальные серверы называемые Виртуальные Частные Серверы (Virtual Pivate Servers, VPS) или Контейнеры (Container, CT).**
* **Виртуализация на уровне операционной системы имеет минимальные накладные расходы и обеспечивает самую высокую степень консолидации, однако эта технология не позволяет запускать ОС с ядрами, отличными от ядра базовой ОС.**
* **Накладные потери производительности контейнеров составляют 3%**
* Экземпляры пространств пользователя (часто называемые контейнерами или зонами) с точки зрения пользователя полностью идентичны реальному серверу, но они в своей работе используют один экземпляр ядра операционной системы.
* Для linux-систем, эта технология может рассматриваться как улучшенная реализация механизма chroot.
* Ядро обеспечивает полную изолированность контейнеров, поэтому программы из раз­ных контейнеров не могут воздействовать друг на друга.
* **Наиболее распространены сейчас OpenVZ, LXC, FreeBSD jail и Solaris Containers**

1. Соотношение контейнеров и процессов ОС. Понятие среды окружения процесса. Решение проблемы зависимостей между компонентами ПО с помощью контейнеров
2. Существующие варианты контейнеризации.
3. Механизмы изоляции, используемые при контейнеризации.

* В 1979 году в UNIX был добавлен системный вызов chroot() — с целью обеспечить изоляцию и предоставить разработчикам отдельную от основной системы площадку для тестирования. ​
* Попытки усовершенствовать механизм chroot и обеспечить более надёжную изоляцию предпринимались неоднократно: так, в частности, появились такие известные технологии, как **FreeBSD Lail**и **Solaris Zones**.​

**CHROOT(): ПЕРВАЯ ПОПЫТКА ИЗОЛЯЦИИ**

* chroot представляет собой сокращение от change root, что дословно переводится как «изменить корень». С помощью системного вызова chroot() и соответствующей команды можно изменить корневой каталог. ​
* Файловая система UNIX представляет собой древовидную иерархию:​



* В ядре Linux изоляция процессов была усовершенствована благодаря добавлению новых подсистем и новых системных вызовов:​
* Механизм пространств имен​
* Механизм групп контроля (cgroups)​
* Пространство имён (англ. namespace) — это механизм ядра Linux, обеспечивающий изоляцию процессов друг от друга на основе использования различных привилегий, назначаемых разным пользователям.​
* Работа по его реализации была начата в версии ядра 2.4.19. (авг. 2003-го).​
* На текущий момент в Linux поддерживается **шесть**типов пространств имён:​

**Каждое пространство имён для внутренних процессов, выглядит так как будто это отдельная машина.**

**МЕХАНИЗМ ИЗОЛЯЦИИ ПРОЦЕССОВ: CGROUPS**​

* Для контейнеризации одной лишь изоляции на основе привилегий доступа недостаточно. ​
* Если мы запускаем какое-либо приложение в изолированном окружении, мы должны быть уверены в том, что:​
* этому приложению выделено достаточно ресурсов ​
* что оно не будет потреблять лишние ресурсы, нарушая тем самым работу остальной системы.​
* Для решения этой задачи в ядре Linux имеется специальный механизм — cgroups (сокращение от control groups, контрольные группы).​
* Cgroups ограничивают количество ресурсов, которые разрешено использовать каждому отдельному контейнеру. ​
* Они дают ограничить доступность к ресурсам, таким как память и процессор для каждого контейнера. ​
* К контейнерам можно применить жесткие ограничения, чтобы быть уверенным в том, что процессы будут убиты, если они начинают требовать слишком много ресурсов.​

1. Методы обеспечения безопасности контейнеров.

