|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство образования и науки Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Информационные системы и телекоммуникации»

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ***

***НА ТЕМУ:***

***Кластерная платформа анализа***

***анализа больших данных***

***на основе Kubernetes с использованием***

***Apache Spark***

Студент \_\_\_ИУ3-41М\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_**Д.К. Загребаев**\_\_\_\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель ВКР **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

Нормоконтролер **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

*2019 г.*

АННОТАЦИЯ

Расчетно-пояснительная записка с.86, рис.45, таблиц 21, источника 17.

АРХИТЕКТУРА ОБЛАЧНЫХ СИСТЕМ, АВТОМАТИЗАЦИЯ РАЗВЕРТЫВАНИЯ СЕРВИСОВ, ОБЛАЧНОЕ ХРАНИЛИЩЕ, APACHE SPARK.

Данная работа посвящена решению проблем, связанных с построением кластерной системы для анализа больших данных c использованием Apache Spark, а именно проблемам автоматизации развертывания, тестирования, обновления, удаления, балансировки нагрузки и отказоустойчивости.

В теоретической части представленной работы проводился анализ заинтересованных сторон, были сформулированы необходимые компоненты и функции системы. Также были установлены условия по обеспечению отказоустойчивости, масштабируемости.

В исследовательской части представленной работы приведен обзор и выбор наилучшей технологии для построения кластерной системы.

В конструкторской части представлен результат создания кластерной архитектуры. Был проведен анализ взаимодействия с созданной системой. Также предложен вариант по ее улучшению.

В заключении дан анализ полученных результатов и дальнейшие планы по улучшению системы.

Итогом работы является отказоустойчивая кластерная платформа, которая может в автоматическом режиме масштабироваться, хранить большой объем данных и обрабатывать их, также предоставляет удобные инструменты и средства командной разработки для анализа больших данных

**СОДЕРЖАНИЕ**

[АННОТАЦИЯ 5](#_Toc10276901)

[СОДЕРЖАНИЕ 6](#_Toc10276902)

[СПИСОК ТЕРМИНОВ 8](#_Toc10276903)

[СПИСОК УСЛОВНЫХ ОБОЗНАЧЕНИЙ 12](#_Toc10276904)

[1 Список условных обозначений бизнес-сущностей для ArchiMate-диаграмм 12](#_Toc10276905)

[2 Список условных обозначений связей для ArchiMate-диаграмм 22](#_Toc10276906)

[ВВЕДЕНИЕ 25](#_Toc10276907)

[1 Исследовательская часть 26](#_Toc10276908)

[1.1 Анализ заинтересованных сторон 26](#_Toc10276909)

[1.2 Сценарий использования системы 29](#_Toc10276910)

[1.3 Выявление характеристик платформы анализа больших данных 32](#_Toc10276911)

[1.4 Показатели качества системы 32](#_Toc10276912)

[1.5 Выводы к главе 1 39](#_Toc10276913)

[2 Конструкторская часть 40](#_Toc10276914)

[2.1 Выбор компонентов Платформы 40](#_Toc10276915)

[2.1.1 Среда для развертывания платформы 41](#_Toc10276916)

[2.1.1.1 Преднастроенные сервера 41](#_Toc10276917)

[2.1.1.2 Облако 41](#_Toc10276918)

[2.1.1.3 Голое железо (Bare metal) 42](#_Toc10276919)

[2.1.1.4 Сравнение сред развертывания Платформы 43](#_Toc10276920)

[2.1.2 Фреймворк для построения прототипов платформы анализа больших данных 45](#_Toc10276921)

[2.1.2.1 Docker Swarm 45](#_Toc10276922)

[2.1.2.2 Kubernetes 46](#_Toc10276923)

[2.1.2.3 OpenShift 47](#_Toc10276924)

[2.1.2.4 Сравнительный анализ Фреймворков 47](#_Toc10276925)

[2.1.3 Хранилище данных 48](#_Toc10276926)

[2.1.3.1 HDFS 48](#_Toc10276927)

[2.1.3.2 CephFS 51](#_Toc10276928)

[2.1.3.3 GlusterFS 51](#_Toc10276929)

[2.1.3.4 Сравнительный анализ кластерных хранилищ данных 51](#_Toc10276930)

[2.1.4 Мониторинг Платформы 52](#_Toc10276931)

[2.2 Развертывание платформы 52](#_Toc10276932)

[2.3 Выводы к главе 2 54](#_Toc10276933)

[3 Технологическая часть 55](#_Toc10276934)

[3.1 Процесс развертывания платформы на базе Kubernetes 55](#_Toc10276935)

[3.2 Процесс развертывания OpenShift платформы в облаке AWS 57](#_Toc10276936)

[3.3 Тестирование платформы 60](#_Toc10276937)

[3.4 Выводы к главе 3 65](#_Toc10276938)

[ЗАКЛЮЧЕНИЕ 66](#_Toc10276939)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 67](#_Toc10276940)

[Приложение А 69](#_Toc10276941)

[ПРИЛОЖЕНИЕ 1 Техническое задание 78](#_Toc10276942)

СПИСОК ТЕРМИНОВ

**Кластер** —группа вычислительных узлов, объединённых каналами связи и представляющая с точки зрения пользователя единый аппаратный ресурс.

**Масштабируемость** — в электронике и информатике означает способность системы, сети или процесса справляться с увеличением рабочей нагрузки (увеличивать свою производительность) при добавлении ресурсов (обычно аппаратных). Масштабируемость — важный аспект электронных систем, программных комплексов, систем баз данных, маршрутизаторов, сетей и т. п., если для них требуется возможность работать под большой нагрузкой. Система называется масштабируемой, если она способна увеличивать производительность пропорционально дополнительным ресурсам. Также под масштабируемостью понимается возможность наращивания дополнительных ресурсов без структурных изменений центрального узла системы [1].

**Шардирование** **(горизонтальное партиционирование)** – это принцип проектирования базы данных, при котором логически независимые строки таблицы базы данных хранятся раздельно, заранее сгруппированные в секции, которые, в свою очередь, размещаются на разных, физически и логически независимых серверах базы данных, при этом один физический узел кластера может содержать несколько серверов баз данных. Наиболее типовым методом горизонтального партицирования является применение хеш функции от идентификационных данных клиента, которая позволяет однозначно привязать заданного клиента и все его данные к отдельному и заранее известному экземпляру баз данных («шарду»), тем самым обеспечив практически неограниченную от количества клиентов горизонтальную масштабируемость.

Этот подход принципиально отличается от вертикального масштабирования, которое при росте нагрузки и объёма данных предусматривает наращивание вычислительных возможностей и объёма носителей информации одного сервера баз данных, имеющее объективные физические пределы — максимальное количество поддерживаемых CPU на один сервер, максимальный поддерживаемый объем памяти, пропускная способность шины и т. д.

Шардирование обеспечивает несколько преимуществ, главное из которых — снижение издержек на обеспечение согласованного чтения (которое для ряда низкоуровневых операций требует монополизации ресурсов сервера баз данных, внося ограничения на количество одновременно обрабатываемых пользовательских запросов, вне зависимости от вычислительной мощности используемого оборудования). В случае шардинга логически независимые серверы баз данных не требуют взаимной монопольной блокировки для обеспечения согласованного чтения, тем самым снимая ограничения на количество одновременно обрабатываемых пользовательских запросов в кластере в целом [1].

**Горизонтальное масштабирование** — разбиение системы на более мелкие структурные компоненты и разнесение их по отдельным физическим машинам (или их группам), и (или) увеличение количества серверов, параллельно выполняющих одну и ту же функцию. Масштабируемость в этом контексте означает возможность добавлять к системе новые узлы, серверы, процессоры для увеличения общей производительности. Этот способ масштабирования может требовать внесения изменений в программы, чтобы программы могли в полной мере пользоваться возросшим количеством ресурсов.

**Вертикальное масштабирование** — увеличение производительности каждого компонента системы с целью повышения общей производительности. Масштабируемость в этом контексте означает возможность заменять в существующей вычислительной системе компоненты более мощными и быстрыми по мере роста требований и развития технологий. Это самый простой способ масштабирования, так как не требует никаких изменений в прикладных программах, работающих на таких системах.

**Контейнеризация** — (виртуализация на уровне операционной системы, контейнерная виртуализация, зонная виртуализация) — метод виртуализации, при котором ядро операционной системы поддерживает несколько изолированных экземпляров пространства пользователя вместо одного. Эти экземпляры (обычно называемые контейнерами или зонами) с точки зрения пользователя полностью идентичны отдельному экземпляру операционной системы. Для систем на базе Unix эта технология похожа на улучшенную реализацию механизма chroot. Ядро обеспечивает полную изолированность контейнеров, поэтому программы из разных контейнеров не могут воздействовать друг на друга.

В отличие от аппаратной виртуализации, при которой эмулируется аппаратное окружение и может быть запущен широкий спектр гостевых операционных систем, в контейнере может быть запущен экземпляр операционной системы только с тем же ядром, что и у хостовой операционной системы (все контейнеры узла используют общее ядро). При этом при контейнеризации отсутствуют дополнительные ресурсные накладные расходы на эмуляцию виртуального оборудования и запуск полноценного экземпляра операционной системы, характерные при аппаратной виртуализации [1].

**Аппаратная виртуализация** – виртуализация с поддержкой специальной процессорной архитектуры. В отличие от программной виртуализации, с помощью данной техники возможно использование изолированных гостевых систем, управляемых гипервизором напрямую.

Гостевая система не зависит от архитектуры хостовой платформы и реализации платформы виртуализации.

Аппаратная виртуализация обеспечивает производительность, сравнимую с производительностью невиртуализованной машины, что дает виртуализации возможность практического использования и влечет её широкое распространение. Наиболее распространены технологии виртуализации Intel-VT и AMD-V.

**Большие данные (Big Data)** – общее название для структурированных и неструктурированных данных огромных объемов, которые эффективно обрабатываются с помощью масштабируемых программных инструментов [2].

**Облачные вычисления (cloud computing)** — модель обеспечения удобного сетевого доступа по требованию к некоторому общему фонду конфигурируемых вычислительных ресурсов (например, сетям передачи данных, серверам, устройствам хранения данных, приложениям и сервисам — как вместе, так и по отдельности), которые могут быть оперативно предоставлены и освобождены с минимальными эксплуатационными затратами или обращениями к провайдеру

СПИСОК УСЛОВНЫХ ОБОЗНАЧЕНИЙ

Список условных обозначений бизнес-сущностей для ArchiMate-диаграмм

1. Бизнес актер (Business Actor) — Business Actor определяется как субъект, который выполняет поведение в организации, такое как бизнес-процессы или функции.

Как правило, Business Actor выполняет поведение, назначенное одной или нескольким бизнес-ролям. Важно отделить актера от роли, потому что бизнес-актер может выполнять более одной бизнес-роли, а бизнес-роль может выполнять более одного бизнес-субъекта.