Песочницы

* для работы с потенциально опасными, непроверенными или просто «сырыми» программами часто используются так называемые песочницы (**sandboxes**) — специально выделенные среды с жёсткими ограничениями. ​
* Для запускаемых в песочницах программ обычно сильно лимитированы:​
* доступ к сети, ​
* возможность взаимодействия с операционной системой на хост-машине ​
* работа с устройств ввода-вывода.​

Seccomp

* Контейнер не является полным аналогом песочницы  , т.к. — представляет собой более универсальную технологию. ​
* Приложение, запущенное в контейнере, вполне может получить доступ к ядру и компрометировать его. ​
* Именно поэтому в современных инструментах контейнеризации используются  различные механизмы для повышения уровня безопасности. ​
* Одним из таких механизмов является — **seccomp (**сокращение от secure computing**)**.​
* **Seccomp** (сокращение от secure computing) — механизм ядра Linuх, позволяющий процессам определять системные вызовы, которые они будут использовать. ​
* Если злоумышленник получит возможность выполнить произвольный код, **seccomp**не даст ему использовать системные вызовы, которые не были заранее объявлены.​

Seccommp и фильтр BPF

* Механизм BPF (сокращение от Berkeley Packet Filters) был изначально создан для фильтрации сетевых пакетов, но впоследствии сфера его применения существенно расширилась. ​
* Сегодня BPF используется, например, для трассировки ядра Linux. ​
* В 2012 году он был интегрирован с **seccomp**; появилась расширенная версия, которая так и называется — **seccomp-bpf**.​
* Писать для BPF — дело очень сложное. Поэтому для работы с этим механизмом безопасности часто применяют библиотеку **libseccomp**, которая предоставляет простой и удобный API для фильтрации системных вызовов.​

1. Docker – как средство быстрого развертывания контейнеров с ПО. История создания Docker. Назначение Docker.

**Docker** — это программное обеспечение для автоматизации развёртывания и управления приложениями в среде виртуализации **LXC**.

Docker позволяет «упаковать» приложение со всем его окружением и зависимостями в контейнер, который может быть перенесён на любой Linux-системе с поддержкой cgroups в ядре, а также предоставляет среду по управлению контейнерами.

Docker, например, используется в облачной платформе Cocaine, разработанной компанией Яндекс.

* Докер — это открытая платформа для разработки, доставки и эксплуатации приложений.
* Позволяющая отделить ваше приложение от вашей инфраструктуры.
* Дает возможность запускать практически любое приложение, безопасно как изолированное в контейнере.
* Проект начат как внутренняя собственническая разработка компании dotCloud, основанной Соломоном Хайксом (*Solomon Hykes*) в 2008 году с целью построения публичной PaaS-платформы с поддержкой различных языков программирования. ​
* Наряду с Хайксом в первоначальной разработке значительное участие приняли инженеры dotCloud Андреа Лудзарди (*Andrea Luzzardi*) и Франсуа-Ксавье Бурле (*François-Xavier Bourlet*). ​
* В **марте 2013**года код Docker был опубликован под лицензией Apache 2.0​
* В **октябре 2013**года выпущен релиз Havana тиражируемой IaaS-платформы OpenStack, в котором реализована поддержка Docker (как драйвер для OpenStack Nova). ​
* С **ноября 2013**года частичная поддержка Docker включена в дистрибутив Red Hat Enterprise Linux версии 6.5 и полная — в 20-ю версию дистрибутива Fedora, ранее было достигнуто соглашение с Red Hat о включении **с 2014**года Docker в тиражируемую PaaS-платформу **OpenShift**.​
* С **2014** года ведутся работы по включению поддержки Docker в среду управления фреймворка распределённых приложений **Hadoop**;​
* На тесте массового создания виртуальных вычислительных узлов прирост потребления **процессорных ресурсов**в Docker зафиксирован в **26 раз ниже**, чем в **KVM**, а прирост потребления ресурсов **оперативной памяти** — **втрое ниже**.​
* С 2017 года вдобавок к свободно распространяемой под лицензией Apache 2.0 редакции продукта выпускается редакция для организаций, продаваемая по ценам от $750 до $2 тыс. в год на узел в зависимости от доступных функций.​

1. Структура контейнера Docker. Компоненты Docker. Принципы использования контейнеров Docker.

Технология

* Docker использует следующие технологии:​
* контейнеры LXC (контейнеры Linux);​
* группы управления cgroups;​
* файловую систему вида copy-on-write (копирование при записи) **AuFS**;​
* простой набор инструментальных средств;​
* единый интерфейс API.​
* Результат: система более мощная, чем составляющие ее технологии.​

Платформа

* ПО Docker предоставляет утилиты и платформу для управления жизненным циклом контейнеров:​
* Обеспечивает инкапсуляцию приложений и поддерживаемых компонент в контейнеры Docker.​
* Обеспечивает распространение и доставку этих контейнеров командам разработчиков для дальнейшей разработки и тестирования​
* Позволяет быстро разворачивать эти контейнеры с приложениями в производственных или тестовых окружениях, не зависимо от того, что бы это ни было, локальный датацентр  (ЦОД) или облако.

Принциры использования

Многоуровневая организация образа контейнера.​

* Каждый образ состоит из набора уровней. ​
* Docker использует AuFS (Advansed multi-layred unification filesystem)  для сочетания этих уровней в один образ.​
* В основе каждого образа находится базовый образ.​
* Можно использовать имеющиеся образы как базу для создания новых образов.​

Одно приложение на контейнер.​

* Рекомендуемая практика использования Docker:​
* Базовый образ ОС не содержит общие возможности системы, такие как:​
* init, service, ssh, syslog и cron;​
* Контейнер запускается командой:​
* docker run image [command]​
* Для запуска нескольких процессов необходимо:​
* дополнительно установить соответствующий инструмент управления (такой как supervisord);​
* или запустить процесс, который в свою очередь запустит несколько новых процессов.​
* Отсутствие состояния контейнера.​

Компоненты

Платформа виртуализации с открытым кодом

Docker hub платформа как сервис для распространения и управления докер контейнерами

1. Архитектура Docker.

Элементы

* Клиент (REST API)​
* Docker Engine (демон)​
* Docker File​
* Образы (images)​
* Реестры (registries)​

Docker Hub​

* Контейнеры​

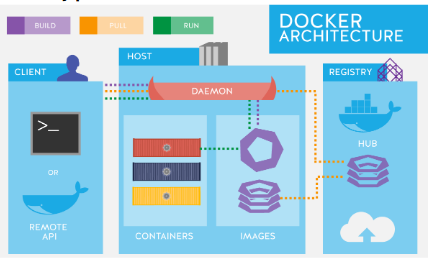
Архитектура клиент-сервер. Докер общается с демоном докер, который берёт на себя тяжесть создания, запуска, распределения контейнеров.

* Docker Engine (демон)​. Запускается на хост-машине. Пользователь не взаимодействует с сервером напрямую, а использует для этого клиент.
* Клиент (REST API)​. Получает команды от пользовтеля и взаимодействует с докер-демоном.

Внутри докера: образы, реестр, контейнеры.

Образ это шаблон(может содержать операционку), исп для создания контейнеров.

Реестр хранит образы.



* Каждый образ состоит из набора уровней.
* Docker использует AuFS (Advansed multi-layred unification folesystem) для сочетания этих уровней в один образ.
* В основе каждого образа находится базовый образ.
* Можно использовать имеющиеся образы как базу для создания новых образов.
* Образы хранятся в реестрах.
* При апгрейде среды, из различных реестров беруться образы и запускаются различные контейнеры.
* Среда компании можент быть помещена в облако Amazon.

**Выводы**

* Docker обладает следующими свойствами:
  + Не имеет состояний.
  + Декларативный.
  + Функциональный.
  + Строгий.
  + Лицензионно чистый продукт.

1. Создание образа и запуск контейнера.

**Как создать Docker образ?**

* запуск команды с указанием:
  + добавление файла или директории
  + создание переменной окружения
  + указания что запускать, когда запускается контейнер этого образа
  + инструкции хранятся в файле Dockerfile
* Docker считывает этот Dockerfile, выполняет находящиеся там инструкции и возвращает конечный образ.
* Важно понимать, что Docker ориентирован на контейнеризацию приложений, а не ОС.

**Как работает контейнер ?**

* состоит из операционной системы, пользовательских файлов и метаданных
* Docker образ доступен только для чтения
* Когда docker запускает контейнер, он создает уровень для чтения/записи сверху образа.

1. Контейнеры Windows Server и Hyper-V.

Разработчики Windows до недавнего времени предлагали две технологии виртуализации:

* 1. виртуальные машины Hyper-V;
  2. виртуальные приложения Server App-V.

В Windows Server 2016 анонсированы контейнеры (Windows Server Containers), а чуть позже и контейнеры Hyper-V .

Контейнеры Windows используют одно ядро с ОС, которое динамично разделяют между собой.