Бизнес-субъекты - это люди, отделы и бизнес-единицы. Они могут быть отдельными лицами или группами. Изображен на рисунке 1.

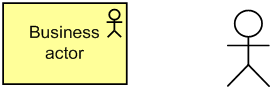


Рисунок 1 – Бизнес актер

1. Продукт (Product) – представляет собой согласованный набор услуг и / или элементов пассивной структуры, сопровождаемый договором / набором соглашений, который предлагается в целом (внутренним или внешним) клиентам.

Продукт состоит из набора Услуг и Контракта, который определяет характеристики, права и требования, связанные с Продуктом.

Продукт может объединять Бизнес-сервисы или Сервисы приложений, а также Контракт.

Продукт может предлагаться как внутри организации, так и за ее пределами.

Значение может быть связано с продуктом.

Название Продукта обычно является именем, которое используется в общении с клиентами, или, возможно, более общим существительным (например, «страхование путешествий»). Изображен на рисунке 2.

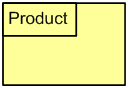


Рисунок 2 – Продукт

1. Бизнес-функция (Business Function) – описывает внутреннее поведение, выполняемое бизнес-ролью, которая требуется для производства набора продуктов и услуг. Это выполняется одной ролью в организации.

Бизнес-процессы описывают поток действий. Бизнес-функции группируют действия в соответствии с их необходимыми навыками, знаниями и ресурсами. Бизнес-процесс формирует строку бизнес-функций.

Бизнес-функция может запускаться или запускаться любым другим элементом делового поведения (бизнес-событие, бизнес-процесс, бизнес-функция или бизнес-взаимодействие).

Бизнес-функция может иметь доступ к бизнес-объектам. Бизнес-функция может реализовывать одно или несколько Business Services и может использовать (внутренние) Business Services или Application Services. Бизнес-роль или прикладной компонент могут быть назначены бизнес-функции. Изображен на рисунке 3.



Рисунок 3 – Бизнес-функция

1. Устройство (Device) – физический ИТ-ресурс, на котором системное программное обеспечение и артефакты могут быть сохранены или развернуты для выполнения. Устройство - это специализация узла, представляющая физический ИТ-ресурс с возможностью обработки. Обычно он используется для моделирования аппаратных систем, таких как мейнфреймы, ПК или маршрутизаторы. Обычно они являются частью узла вместе с системным программным обеспечением. Устройства могут быть составными; то есть состоят из-под устройств. Устройства могут быть связаны между собой сетями. Устройства могут быть назначены для Артефактов и для Системного ПО, чтобы моделировать, что Артефакты и Системное ПО развертываются на этом Устройстве. Узел может содержать одно или несколько устройств. Изображен на рисунке 4.

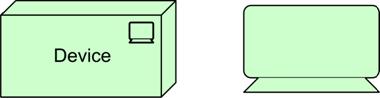


Рисунок 4 – Устройство

1. Сеть (Network) – представляет собой набор структур и поведений, которые соединяют компьютерные системы или другие электронные устройства для передачи, маршрутизации и приема данных или обмена данными на основе данных, таких как голос и видео. Сеть связи соединяет два или более устройств. Самая базовая сеть связи представляет собой единую связь между двумя устройствами. Сеть связи реализует один или несколько путей. Он воплощает физическую реализацию логического пути между узлами. Сеть связи может состоять из подсетей. Она может объединять устройства и системное программное обеспечение, например, для моделирования маршрутизаторов, коммутаторов и брандмауэров, которые являются частью сетевой инфраструктуры. Изображен на рисунке 5.

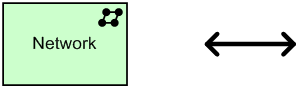


Рисунок 5 – Сеть

1. Группа (Group) – элемент агрегирует или составляет концепции, которые объединяются на основе некоторой общей характеристики. Элемент группировки используется для объединения или составления произвольной группы понятий, которые могут быть элементами и / или взаимосвязями одного и того же или разных типов. Отношение Агрегация или Состав используется для связывания элемента группировки с сгруппированными понятиями. Изображен на рисунке 6.

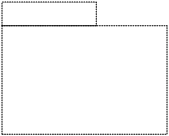


Рисунок 6 – Группа

1. Заинтересованные стороны (stakeholders) – заинтересованная сторона определяется как роль отдельного лица, команды или организации (или их классов), которая представляет их интересы или опасения относительно результатов архитектуры. Заинтересованная сторона имеет один или несколько интересов или проблем в организации и ее корпоративной архитектуре. Изображен на рисунке 7.



Рисунок 7 – Заинтересованные стороны

1. Цель (Goal) – Цель определяется как конечное состояние, которую заинтересованная сторона намерена достичь. Цель может представлять все, что может заинтересовать заинтересованную сторону , например, положение дел или произведенную ценность. Изображен на рисунке 8.



Рисунок 8 – Цель

1. Бизнес-роль (Business Role) - бизнес-роль определяется как именованное специфическое поведение бизнес-субъекта, участвующего в данном контексте. Актер выполняет поведение роли. Бизнес-роль может выполнять более одного бизнес-субъекта. И наоборот, бизнес-субъект может выполнять более одной бизнес-роли. Например, если указан учитель, его роли могут включать роли в доменах. Бизнес-роль обычно будет существовать в организации, независимо от того, выполняет ее тот или иной субъект или нет. Бизнес-роль может быть назначена одному или нескольким бизнес-процессам или бизнес-функциям. Бизнес-интерфейс или интерфейс приложения могут использоваться бизнес-ролью, в то время как бизнес-интерфейс может быть частью бизнес-роли (отношение композиции). Изображен на рисунке 9.

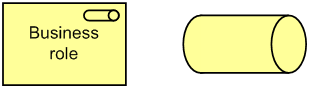


Рисунок 9 – Бизнес-роль

1. Бизнес-процесс (business process) – бизнес-процесс определяется как единица внутреннего поведения или совокупность причинно-связанных единиц внутреннего поведения, предназначенных для производства определенного набора продуктов и услуг. Бизнес-процесс описывает внутреннее поведение, выполняемое бизнес-ролью, которая требуется для производства набора продуктов и услуг. Для потребителя требуемое поведение не представляет интереса, поэтому процесс обозначен как «Внутренний». Название бизнес-процесса предпочтительно должно быть или содержать глагол в простом настоящем времени. Изображен на рисунке 10.

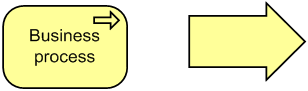


Рисунок 10 – Бизнес-процесс

1. Ограничение (constraint) – ограничение определяется как ограничение способа реализации системы. Это может быть ограничение на внедрение системы (например, конкретная технология, которая должна использоваться), или ограничение на процесс внедрения (например, ограничения по времени или бюджету). Изображен на рисунке 11.



Рисунок 11 – Ограничение

1. Требование (requirement) – требование представляет собой заявление о необходимости, которое должно быть выполнено архитектурой. Требования моделируют свойства этих элементов, которые необходимы для достижения «целей», которые моделируются целями. В этом отношении требования представляют собой «средства» для достижения целей. Изображен на рисунке 12.



Рисунок 12 – Требование

1. Компонент приложения (Application Component) – компонент приложения представляет собой инкапсуляцию функциональности приложения, выровненную по структуре реализации, которая является модульной и заменяемой. Он инкапсулирует свое поведение и данные, предоставляет сервисы и делает их доступными через интерфейсы. Компонент приложения является автономным блоком. Как таковой, он может быть независимо развернут, повторно использован и заменен. Компонент приложения выполняет одну или несколько функций приложения. Он инкапсулирует его содержимое: его функциональность доступна только через набор прикладных интерфейсов. Сотрудничающие прикладные компоненты подключаются через прикладные коллаборации. Компонент приложения может быть назначен одной или нескольким функциям приложения. Компонент приложения имеет один или несколько прикладных интерфейсов, которые предоставляют его функциональность. Изображен на рисунке 13.

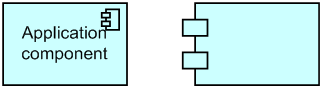


Рисунок 13 – Компонент приложения

1. Функция приложения – функция приложения представляет собой автоматизированное поведение, которое может выполнять компонент приложения. Функция приложения описывает внутреннее поведение компонента приложения. Если это поведение проявляется извне, это делается через один или несколько сервисов. Функция приложения может реализовывать одну или несколько прикладных услуг. Прикладные сервисы других Прикладных функций и Технологические сервисы могут выполнять Прикладную функцию. Функция приложения может обращаться к объектам данных. Прикладной компонент может быть назначен прикладной функции (что означает, что прикладной компонент выполняет прикладную функцию). Изображен на рисунке 14.



Рисунок 14 – Функция приложения

1. Сервис приложения (application service) – Сервис приложения представляет явно определенное поведение приложения. Сервис приложения предоставляет функциональность компонентов для их среды. Эта функциональность доступна через один или несколько прикладных интерфейсов. Служба приложения реализуется одной или несколькими функциями приложения, которые выполняются компонентом. Сервис приложения может потребовать, использовать и производить объекты данных. Сервис приложения должен обеспечивать единицу поведения, которая сама по себе полезна для ее пользователей. У него есть цель, которая заявляет об этой полезности для окружающей среды. Цель может быть связана с сервисом приложения. Служба приложений может обслуживать бизнес-процессы, бизнес-функции, бизнес-взаимодействия или функции приложений. Функция приложения может реализовывать сервис приложения. Интерфейс приложения может быть назначен сервису приложения. Прикладная служба может получить доступ к объектам данных. Изображен на рисунке 15.

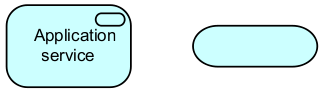


Рисунок 15 – Сервис приложения

1. Артефакт (artifact) – Артефакт представляет собой фрагмент данных, который используется или создается в процессе разработки программного обеспечения или при развертывании и эксплуатации информационной системы. Артефакт представляет собой осязаемый элемент в мире IT. Обычно он используется для моделирования (программных) продуктов, таких как исходные файлы, исполняемые файлы, сценарии, таблицы базы данных, сообщения, документы, спецификации и файлы моделей. Прикладной компонент или системное программное обеспечение может быть реализовано одним или несколькими артефактами. Объект данных может быть реализован одним или несколькими артефактами. Узел может быть назначен Артефакту для моделирования того, что Артефакт развернут на Узле. Изображен на рисунке 16.