Процесс распределения (CPU, ОЗУ, сеть) берет на себя ОС.

При необходимости можно ограничить максимально доступные ресурсы, выделяемые контейнеру.

Файлы ОС и запущенные службы проецируются в пространство имен каждого контейнера.

Этот режим в чем-то напоминает FreeBSD Jail или Linux OpenVZ.

* Микрософт разработал два формата контейнеров:
  + Windows Server Containers (WSC)
  + Hyper-V containers (HVC)
* WSC – является аналогичным Linux контейнерам.
* HVC – позволяет гостевым ОС запускать «родные» приложения на «чужой платформе», посредством виртуальной машины.
* Важная особенность контейнеров в Win 2016 состоит в том, что тип выбирается не в момент создания, а в момент деплоя. То есть любой контейнер может быть запущен и как Windows, и как Hyper-V.

**Контейнеры Hyper-V**

* открывают более широкие возможности изоляции по сравнению с контейнерами Windows Server, так как каждый контейнер запускается в виртуальной машине со специальной легковесной версией ОС.
* В этой конфигурации каждый контейнер использует свою копию ядра, изолированную от других контейнеров, но при этом он обладает характеристиками обычного контейнера (быстрое развёртывание, использование библиотеки шаблонов Docker, stateless, мощные возможности по управлению).

**Основные понятия контейнера Windows**

* **Узел контейнера.** Физический или виртуальный компьютер под управлением Windows Server 2016, настроенный для работы с контейнерами Windows. На хосте работают один или несколько контейнеров Windows.
* **Образ контейнера.** По мере того как в файловую систему или реестр контейнера вносятся изменения (например, при установке программного обеспечения), они регистрируются в «песочнице». Во многих случаях может потребоваться зарегистрировать это состояние, чтобы применить внесенные изменения при создании новых контейнеров. В этом и заключается суть образа: после остановки работы контейнера можно либо отключить «песочницу», либо преобразовать ее в новый образ контейнера.
* **«Песочница».** После запуска контейнера все операции записи (изменения файловой системы и реестра либо установка программного обеспечения) регистрируются на уровне «песочницы».
* **Образ ОС контейнера.** Контейнеры развертываются из образов. Образ ОС контейнера — это первый из возможного множества слоев образа, составляющих контейнер. Этот образ представляет собой среду операционной системы. Образ ОС контейнера постоянный, его невозможно изменить.
* **Репозиторий контейнера.** При каждом создании образа контейнера этот образ и его зависимости сохраняются в локальном репозитории. Эти образы можно использовать повторно много раз на узле контейнера. Образы контейнеров также можно хранить в открытом или закрытом реестре (например, Docker Hub), чтобы использовать на многих других узлах контейнеров.

1. Использование контейнеров в Windows Azure

Azure Container Service — облачный сервис контейнеризации на базе Docker Swarm и DC/OS (Apache Mesos).

Этот сервис позволяет тысячам заказчиков по всему миру эффективно развёртывать масштабные решения с использованием контейнеров популярных форматов Docker и Mesos.

Windows Server для контейнеризации использует одну из самых популярных систем в индустрии — Docker на базе ОС Linux.

1. Служебная шина Microsoft Azure. Управление обменом сообщений через сервисную шину в Microsoft Azure.

**Service Bus** — интеграционная шина предоставляет возможности ретрансляции и безопасного обмена сообщениями и позволяет создавать распределённые и слабосвязанные приложения в облаке, а также гибридные приложения, размещенные одновременно в частных и общедоступных облачных службах. Оперирует терминами Relay, Topics, Queues, Notification Hubs. С июня 2013 года в Service Bus была внедрена глобальная доступность поддержки [открытого стандарта AMQP](http://blogs.msdn.com/b/interoperability/archive/2012/11/05/advance-message-queuing-protocol-amqp-1-0-approved-as-an-oasis-standard.aspx).

* Поддерживает набор облачных технологий промежуточного уровня, ориентированных на обработку сообщений.
* Эти технологии представлены надежными очередями сообщений, а также возможностями публикации и подписки в рамках обмена сообщениями.
* Эти возможности обмена сообщениями через посредника могут рассматриваться как разделенные функции обмена сообщениями, поддерживающие публикацию и подписку, временное разделение, а также сценарии балансировки нагрузки с использованием рабочей нагрузки обмена сообщениями служебной шины.