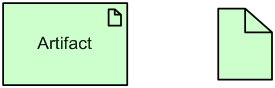


Рисунок 16 – Артефакт

1. Системное программное обеспечение (system software) - Системное программное обеспечение представляет собой программное обеспечение, которое обеспечивает или вносит вклад в среду для хранения, исполнения и использования программного обеспечения или данных, развернутых в нем. Системное ПО - это специализация узла, который используется для моделирования программной среды, в которой работают артефакты. Обычно системное программное обеспечение объединяется с устройством, представляющим аппаратную среду, для формирования общего узла. Устройство или системное программное обеспечение может быть назначено другому системному программному обеспечению; например, для моделирования различных слоев программного обеспечения, работающих поверх друг друга. Системное ПО может быть назначено для Артефактов, чтобы смоделировать, что эти Артефакты развернуты в этом программном обеспечении. Системное ПО может реализовывать другое Системное ПО. Узел может состоять из системного программного обеспечения. Изображен на рисунке 17.

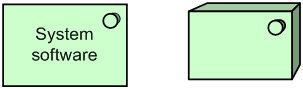


Рисунок 17 – Системное программное обеспечение

1. Бизнес интерфейс (business interface) – это точка доступа, где бизнес-сервис становится доступным для среды. Функциональность, предоставляемая бизнес-ролью, предоставляется ее среде посредством одного или нескольких бизнес-интерфейсов. И наоборот, бизнес-роль может использовать функции, предоставляемые другими бизнес-ролями через бизнес-интерфейс. Бизнес-интерфейс определяет, как функциональность бизнес-роли может использоваться другими бизнес-ролями или функциональностью, которую бизнес-роли требуют от своей среды. Бизнес-интерфейс предоставляет бизнес-услугу, предоставляемую бизнес-ролью или бизнес-сотрудничеством, своей среде. Бизнес-сервис также может быть доступен через различные интерфейсы. Изображен на рисунке 18.

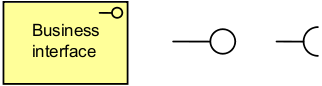


Рисунок 18 – Бизнес интерфейс

1. Интерфейс приложения представляет собой точку доступа, где службы приложений становятся доступны пользователю, другому компоненту приложения или узлу. Интерфейс приложения определяет, как функциональность компонента может быть доступна другим элементам. Интерфейс приложения предоставляет службы приложений среде. Интерфейс приложения может быть частью компонента приложения. Интерфейс приложения может быть назначен службам приложений, что означает, что интерфейс предоставляет эти службы среде. Изображен на рисунке 19.

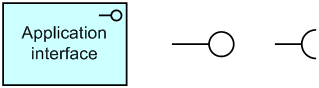


Рисунок 19 – Интерфейс приложения

Список условных обозначений связей для ArchiMate-диаграмм

1. Связь ассоциации (association relationship) – связь ассоциации моделирует отношения между объектами, которые не охватываются другими, более конкретными отношениями. Используется для моделирования отношений между бизнес-объектами или объектами данных, которые не моделируются стандартными отношениями. Связь ассоциация также используется для связи информационных понятий с другими понятиями: бизнес-объект с представлением, представление со смыслом и бизнес-служба с целью. Связи ассоциации могут использоваться при рисовании первой высокоуровневой модели, где отношения первоначально обозначаются общим образом, а затем уточняются, чтобы показать более конкретные типы отношений. Связь изображена на рисунке 20.

G:\Dropbox\МГТУ\Дипломы\Гутников Илья весна 2015\Гутников\РПЗ\Картинки\обозначения\связи Archi\image106.png

Рисунок 20 – Связь ассоциации

1. Отношение реализации (realization relationship) – Отношение Реализация связывает логическую сущность с более конкретной сущностью, которая ее реализует. Отношение Реализация указывает, как логические объекты («что» или «логические»), такие как сервисы, реализуются посредством более конкретных объектов («как» или «физический»). Бизнес-процесс или функция может реализовать Услугу. Объект данных может реализовывать бизнес-объект. Артефакт может реализовать прикладной компонент. Связь изображена на рисунке 21.

G:\Dropbox\МГТУ\Дипломы\Гутников Илья весна 2015\Гутников\РПЗ\Картинки\обозначения\связи Archi\image100.jpg

Рисунок 21 – Отношение реализации

1. Отношение назначение (assignment relationship) – отношение связывает активные элементы (например, бизнес-роли или компоненты приложения) с выполняемыми ими единицами поведения или бизнес-субъектами с бизнес-ролями, которые они выполняют. Связь изображена на рисунке 22.

G:\Dropbox\МГТУ\Дипломы\Гутников Илья весна 2015\Гутников\РПЗ\Картинки\обозначения\связи Archi\image098.png

Рисунок 22 – Отношение назначение

1. Отношение специализации (specialization relationship) - отношение специализации указывает, что объект является специализацией другого объекта. Отношения специализации могут связывать любой экземпляр концепции с другим экземпляром той же концепции. Специализация всегда возможна между двумя экземплярами одного и того же понятия. Связь изображена на рисунке 23.

G:\Dropbox\МГТУ\Дипломы\Гутников Илья весна 2015\Гутников\РПЗ\Картинки\обозначения\связи Archi\image116.jpg

Рисунок 23 – Отношение специализации

1. Отношение влияния (influence relationship) – отношение влияния используется для описания того, что некоторый мотивационный элемент может влиять (реализация) другого мотивационного элемента. Влияние какого-либо другого мотивационного элемента может повлиять на эту степень положительно или отрицательно Атрибуты могут использоваться для указания знака и / или силы влияния. Выбор возможных значений атрибута остается за архитектором; например, [0..10]. По умолчанию отношение влияние моделирует вклад с неопределенным знаком и силой. Связь изображена на рисунке 24.

G:\Dropbox\МГТУ\Дипломы\Гутников Илья весна 2015\Гутников\РПЗ\Картинки\обозначения\связи Archi\image203.jpg

Рисунок 24 – Отношение влияния

1. Отношение используется (used by relationship) – моделирует использование сервисов процессами и функциями, взаимодействия и доступа к интерфейсам ролей и компонентов. Связь изображена на рисунке 25.

G:\Dropbox\МГТУ\Дипломы\Гутников Илья весна 2015\Гутников\РПЗ\Картинки\обозначения\связи Archi\image102.png

Рисунок 25 – Отношение используется

1. Отношение композиции (composition relationship) - отношение композиция указывает, что объект состоит из ряда других объектов. В отличие от отношения агрегации, объект может быть частью только одной композиции. Отношение композиции всегда разрешено между двумя экземплярами одного и того же типа элемента. Связь изображена на рисунке 26.

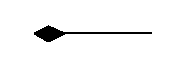


Рисунок 26 – Отношение композиции

ВВЕДЕНИЕ

К современным информационным системам предъявляются серьезные требования по функциональности, времени ответа и другим параметрам. Зачастую, современные системы обрабатывают, хранят большие объемы данных, обрабатывают миллионы запросов в секунду и должны давать ответ пользователю незамедлительно. Такие системы обычно не умещаются на одном физическом сервере. Они работают распределено на многих физических серверах.

Обеспечение правильной балансировки распределения нагрузки на систему и ее отказоустойчивость - две фундаментальные задачи в облачных системах. В такие системы обычно на стадии проектирования закладывают принципы масштабируемости, отказоустойчивости.

Немаловажную роль в облачных системах играют механизмы развёртывания сервисов. Зачастую в крупных проектах количество сервисов превышает сотню. Даже полсотни сервисов проблематично развёртывать на сервере вручную, т. к. это чревато ошибками при конфигурировании этих сервисов. Еще хуже дела становятся, когда в система разрастается до таких размеров, что не помещается на одном сервере. Правильно развернуть и сконфигурировать сотню сервисов на разных машинах практический невозможно, т. к. этим могут заниматься разные люди, да и один и тот же человек может попросту потерять бдительность. В связи с этим появляется необходимость автоматизировать процесс выкладки развертывания и настройки серверов и сервисов, что будут на них развернуты.

Решению проблем отказоустойчивости, распределению нагрузки системы и облачному хранению данных посвящена эта работа.

# Исследовательская часть

Перед тем, как приступить к разработке системы, необходимо провести анализ заинтересованных сторон и сценариев использования Системы; определить требования к Системе, которым Система должна соответствовать.

На данный момент существует эффективное средство обработки большого объема данных – Apache Spark [3]. Но сам по себе этот инструмент достаточно сложно развернуть вручную в кластерных масштабах. Для решения этой проблемы будет разработана архитектура кластерная платформы для легкого и быстрого размещения служб Apache Spark в кластере.

## Анализ заинтересованных сторон

Для разрабатываемой системы был проведен анализ заинтересованных сторон, и были определены стороны, которые были отражены в диаграмме и в таблице заинтересованных сторон.

Таблица заинтересованных сторон представлена в таблице 1, а диаграмма, иллюстрирующая заинтересованных сторон, изображена на рисунке 27.

Таблица 1 - Заинтересованные стороны и их интересы

|  |  |  |
| --- | --- | --- |
| **Роль** | **Заинтересованная сторона** | **Интересы** |
| Пользователи | Администратор | Отдача команд системе |
| Разработчики | Разработчик ПО | Разработка системы, обслуживание, исправление ошибок, обнаруженных при работе системы |
| Тестировщик | Определение состояния системы |
| Ведущий разработчик системы | Контроль за деятельностью разработчиков программного обеспечения |
| Поставщики сторонних Docker-образов | Прозрачная выкладка контейнеров в рабочую систему |
| Вредители | Конкурент | Копирование технологий |
| Злоумышленник | Получение привилегированных прав при работе с системой, сбой системы |
| Заказчики | Заказчик | Соответствие формальным требованиям, производительность системы |
| Клиенты | Клиент | Доступность системы, работоспособность системы отзывчивость системы |



Рисунок 27 – Заинтересованные стороны

В блоке «Система» есть люди, которые имеют прямой доступ к системе во время ее работы.

Блок «Сопровождающая среда» отображает стороны, сопровождающие систему и обеспечивающие качество ее работы.

В блоке «Воздействующая среда» отображаются стороны, которые могут быть представлены во время разработки системы или во время ее работы.

В блоке «Внешняя среда» отображается сторона, которая сможет использовать систему в качестве конечного продукта.

Для выявленных заинтересованных сторон были определены их представители, обратимся к диаграмме на рисунке 28. 

Рисунок 28 – Представители заинтересованных сторон

## Сценарий использования системы

У каждой из заинтересованных сторон есть интересы, которые они преследуют при взаимодействии с разрабатываемой системой. Цели взаимоотношений с заинтересованными сторонами изображены на рисунке 29.

Рисунок 29. Заинтересованные лица и преследуемые ими цели

Основной сценарий использования системы планируется:

**Основной сценарий:**

1. Администратор разворачивает кластер.
2. Администратор разворачивает в кластере все необходимые сервисы.
3. Разработчик или администратор может загрузить данные в кластер.
4. Разработчик пишет программный код и запускает его в кластере.
5. Кластер обрабатывает программный код и показывает разработчику результат.

Далее рассмотрим диаграмму сценариев, представленную на рисунке 30.