1. Проблемы переноса приложений в «облако».

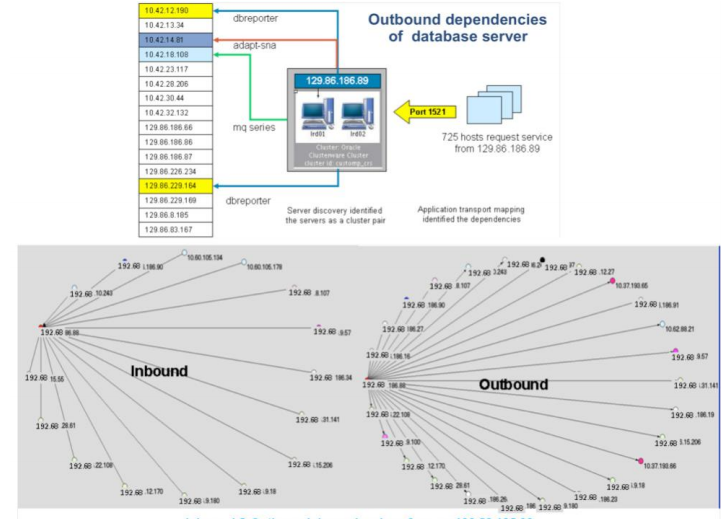
Итак, одна из основных проблем – это перенос существующих приложений в “облако”. Во-первых, это не всегда возможно сделать из-за особенностей архитектуры конкретного приложения, его привязок к другим системам или сервисам, которые ещё/уже не будут перенесены в “облака”. Зачастую переход к облачным вычислениям невозможен из-за использования специфических API ОС или вызова низкоуровневых функций для оптимизации работы. Таких приложений, может быть, не очень много, но они есть. Бывают и такие случаи, когда перенос теоретически возможен, но это требует либо значительной переработки кода, либо переписывания всего набора ПО с нуля. Очень часто это экономически невыгодно

## Ошибки и подводные камни миграции

Нередко при миграции в облако компании допускают ошибки, а все потому, что не уделяют должного внимания некоторым важным деталям. Но если соблюдать установленные рекомендации и действовать по плану, конечный результат будет оправданным и ожидаемым.

### Ошибка 1. Отсутствие схемы зависимости приложений

При миграции в облако следует уделить внимание схеме зависимости приложений друг от друга и инфраструктуры в целом. Результаты лучше всего отражать визуально в виде карты зависимостей. Допустим, приложения A, B, C используют одну и ту же базу данных. Следовательно, необходимо учесть это и проработать механизм комбинированного перемещения в облако. Если этого не сделать, велика вероятность некорректной работы приложения.



Пример схемы зависимости приложений

### Ошибка 2. Отсутствие плана миграции

Что вы будете переносить в облако, в какой последовательности, в какой момент, в какие сроки? До начала миграции подготовьте ответы на эти и другие вопросы, отобразив основные шаги в плане миграции. План позволяет прописать каждый этап переезда, избавляя от хаотичных действий и необдуманных шагов. Если в инфраструктуре компании существуют приложения, перенос которых невозможно разделить на части, следует придерживаться варианта единовременного, неделимого переноса и максимального контроля каждого выполненного шага. Анализ ситуации и выявление ошибок на ранних этапах помогут сэкономить время и добиться желаемых результатов.

### Ошибка 3. Запуск миграции без предварительных тестов

Прежде чем мигрировать в облако, выберите надежного и проверенного хостинг-провайдера. В одной из [статей](http://iaas-blog.it-grad.ru/%D0%BE%D0%B1%D0%BB%D0%B0%D1%87%D0%BD%D1%8B%D0%B9-%D1%86%D0%BE%D0%B4-%D0%BA%D0%B0%D0%BA-%D0%BF%D1%80%D0%BE%D0%B2%D0%B5%D1%80%D0%B8%D1%82%D1%8C-%D0%BD%D0%B0%D0%B4%D0%B5%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D1%8C-%D0%B4%D0%B0%D1%82%D0%B0-%D1%86%D0%B5%D0%BD%D1%82%D1%80%D0%B0-%D0%BF%D1%80%D0%B8-%D0%B2%D1%8B%D0%B1%D0%BE%D1%80%D0%B5-iaas-%D0%BF%D1%80%D0%BE%D0%B2%D0%B0%D0%B9%D0%B4%D0%B5%D1%80%D0%B0) мы уже рассказывали, как это делать. Выбрав поставщика, запросите тестовый доступ к облачной площадке и проработайте процессы миграции. Сначала перенесите в облако «легкий» сервис, оцените потраченное время, проверьте, как все работает, проанализируйте, нет ли ошибок, и перейдите к следующему сервису. Последовательно выполненные мероприятия по переносу инфраструктуры в дальнейшем дадут хорошие результаты.