*Рисунок 30. Диаграмма рабочих сценариев*

## Выявление характеристик платформы анализа больших данных

После того как была выполнена работа по анализу заинтересованных сторон, необходимо провести работу по выявлению основных характеристик платформы анализа больших данных.

Основными характеристиками платформы являются:

* Легкое развертывание сервисов;
* Система мониторинга;
* Надежное и быстрое хранилище данных;
* Отказоустойчивость;
* Масштабируемость;
* Предоставление интерфейса для загрузки данных в кластер
* Предоставления интерфейса пользователю для написания кода, который будет обрабатывать Apache Spark анализировать данные;

В идеальном сценарии при выполнении всех требований платформу можно будет относительно легко адаптировать под разные Фреймворки, речь о которых пойдет ниже.

## Показатели качества системы

Показатели качества системы – характеристики, которые с минимальным дублированием описывают качество программного обеспечения.

Показатели качества разрабатываемой системы представлены в таблицах 2 – 7

Таблица 2 – Показатели производительности системы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Источник стимула** | **Стимул** | **Условия** | **Артефакт** | **Реакция** | **Количественная мера** |
| Администратор | Обновление сервисов | Нормальные условия | Платформа | Последовательное обновление сервисов в кластере, относящихся к одному приложению | Менее 1 секунды |
| Администратор | Добавление нового сервиса в кластер | Нормальные условия | Платформа | Создание экземпляров сервисов | Реакция платформы менее 1 секунды |
| Платформа | Детектирование сбоя приложения | Нормальные условия | Платформа | Платформа пытается перезапустить приложение | Менее 30 секунд |
| Администратор | Загрузка данных | Нормальные условия | Платформа | Платформа начинает загрузку данных в распределенную файловую систему | Менее 1 секунд |

Таблица 3 – Показатели модифицируемости платформы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Источник стимула** | **Стимул** | **Условия** | **Артефакт** | **Реакция** | **Количественная мера** |
| Разработчик платформы | Написание плагинов для платформы | В период разработки | Исходный код плагина платформы | Не потребовалось существенных изменений и доработок ранее разработанных частей платформы | В течение часа |
| Разработчик  платформы | Подключение плагинов к платформе | В период разработки | Исходный код плагина платформы | Не потребовалось существенных изменений и доработок ранее разработанных частей платформы | В течение часа |

Таблица 4 – Показатели готовности платформы

| **Источник стимула** | **Стимул** | **Условия** | **Артефакт** | **Реакция** | **Количественная мера** |
| --- | --- | --- | --- | --- | --- |
| Платформа | При развёртывании приложения произошла ошибка | Нормальные условия | Платформа | Журналирование ошибки, выдача ошибки | Платформа продолжает работать |
| Платформа | Длительное бездействие | Нормальные условия | Платформа | Платформа продолжает работу | Платформа работает в штатном режиме |
| Платформа | Аварийная ситуация | Нормальные условия | Платформа | Выдача сообщения об ошибке работы платформы, перезапуск компонента платформы, журналирование ошибки | Платформа работает в штатном режиме |

Таблица 5 – Показатели контролепригодности системы

| **Источник стимула** | **Стимул** | **Условия** | **Артефакт** | **Реакция** | **Количественная мера** |
| --- | --- | --- | --- | --- | --- |
| Тестировщик платформы | Тестирование функций платформы | В момент тестирования платформы | Платформа | Выполнение команд, пришедших от тестировщика | Задействованы весь необходимый функционал платформы |
| Тестировщик платформы | Нахождение ошибок в работе платформы | В момент тестирования платформы | Платформа | Платформа работает без ошибок | Задействовано не менее 90% функциональных возможностей Платформы |
| Тестировщик платформы | Контроль соблюдения принятых соглашений | В момент тестирования платформы | Платформа | Конфигурация платформы соответствует принятым соглашениям | 100% конфигураций платформы принятым соглашениям |
| Тестировщик платформы | Добавление новых модулей | В момент тестирования системы | Платформы | Новый модуль добавляется в платформу без возникновения ошибок | 100% модулей, удовлетворяющих интерфейсным требованиям |

Таблица 6 – Показатели практичности системы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Источник стимула** | **Стимул** | **Условия** | **Артефакт** | **Реакция** | **Количественная мера** |
| Конечный пользователь | Использование возможности платформы | В момент пользования платформы | Платформа | Повторное использование конфигураций | 80% объёма конфигурации можно использовать повторно |
| Конечный пользователь | Легкое и понятное пользование платформой | В момент пользования платформы | Платформа | Платформа имеет понятный и хорошо задокументированный интерфейс | Интерфейс взаимодействия с платформой понятен для конечного пользователя |
| Конечный пользователь | Сведение к минимуму количества ошибок при использовании платформы | В момент пользования платформы | Платформа | Саморазвертываемость платформы | Функционирование платформы не требует взаимодействия с конечным пользователем |

Таблица 7 – Атрибуты безопасности

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Источник стимула** | **Стимул** | **Условия** | **Артефакт** | **Реакция** | **Количественная мера** |
| Взломщик | Кража данных пользователя | В момент использования | Данные в платформе | Система проверяет доступность данных пользователю, основываясь на RBAC | Данные пользователя находятся под защитой от неавторизованного доступа |
| Взломщик | Получение доступа к управлению платформой | В момент использования | База знаний | Система проверяет доступность данных и механизмов управления для пользователя, основываясь на RBAC | Данные пользователя находятся и элементы управления платформой находтся под защитой от неавторизованного доступа |

## Выводы к главе 1

В первом разделе представленной квалификационной работы магистра был проведен анализ, выявивший: заинтересованные стороны платформы анализа больших данных, функции платформы, критерии показателя качества платформы.

Основываясь на анализе заинтересованных сторон и ресурсов глобальной сети Интернета, были получены реальные представители заинтересованных сторон. После чего был проведен анализ по выявлению целей заинтересованных сторон и определению сценариев и ограничений платформы.

Затем были проанализированы основные функции платформы анализа больших данных, представленные в пункте 1.3.

Основываясь на функциях платформы анализа больших данных и выявленных сценариях использования платформы анализа больших данных, и ограничений платформы анализа больших данных был проведен анализ по выявлению атрибутов качества платформы анализа больших данных, которые представленные в пункте 1.4.

Результатом всей описанной выше работы является техническое задание, представленное в приложении 1.

# Конструкторская часть

Исходя из проведенного анализа заинтересованных сторон, выявления основных функций Платформы, показателей качества Платформы и технического задания была предложена и поэтапно реализовывалась следующая архитектура кластерной платформы анализа больших данных. Базовая архитектура представлена на рисунке 31.

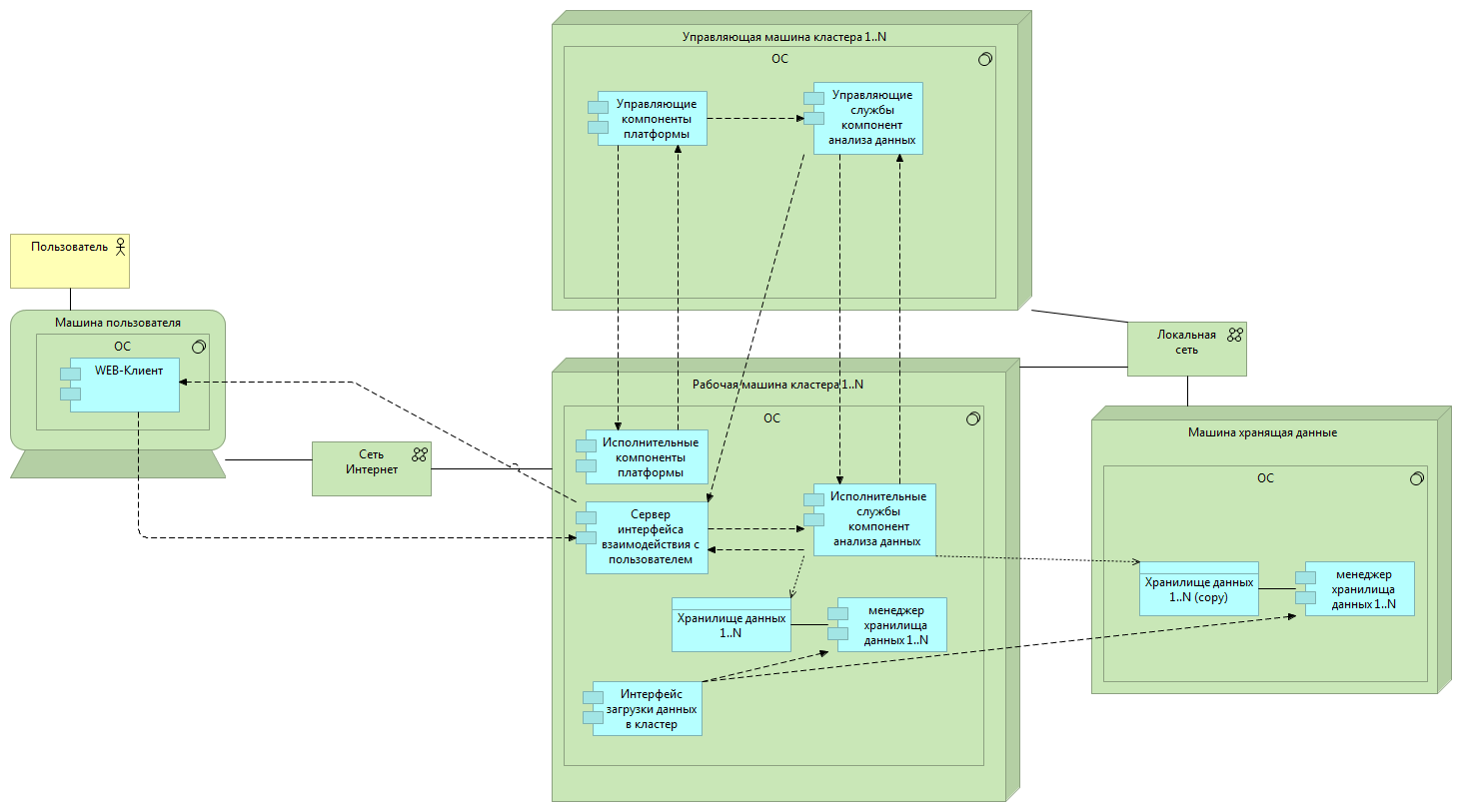


Рисунок 31. Базовая архитектура платформы анализа больших данных

Описание развертывания платформы в кластере описано в разделе 2.2

## Выбор компонентов Платформы

В данном разделе представлен анализ компонентов системы, которые лягут в основу ее реализации. Необходимо выбрать несколько компонент для построения платформы:

* Среду для развертывания платформы
* Фреймворк для построения прототипов платформы анализа больших данных
* Компонент, отвечающий за хранение данных
* Компонент мониторинга системы

В последующих разделах будет приведен анализа выбора компонент для построения платформы, с описанием результатов применения некоторых из них на практике.