1. Функциональный маппинг облачных приложений и облачных сервисов.
2. Атрибуты (характеристики) приложений.
3. Атрибуты (характеристики) облачных сервисов.
4. Пример развертывания ИС предприятия в облаке.

[**Delivery** **Club**](http://www.delivery-club.ru/)**,** абсолютный лидер российского рынка по заказу доставки еды, является примером компании, осуществившей перенос всех сервисов в облако провайдера. Поскольку заказ еды через интернет является достаточно востребованной услугой, наблюдался постоянный рост нагрузок на сервисы Delivery Club, что требовало быстрой реакции. Миграция в облако решила проблему. Теперь все сервисы компании виртуализированы и полностью отсутствует связка используемых серверов с физическими.

Для Delivery Club облако оказалось выгоднее и удобнее в плане администрирования, поддержки и обеспечения отказоустойчивости. Миграция в облако позволила компании не задумываться о замене или покупке новых серверов, а динамическое управление всем пулом выделенных ресурсов оказалось не только выгодным, но и надежным решением.

1. Пример использования облачных ресурсов при возрастании нагрузки на приложения ИС.
2. АРI облачных служб.
3. Роль стандартизации API облачных сервисов. Организации занимающиеся стандартизацией облачных сервисов.
4. Обзор существующих угроз информационной безопасности облачных вычислений

CSA (Cloud Security Alliance) представила список 12 угроз облачной безопасности, с которыми сталкиваются организации.

**Угроза 1: утечка данных:** Из-за большого количества данных, которые сегодня часто переносятся в облака, площадки облачных хостинг-провайдеров становятся привлекательной целью для злоумышленников. При этом серьезность потенциальных угроз напрямую зависит от важности и значимости хранимых данных.

Чтобы минимизировать риски и угрозы утечки данных, CSA рекомендует использовать многофакторную аутентификацию и шифрование.

**Угроза 2: компрометация учетных записей и обход аутентификации:** Утечка данных зачастую является результатом небрежного отношения к механизмам организации проверки подлинности, когда используются слабые пароли, а управление ключами шифрования и сертификатами происходит ненадлежащим образом. Кроме того, организации сталкиваются с проблемами управления правами и разрешениями, когда конечным пользователям назначаются гораздо б*о*льшие полномочия, чем в действительности необходимо. CSA рекомендует использовать механизмы многофакторной аутентификации, включая одноразовые пароли, токены, смарт-карты, USB-ключи.

**Угроза 3: взлом интерфейсов и API:** От того, насколько хорошо проработаны механизмы контроля доступа, шифрования в API, зависит безопасность и доступность облачных сервисов. При взаимодействии с третьей стороной, использующей собственные интерфейсы API, риски значительно возрастают. Почему? Потому что требуется предоставлять дополнительную информацию, вплоть до логина и пароля пользователя. Слабые с точки зрения безопасности интерфейсы становятся узким местом в вопросах доступности, конфиденциальности, целостности и безопасности. CSA рекомендует:

* организовать адекватный контроль доступа,
* использовать инструменты защиты и раннего обнаружения угроз.
* умение моделировать угрозы и находить решения по их отражению — достойная профилактика от взломов. - выполнять проверку безопасности кода и запускать тесты на проникновение.

**Угроза 4: уязвимость используемых систем**: Уязвимость используемых систем — проблема, встречающаяся в мультиарендных облачных средах.

Распространена ошибка, когда при использовании облачных решений в модели IaaS компании уделяют недостаточно внимания безопасности своих приложений, которые размещены в защищенной инфраструктуре облачного провайдера.