### Среда для развертывания платформы

#### Преднастроенные сервера

Преднастроенные сервера – сервера, на которых уже установлены и сконфигурированы некоторые компоненты, например, операционная система или среда выполнения приложения. Такой подход хорош тем, что системный администратор зачастую контролирует всю инфраструктуру.

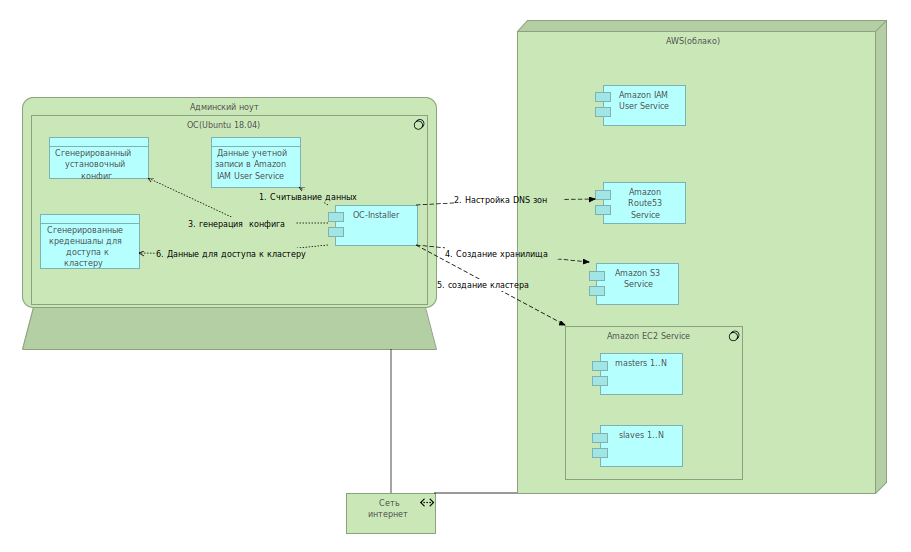
На базе четырех машин был построен Kubernetes кластер, в котором был развернут Apache Spark со всеми необходимыми компонентами и развернута распределенная файловая система – CephFS[5].

#### Облако

Облако – платформа для развертывания приложений. В настоящее время существует множество облачных провайдеров. Для развертывания прототипа платформы анализа больших данных был выбран AWS, как наиболее популярный провайдер.

В ходе экспериментов с облаком удалось развернуть платформу на фреймворке Openshift. Более подробно с процессом развертывания можно ознакомиться в моей статье [4].

Процесс развертывания показан на рисунке 32.

Рисунок 32. Процесс развёртывания платформы на AWS

Был выявлен главный недостаток такого подхода – дорого. Около 100000 рублей на работу четырех машин на протяжении месяца.

#### Голое железо (Bare metal)

Физические серверы предоставляют пользователям исключительный доступ ко всему серверу. В отличие от виртуального сервера с несколькими арендаторами, чистый физический сервер предполагает работу одного арендатора без гипервизора, исключая тем самым проблему «шумных соседей» и закономерное снижение производительности при наличии гипервизора [6].

Данный подход предоставляет гибкость облака, но очень сложен в развертывании. В настоящее время создается прототип платформы на базе этого подхода с использованием машин, которые были применены в 2.1.1.1.

Используя данный подход, ведется работа по разворачиванию платформы на базе OpenShift. Основная сложность заключается в том, что требуется развернуть дополнительную инфраструктуру для инициализации и сетевой загрузки [7] операционных систем.

#### Сравнение сред развертывания Платформы

В таблице 8 представлены основные среды для развертывания платформы анализа больших данных.

Анализ и результаты применения каждой среды, представлены в таблице 8, описаны в разделах **Ошибка! Источник ссылки не найден.** – 2.1.1.3. К каждой из сред применялись следующие критерии:

1. Воспроизводимость развертывания;
2. Дешевизна развертывания;
3. Наличие инструментов для реализации платформы;
4. Подход должен быть актуальным на состояние 2019 года;

Таблица 8 – Сравнительный анализ сред развертывания платформы

| **Среда развертывания** | **Воспроизводимость** | **Дешевизна развертывания** | **Наличие инструментов для реализации платформы** | **Актуальность на 2019 год** |
| --- | --- | --- | --- | --- |
| Преднастроенные сервера | Легко | Единовременные затраты на сервера | Да | Актуально |
| Облако | Легко | Постоянные траты на работу серверов | Да | Актуально |
| Голое железо | Требует усилий | Единовременные затраты на сервера | Да | Актуально |

На основе проведенного анализа были выбраны две среды для развёртывания платформы. Преднастроенные сервера – для первых двух прототипов, Голое железо – для финальной версии платформы.

### Фреймворк для построения прототипов платформы анализа больших данных

#### Docker Swarm

Данный фреймворк [8] не подходит по ТЗ, т.к. не основан на Kubernetes. Но все же рассмотрим его, т.к. на его базе был создан первый прототип платформы.

В режиме swarm все ноды делятся на два типа: manager и worker. При этом полноценный кластер может обходиться вообще без рабочих серверов, то есть менеджеры по умолчанию являются также и рабочими.

Среди менеджеров всегда присутствует один, который на данный момент является лидером кластера. Все управляющие команды, которые выполняются на других менеджерах автоматически перенаправляются на него. На рисунке 33 показана архитектура Swarm кластера.

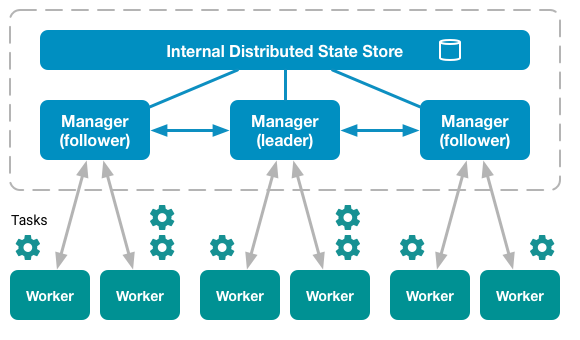


Рисунок 33. Архитектура Swarm кластера

#### Kubernetes

Kubernetes [9] является проектом с открытым исходным кодом, предназначенным для управления кластером контейнеров Linux как единой системой. Kubernetes управляет и запускает контейнеры Docker на большом количестве хостов, а также обеспечивает совместное размещение и репликацию большого количества контейнеров. Проект был начат Google и теперь поддерживается многими компаниями, среди которых Microsoft, RedHat, IBM и Docker.

Компания Google пользуется контейнерной технологией уже более десяти лет. Она начинала с запуска более 2 млрд контейнеров в течение одной недели. С помощью проекта Kubernetes компания делится своим опытом создания открытой платформы, предназначенной для масштабируемого запуска контейнеров.

Проект преследует две цели. Если вы пользуетесь контейнерами Docker, возникает следующий вопрос о том, как масштабировать и запускать контейнеры сразу на большом количестве хостов Docker, а также как выполнять их балансировку. В проекте предлагается высокоуровневый API, определяющее логическое группирование контейнеров, позволяющее определять пулы контейнеров, балансировать нагрузку, а также задавать их размещение.

На рисунке 34 изображена архитектура созданной платформы.

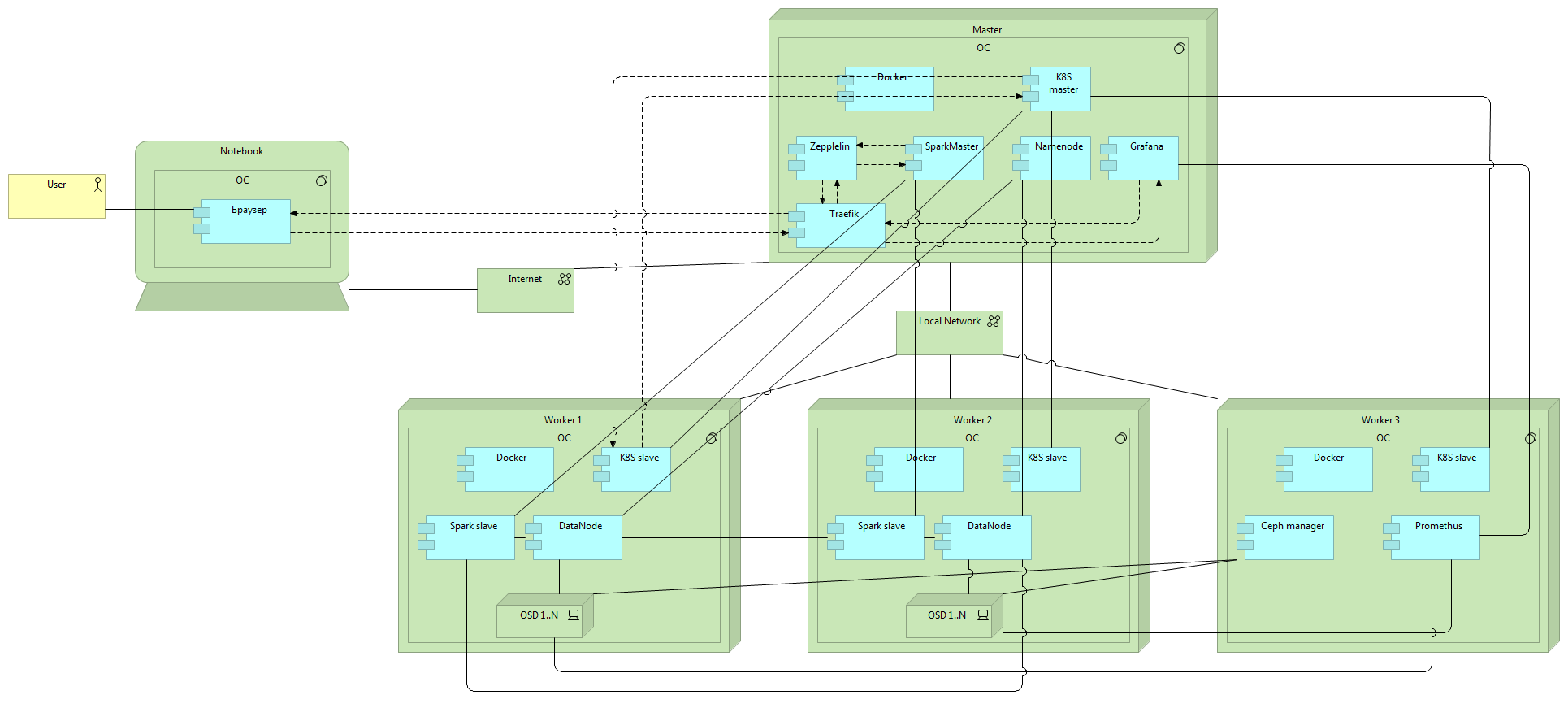


Рисунок 34. Платформа на основе Kubernetes

#### OpenShift

OpenShift [10] – модифицированная версия Kubernetes, которая в отличии от своего отца уже содержит в себе настроенные компоненты мониторинга, масштабирования, драйверы файловых систем.

На базе этой платформы строится заключительный прототип кластерной платформы анализа больших данных на голом железе. Был создан также прототип в облаке, но от этого решения отказался, т.к. оно дорогое.

Данная платформа обладает рядом преимуществ в сравнении с обычным Kubernetes. В своей последней версии данная платформа обзавелась автоматическим установщиком, который, правда, пока хорошо работает только в облачном сценарии, может производить автоматическое обновление как самой платформы, так и контейнеров, запущенных внутри нее без остановки самой платформы.

#### Сравнительный анализ Фреймворков

Сравнительный анализ фреймворков представлен в таблице 9.

Таблица 9. Сравнительный анализ фреймворков

|  |  |
| --- | --- |
| **Фреймворк** | **Поддержка Kubernetes** |
| Docker Swarm | Нет |
| Kubernetes | Да |
| OpenShift | Да |

Так как от фреймворка требуется поддержка Kubernetes, то были выбраны Kubernetes и OpenShift.

На базе Kubernetes было построен один работоспособный прототип с распределенной файловой системой CephFS в качестве хранилища. Но оказалось, что построенную платформу достаточно трудно обновлять вручную, поэтому было решено финальную версию прототипа построить на базе OpenShift.

### Хранилище данных

В данном разделе будет дан анализ кластерных хранилищ данных, пригодных для хранения данных кластерной платформы обработки больших данных на основе Apache Spark и опыт использования некоторых из них.

Кластерное хранилище данных должно отвечать следующим требованиям: быть масштабируемым, отказоустойчивым, отзывчивым, должно быть поддерживаемым и развиваемым сообществом на 2019 год.

#### HDFS

HDFS - файловая система [11], предназначенная для хранения файлов больших размеров, поблочно распределённых между узлами вычислительного кластера. Все блоки в HDFS (кроме последнего блока файла) имеют одинаковый размер, и каждый блок может быть размещён на нескольких узлах, размер блока и коэффициент репликации (количество узлов, на которых должен быть размещён каждый блок) определяются в настройках на уровне файла. Благодаря репликации обеспечивается устойчивость распределённой системы к отказам отдельных узлов. Файлы в HDFS могут быть записаны лишь однажды (модификация не поддерживается), а запись в файл в одно время может вести только один процесс. Организация файлов в пространстве имён — традиционная иерархическая: есть корневой каталог, поддерживается вложение каталогов, в одном каталоге могут располагаться и файлы, и другие каталоги.

Развёртывание экземпляра HDFS предусматривает наличие центрального узла имён (англ. name node), хранящего метаданные файловой системы и метаинформацию о распределении блоков, и серии узлов данных (англ. data node), непосредственно хранящих блоки файлов. Узел имён отвечает за обработку операций уровня файлов и каталогов — открытие и закрытие файлов, манипуляция с каталогами, узлы данных непосредственно отрабатывают операции по записи и чтению данных. Узел имён и узлы данных снабжаются веб-серверами, отображающими текущий статус узлов и позволяющими просматривать содержимое файловой системы. Административные функции доступны из интерфейса командной строки.

HDFS является традиционным хранилищем для Apache Spark. Были попытки заменить HDFS на CephFS, но не получилось их сынтегрировать друг с другом. Диаграмма интеграции CephFS и HDFS представлена на рисунке 35.

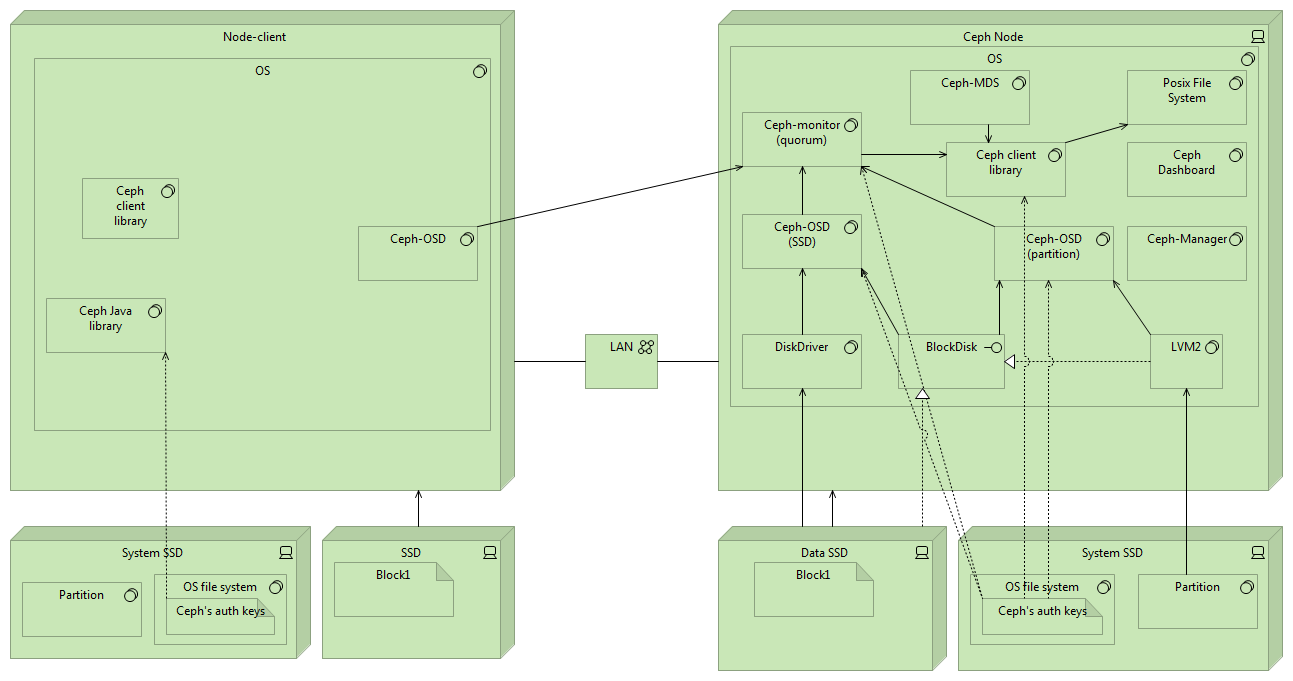


Рисунок 35. Интеграция CephFS и HDFS

#### CephFS

Файловая система Ceph (Ceph FS) - это POSIX-совместимая файловая система, которая использует Ceph Storage Cluster для хранения своих данных. Файловая система Ceph использует ту же систему кластера хранения Ceph, что и блочные устройства Ceph, хранилище объектов Ceph с его API-интерфейсами S3 и Swift или собственные привязки (librados).

CephFS позвоялет построить многоуровневую файловую, т.е. можно настроить быстрый кеш, основанный на SSD дисках, а нижняя файловая система будет представлять из себя хранилище из HDD дисков, предназначенных для надежного хранения больших объемов данных.

#### GlusterFS

GlusterFS [12] - это распределённая, параллельная, линейно масштабируемая файловая система с возможностью защиты от сбоев. С помощью InfiniBand RDMA или TCP/IP GlusterFS может объединить хранилища данных, находящиеся на разных серверах, в одну параллельную сетевую файловую систему. GlusterFS работает в пользовательском пространстве при помощи технологии FUSE, поэтому не требует поддержки со стороны ядра операционной системы и работает поверх существующих файловых систем (ext3, ext4, XFS, reiserfs и т. п.). В отличие от других распределённых файловых систем, таких как Lustre и Ceph, для работы GlusterFS не требуется отдельный сервер для хранения метаданных.

#### Сравнительный анализ кластерных хранилищ данных

Не получилось заменить HDFS ни на одну аналогичную файловую систему. Поэтому пришлось использовать следующую модель: HDFS работает поверх кластерной файловой системы, которая использует физические диски, и которая используется платформой. Платформа выделяет место и папку для конкретного узла HDFS.

В качестве кластерной файловой системы был выбран CephFS по ряду причин: актуальная документация, самое быстрое развитие из всех файловых систем с открытым исходным кодом, поддержка от сообщества.

На базе этой файловой системы построен прототип платформы с использованием чистого Kubernetes.

### Мониторинг Платформы

В данном разделе будет представлен обзор и анализ современных средств мониторинга кластеров на основе Kubernetes

В качестве дашборда для анализа метрик выбрана Grafana [13], т.к. на сегодняшний день это единственный визуализатор данных, поддерживающий множество источников данных и их форматов. С помощью Grafana возможно визуализировать метрики как самой платформы, так и контейнеров запущенных внутри нее. Также с помощью Grafana можно визуализировать метрики, поступающие от Ceph.

На данный момент широко используются две системы хранилища данных – Prometheus [14] и Influx DB [15]. Выбираем Prometheus, т.к. этот проект одобрен CNCF [16]. Подробнее про средства мониторинга можно прочитать в моей статье [17]

## Развертывание платформы

Развертывание платформы проходило в несколько этапов.

На первом этапе платформа была развернута в Docker Swarm кластере без распределенной файловой системы.

Далее платформа разворачивалась на преднастроенных серверах, рисунок 36. В качестве основы был взят Kubernetes v.1.9.6.