И уязвимость самих приложений становится узким местом в безопасности корпоративной инфраструктуры.

Рекомендации:

регулярное сканирование на выявление уязвимостей,

применение последних патчей ;

быструю реакцию на сообщения об угрозах безопасности.

**Угроза 5: кража учетных записей**: Фишинг, мошенничество, эксплойты встречаются и в облачном окружении.

Сюда добавляются угрозы в виде попыток манипулировать транзакциями и изменять данные.

Защита:

* Необходимо запретить «шаринг» учетных записей пользователей и служб между собой, а также обратить внимание на механизмы многофакторной аутентификации.
* Сервисные аккаунты и учетные записи пользователей необходимо контролировать, детально отслеживая выполняемые транзакции.
* Главное — обеспечить защиту учетных записей от кражи, рекомендует CSA.

**Угроза 6: инсайдеры-злоумышленники**: Инсайдерская угроза может исходить от нынешних или бывших сотрудников, системных администраторов, подрядчиков или партнеров по бизнесу.

Инсайдеры-злоумышленники преследуют разные цели, начиная от кражи данных до желания просто отомстить.

В случае с облаком цель может заключаться в полном или частичном разрушении инфраструктуры, получении доступа к данным и прочем.

Системы, напрямую зависящие от средств безопасности облачного поставщика, — большой риск.

CSA рекомендует:

позаботиться о механизмах шифрования и взять под собственный контроль управление ключами шифрования.

Не стоит забывать про логирование, мониторинг и аудит событий по отдельно взятым учетным записям.

**Угроза 7: целевые кибератаки**: Злоумышленника, задавшегося целью установить и закрепить собственное присутствие в целевой инфраструктуре, не так легко обнаружить. (Развитая устойчивая угроза, или целевая кибератака)

Для минимизации рисков и профилактики подобных угроз поставщики облачных услуг используют продвинутые средства безопасности.

Но помимо современных решений, требуется понимание сущности и природы такого вида атак.

CSA рекомендует:

проводить специализированное обучение сотрудников по распознаванию техник злоумышленника,

использовать расширенные инструменты безопасности,

уметь правильно управлять процессами,

знать о плановых ответных действиях на инциденты,

применять профилактические методы,

повышающие уровень безопасности инфраструктуры.

**Угроза 8: перманентная потеря данных**: Облачные хостинг-провайдеры для соблюдения мер безопасности рекомендуют отделять пользовательские данные от данных приложений, сохраняя их в различных локациях.

Не стоит забывать и про эффективные методы резервного копирования. Ежедневный бэкап и хранение резервных копий на внешних альтернативных защищенных площадках особенно важны для облачных сред.

Кроме того, если клиент шифрует данные до размещения в облаке, стоит заранее позаботиться о безопасности хранения ключей шифрования.

Как только они попадают в руки злоумышленнику, с ними становятся доступны и сами данные, потеря которых может быть причиной серьезных последствий.

**Угроза 9: недостаточная осведомленность**: Организации, которые переходят в облако без понимания облачных возможностей, сталкиваются с рисками. Если, к примеру, команда разработчиков со стороны клиента недостаточно знакома с особенностями облачных технологий и принципами развертывания облачных приложений, возникают операционные и архитектурные проблемы.

CSA напоминает о необходимости понимать функционирование облачных сервисов, предоставляемых поставщиком услуг. Это поможет ответить на вопрос, какие риски берет на себя компания, заключая договор с хостинг-провайдером.

**Угроза 10: злоупотребление облачными сервисами**: Облака могут использоваться легитимными и нелегитимными организациями.

Цель последних — использовать облачные ресурсы для совершения злонамеренных действий: з

апуска DDoS-атак, отправки спама,

распространения вредоносного контента и т. д.

Поставщикам услуг крайне важно уметь распознавать таких участников, для чего рекомендуется детально изучать трафик и использовать инструменты мониторинга облачных сред.

**Угроза 11: DDoS-атаки**: Несмотря на то что DoS-атаки имеют давнюю историю, развитие облачных технологий сделало их более распространенными.

В результате DoS-атак может сильно замедлиться или вовсе прекратиться работа значимых для бизнеса компании сервисов.