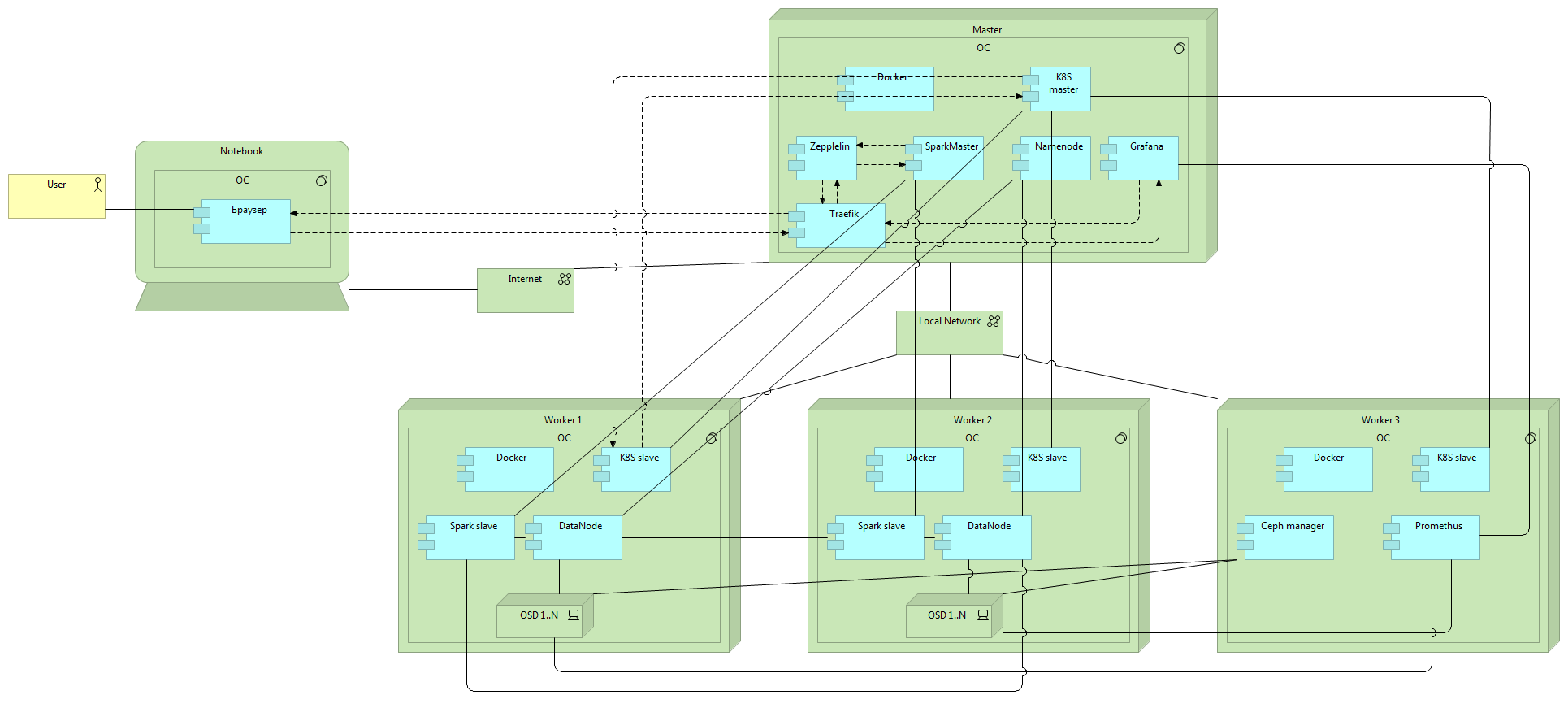


Рисунок 36. Платформа развернута на преднастроенных серверах

Затем был развернут OpenShift кластер в облаке AWS. Процесс развертывания представлен на рисунке 37

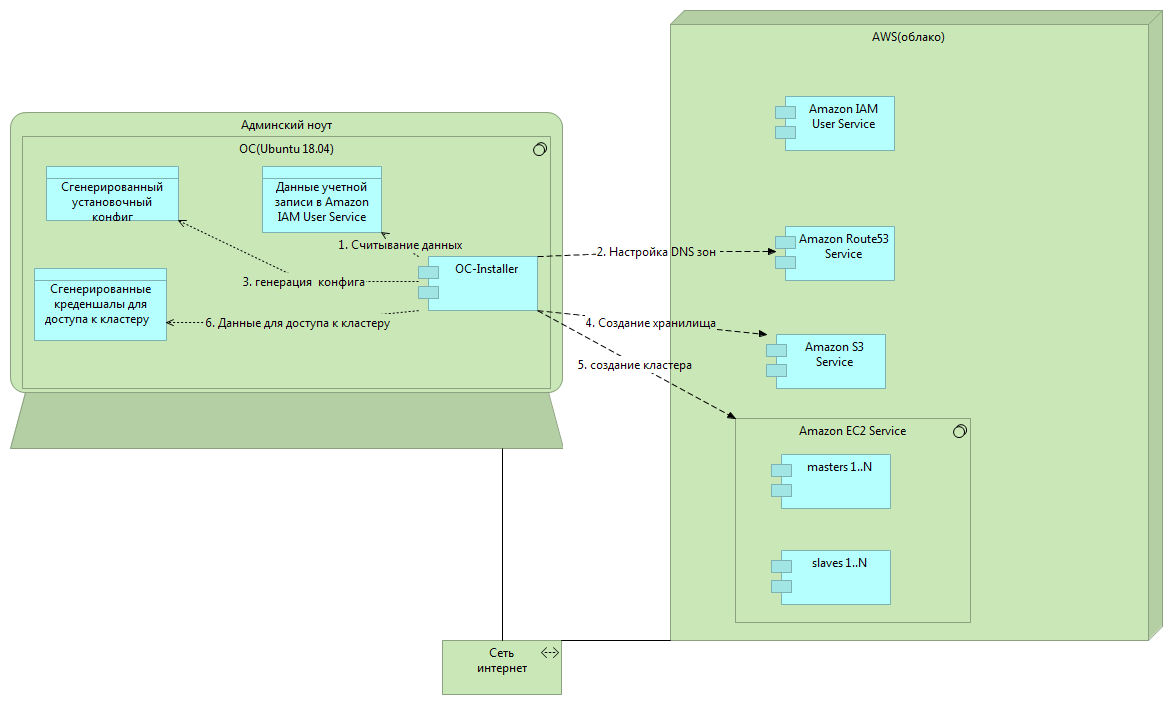


Рисунок 37. Процесс развертывания OpenShift кластера в облаке AWS

В текущий момент идут работы по развертыванию платформы на голом железе на базе OpenShift.

## Выводы к главе 2

Во второй главе представленной выпускной квалификационной работы магистра была проделана работа по выбору и анализу всех основных компонент, которые будут использоваться в работе платформы или уже используются.

На основе проведенного анализа были выбраны и развернуты три архитектуры и одна находится в разработке. Docker Swarm – подходит для начала знакомства с данной предметной областью. Kubernetes + CephFS – рабочее решение, но имеет ряд недостатков. OpenShift в облаке AWS – дорого. Openshift на голом железе – в разработке, на бумаге имеет ряд преимуществ перед Kubernetes + CephFS. Самое большое преимущество – автоматическое обновление.

# Технологическая часть

В технологической части представленной выпускной квалификационной работы магистра показан способ развертывания, двух вариантов платформ. Платформа на преднастроенном железе на основе Kubernetes и OpenShift в облаке AWS. В работе использовались следующие инструменты:

* операционная система Ubuntu16.04, Ubuntu 18.04;
* текстовый редактор nano;
* текстовый редактор sublimeText;
* система контроля версий Git;
* фреймворк Kubernetes;
* фреймворк OpenShift;
* OpenShift Installer (можно запустить и на Mac OS X);
* установщик kubeadm;
* командная строка Linux;
* Docker – среда выполнения приложений;
* операционная система RHEL
* AWS облако

## Процесс развертывания платформы на базе Kubernetes

Для данного этапа вам необходимы:

* операционная система Ubuntu16.04, Ubuntu 18.04;
* система контроля версий Git;
* командная строка Linux;

Проделайте следующие шаги:

1. Установите все необходимые библиотеки следующими командами;

apt-get update && apt-get install -y apt-transport-https curl

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -

cat <<EOF >/etc/apt/sources.list.d/kubernetes.list

deb https://apt.kubernetes.io/ kubernetes-xenial main

EOF

apt-get update

apt-get install -y kubelet kubeadm kubectl

apt-mark hold kubelet kubeadm kubectl

1. Настройте домашнюю директроию;

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

1. Проинициализируйте мастер

kubeadm init --pod-network-cidr=192.168.0.0/16

kubectl apply -f https://docs.projectcalico.org/v3.3/getting-started/kubernetes/installation/hosted/rbac-kdd.yaml

kubectl apply -f https://docs.projectcalico.org/v3.3/getting-started/kubernetes/installation/hosted/kubernetes-datastore/calico-networking/1.7/calico.yaml

1. Разрешите запускать на мастере контейнеры

kubectl taint nodes --all node-role.kubernetes.io/master-

1. Установите в кластер Apache Spark

https://github.com/DmitryZagr/docker-spark-hive-zeppelin/tree/master/kubernetes

## Процесс развертывания OpenShift платформы в облаке AWS

Настройка DNS. Вы можете пропустить этот шаг, если используете существующий домен и регистратора. Вы переместите авторитетный DNS на Route53 или отправите запрос на делегирование для субдомена на более позднем этапе. Route53 может также приобрести домены для вас и действовать в качестве регистратора. Если вы разрешите Route53 приобрести новый домен для вас, вы можете пропустить оставшуюся часть этих шагов (домен создан, и размещенная зона создана для вас правильно)!

Получение домена. Вы можете пропустить этот шаг, если используете существующий домен и регистратора. Вы переместите авторитетный DNS на Route53 или отправите запрос на делегирование для субдомена на более позднем этапе. Route53 может также приобрести домены для вас и действовать в качестве регистратора. Если вы разрешите Route53 приобрести новый домен для вас, вы можете пропустить оставшуюся часть этих шагов (домен создан, и размещенная зона создана для вас правильно)!

Создание общедоступной зоны. Независимо от того, используете ли вы корневой домен или поддомен, вы должны создать публичную размещенную зону. Чтобы использовать корневой домен, вы должны создать размещенную зону со значением openshiftcorp.com. Чтобы использовать поддомен, вы должны создать размещенную зону со значением clusters.openshiftcorp.com. (Используйте соответствующие значения домена для вашей ситуации.)

Настройка учетной записи. Независимо от того, используете ли вы корневой домен или поддомен, вы должны создать публичную размещенную зону. Чтобы использовать корневой домен, вы должны создать размещенную зону со значением openshiftcorp.com. Чтобы использовать поддомен, вы должны создать размещенную зону со значением clusters.openshiftcorp.com. (Используйте соответствующие значения домена для вашей ситуации.)

Установка кластера. Воспользуйтесь новым инструментом от Red Hat под названием установщик кластера и выполните следующие команды:

* Создание конфигурации

[~]$ openshift-install-linux-amd64 create install-config

? SSH Public Key /home/zagd/.ssh/id\_rsa.pub

? Platform aws

? Region us-east-2

? Base Domain example.com

? Cluster Name test

? Pull Secret [? for help]

* Создание кластера

[~]$ openshift-install-linux-amd64 create cluster

INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...

INFO API v1.11.0+85a0623 up

INFO Waiting up to 30m0s for the bootstrap-complete event...

INFO Destroying the bootstrap resources...

INTO Waiting up to 30m0s for the cluster at https://api.test.example.com:6443 to initialize...

INFO Waiting up to 10m0s for the openshift-console route to be created...

INFO Install complete!

INFO Run 'export KUBECONFIG=/home/user/auth/kubeconfig' to manage the cluster with 'oc', the OpenShift CLI.

INFO The cluster is ready when 'oc login -u kubeadmin -p XXXX' succeeds (wait a few minutes).

INFO Access the OpenShift web-console here: https://console-openshift-console.apps.test.example.com

INFO Login to the console with user: kubeadmin, password

## Тестирование платформы

Тестирование платформы было произведено при следующих входных параметрах:

* Операционная система Ubuntu 18.04;
* Развертывание сервиса Apache Spark;

Перед началом тестирования был разработан план тестирования, представленный в таблице 9 и тестовые случаи, представленные в таблице 10.

После этого было проведено тестирование и получены результаты, которые представлены в таблице 11.