Известно, что DoS-атаки расходуют большое количество вычислительных мощностей, за использование которых будет платить клиент.

Несмотря на то что принципы DoS-атак, на первый взгляд, просты, необходимо понимать их особенности на прикладном уровне:

они нацелены на уязвимости веб-серверов и баз данных.

Облачные поставщики, безусловно, лучше справляются с DoS-атаками, чем отдельно взятые клиенты. Главное — иметь план смягчения атаки до того, как она произойдет.

**Угроза 12: совместные технологии, общие риски**: Уязвимости в используемых технологиях — достаточная угроза для облака.

Поставщики облачных услуг предоставляют виртуальную инфраструктуру, облачные приложения, но если на одном из уровней возникает уязвимость, она влияет на все окружение.

CSA рекомендует использовать стратегию «безопасности в глубину»:

внедрять механизмы многофакторной аутентификации,

системы обнаружения вторжений,

придерживаться концепции сегментирования сети,

принципа предоставления наименьших привилегий.

1. Средства обеспечения безопасности облачных вычислений.

* Предполагается что безопасное состояние облачных инфраструктур обеспечивается соблюдением требований по применению следующих трех составляющих средств защиты:

**Средства физической защиты** ЦОД с периодом обновления один год, такие как:

* + Считыватели биометрических данных;
  + ССTV – коротко-замкнутые цепи ТВ для построения систем проникновения через границы периметра защиты;
  + Системы определения перемещений по защищаемой территории;
  + Различные ловушки позволяющие обнаружить физическое проникновение посторонних на защищаемую территорию.

**Средства сетевой защиты** включая:

* + Внешние межсетевые экраны устойчивые в отказам;
  + Системы обнаружения вторжений (IDS – Intrusion Detected Systems);
  + Различные средства оценки защищенности, и определения имеющихся уязвимостей – сканеры безопасности или системы обнаружения уязвимостей.

**Платформы безопасности:**

* + Криптографической защиты данных и сетевого трафика.
  + Жесткой политики выбора паролей.

Обязательная сертификация защищенности облачной системы.

1. Средства защиты среды облачных вычислений.

Следующие облачные компоненты требуют специальной защиты:

- защита физ.серверов от враждебного ПО и вирусов.

- защита гипервизоров или мониторов VM от программных атак и уязвимостей.

- защита VM и мониторв от разрушения сервисов и DoS атак.

1. Средства защиты на уровне приложения.

* Анализ исполняемых кодов,
* Сканеры безопасности,
* МЭ уровня WebApp.
* Защита прав на использование – копирайт.

1. Средства защиты на уровне данных.

* Защита от искажения и потери данных.
* Ведение файлов журналов транзакций данных.
* Мониторинг данных.
* Шифрование данных.

1. Средства защиты на уровне управления.

* Руководство рисками потери управления облачными сервисами.
* Управление идентичностью и доступностью виртуальных машин.
* Управление обновлением ПО ОС и мониторов виртуальных машин (гипервизоров).
* Сетевые IDS/IPS
* Сетевые МЭ.
* Защита от DoS и DDoS атак.
* Обеспечение QoS.
* Применение защищенного протокола DNS – DNSSEC

1. Средства защиты на уровне управления.

* Руководство рисками потери управления облачными сервисами.
* Управление идентичностью и доступностью виртуальных машин.
* Управление обновлением ПО ОС и мониторов виртуальных машин (гипервизоров).
* Сетевые IDS/IPS
* Сетевые МЭ.
* Защита от DoS и DDoS атак.
* Обеспечение QoS.
* Применение защищенного протокола DNS – DNSSEC

1. Средства защиты на уровне доверенных вычислений.

* Trusted Computing – Доверительные вычисления, реализуется с помощью модуля TPM (trusted platform module). Назначение модуля - предотвратить выполнение вредоносного кода.
* RoT – Root of Trust, средства защиты аппаратной платформы компьютера и прежде всего:
  + Защита периметра,
  + Защита ЦПУ и ОЗУ,
  + Аппаратный криптографический акселератор.

1. Средства защиты на уровне вычислений и хранения.

* Системы IDS/IPS – (IPS – Intrusion Preventation System).
* МЭ на уровне хоста.
* Управление целостностью и ведением журналов.
* Шифрование.