Таблица – План тестирования

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Объект тестирования** | **Уровень тестирования** | **Тестируемая ошибка** | **Виды деятельности** | **Частота тестирования** | **Ответственный исполнитель** |
| Платформа | Административный | Не работоспособность платформы | Выбор тестовых случаев | До сдачи варианта платформы заказчику | Тестировщик |

Таблица – Тестовые случаи

| **№** | **Тестируемые части системы** | **Входные данные** | **Ожидаемые результаты** | **Полученные результаты** |
| --- | --- | --- | --- | --- |
| 1 | Вся Платформа | Платформа запускается в обычном режиме | Корректная инициализация Платформы |  |
| 2 | Компонент добавления нового сервиса | конфигурация | Система корректно читает конфигурацию, инициализирует и запускает новый сервис |  |

Продолжение таблицы 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Тестируемые части системы** | **Входные данные** | **Ожидаемые результаты** | **Полученные результаты** |
| 3 | Компонент чтения конфигурации | Ошибочная конфигурация сервиса | Платформа сообщает об ошибке и корректно продолжает свою работу |  |
| 4 | Компонент ввода команды | Команда сисадмина | Платформа получает команду и выдает результат в консоль |  |
| 5 | Система мониторинга | Команда показать графики | Платформа распознает команду и открывает интерфейс системы мониторинга |  |

Таблица – Результаты тестирования

| **№** | **Тестируемые части системы** | **Входные данные** | **Ожидаемые результаты** | **Полученные результаты** | **Заключение** |
| --- | --- | --- | --- | --- | --- |
| 1 | Вся Платформа | Платформа запускается в обычном режиме | Корректная инициализация Платформы | Корректная инициализация Платформы | Штатная ситуация |
| 2 | Компонент добавления нового сервиса | конфигурация | Система корректно читает конфигурацию, инициализирует и запускает новый сервис | Система корректно читает конфигурацию, инициализирует и запускает новый сервис | Штатная ситуация |
| 3 | Компонент чтения конфигурации | Ошибочная конфигурация сервиса | Платформа сообщает об ошибке и корректно продолжает свою работу | Платформа сообщает об ошибке и корректно продолжает свою работу | Штатная ситуация |

Продолжение таблицы 4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Тестируемые части системы** | **Входные данные** | **Ожидаемые результаты** | **Полученные результаты** | **Заключение** |
| 4 | Компонент ввода команды | Команда сисадмина | Платформа получает команду и выдает результат в консоль | Платформа получает команду и выдает результат в консоль | Штатная ситуация |
| 5 | Система мониторинга | Команда показать графики | Платформа распознает команду и открывает интерфейс системы мониторинга | Система мониторинга | Штатная ситуация |

## Выводы к главе 3

В третьей главе представленной выпускной квалификационной работы магистра описаны основные используемые способы, для развертывания платформы как на основе чистого Kubernetes, так и OpenShift.

Разработанные конфигурационные файлы успешно применяются к платформе, о чем свидетельствуют результаты тестирования.

Также были разработаны: план тестирования, тестовые случаи и было проведено тестирование. Проведенное тестирование подтверждает работоспособность платформы.

ЗАКЛЮЧЕНИЕ

В первой главе представленной выпускной квалификационной работы магистра был проведен анализ заинтересованных сторон платформы анализа больших данных, были выявлены функции этой Системы и ее показатели качества.

Во второй главе была проделана работа по созданию архитектуры разработанной Платформы. Было развернуто несколько прототипов и еще один находится в работе.

В третьей главе представленной ВКРМ была проведена работа развертыванию Платформы в разных средах и с помощью разных фреймворках.

В результате была решена поставленная цель ВКРМ – получение кластерной платформы анализа больших данных на основе Kubernetes с использованием Apache Spark.

В процессе работы над представленной ВКРМ были подготовлены и опубликованы статьи [4] и [17].

В дальнейшем планируется дополнить разработанную Систему модулем голосового управления.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1) Словарь терминов: [Электронный ресурс]. URL: https://ru.bmstu.wiki/ (дата обращения 03.10.2018)

2) Технологии Big Data: как использовать большие данные в маркетинге: [Электронный ресурс]. URL: https://www.uplab.ru/blog/big-data-technologies/ (дата обращения: 03.10.2018)

3) Apache Spark: [Электронный ресурс]. URL: https://spark.apache.org/ (дата обращения: 03.10.2018).

4) Загребаев Д. К. Развертывание OpenShift кластера в облаке AWS. Проблемы прикладной и школьной информатики (дата обращения 05.10.2018).

5) CephFS: [Электронный ресурс]. URL: https://ceph.com/ (дата обращения: 09.10.2018).

6) Bare metall: [Электронный ресурс]. URL: https://www.ibm.com/ru-ru/cloud/bare-metal-servers (дата обращения: 19.10.2018).

7) Сетевая загрузка: [Электронный ресурс]. URL: https://www.opennet.ru/docs/HOWTO-RU/Diskless-HOWTO/x606.html (дата обращения: 19.03.2019).

8) Docker Swarm: [Электронный ресурс]. URL: https://docs.docker.com/engine/swarm/ (дата обращения: 13.12.2018).

9) Kubernetes: [Электронный ресурс]. URL: https://kubernetes.io/ (дата обращения: 26.12.2018).

10) OpenShift: [Электронный ресурс]. URL: https://www.openshift.com/ (дата обращения: 19.03.2019).

11) HDFS: [Электронный ресурс]. URL: https://hadoop.apache.org/ (дата обращения: 19.12.2018).

12) GlusterFS: [Электронный ресурс]. URL: https://www.gluster.org/ (дата обращения 10.01.2019).

13) Grafana: [Электронный ресурс]. URL: Grafana (дата обращения: 25.012.2018).

14) Prometheus: [Электронный ресурс]. URL: https://prometheus.io/ (дата обращения: 26.12.2018).

15) Influx DB: [Электронный ресурс]. URL: https://www.influxdata.com/ (дата обращения: 26.12.2018).

16) CNCF: [Электронный ресурс]. URL: https://www.cncf.io/ (дата обращения: 27.12.2018).

17) Загребаев Д.К. Мониторинг кластера анализа больших данных Apache Spark на основе Kubernetes: [Электронный ресурс]. // Достижения науки и образования №5 (46), 2019 год. URL: https://SCIENTIFICTEXT.RU (дата обращения: 26.11.2018).

Приложение А

**В графическую часть квалификационной работы входят:**

В графическую часть квалификационной работы входят:

* диаграмма Заинтересованные стороны (рисунок 38);
* диаграмма Представители заинтересованных сторон (рисунок 39);
* диаграмма Цели заинтересованных сторон (рисунок 40);
* диаграмма Рабочие сценарии (рисунок 41);
* диаграмма Базовая архитектура кластера (рисунок 42);
* диаграмма Архитектура Kubernetes кластера (рисунок 43);
* диаграмма Интеграция Kubernetes и Ceph (рисунок 44);
* диаграмма Развёртывание в AWS (рисунок 45).

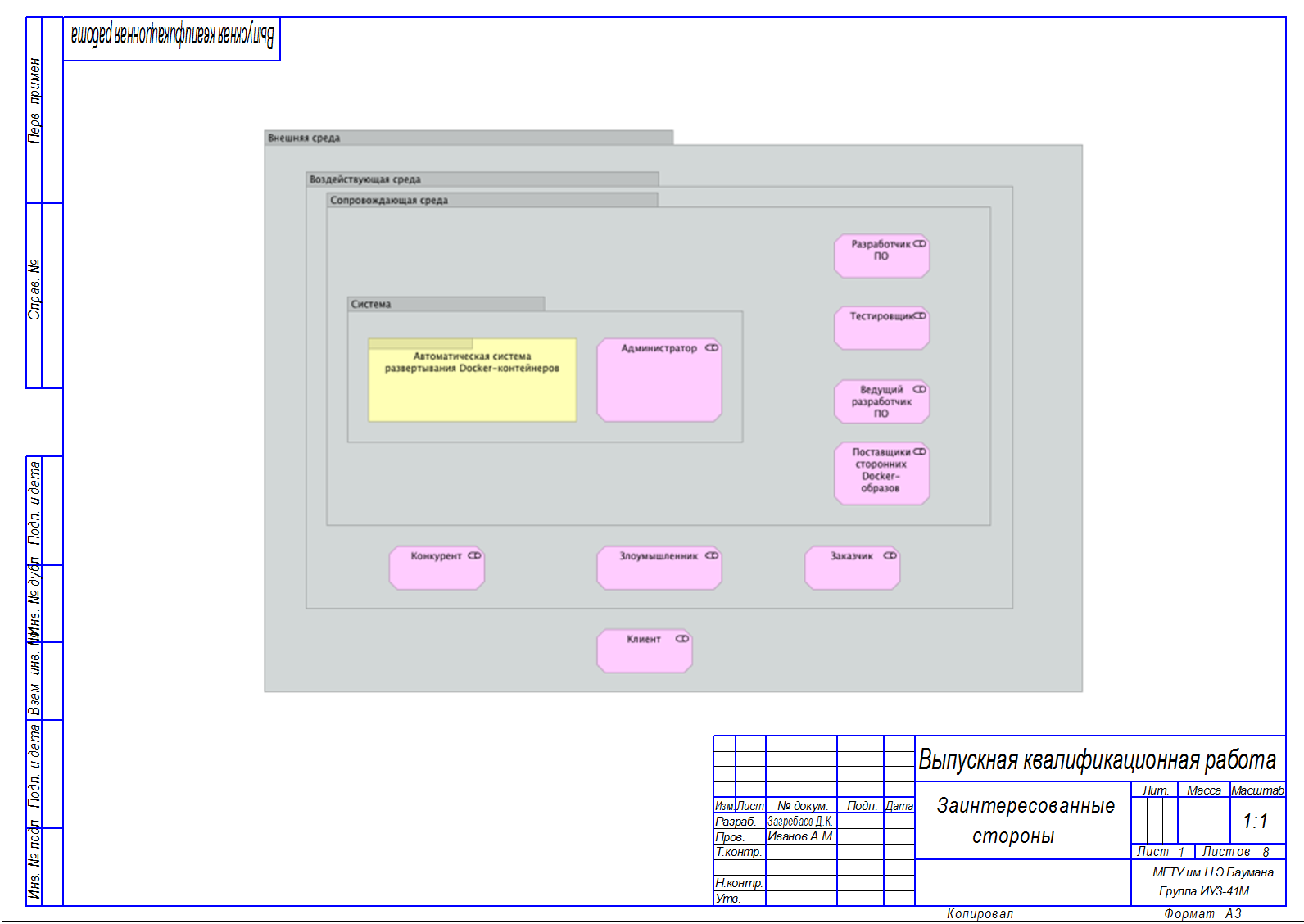


Рисунок 38 – Заинтересованные стороны

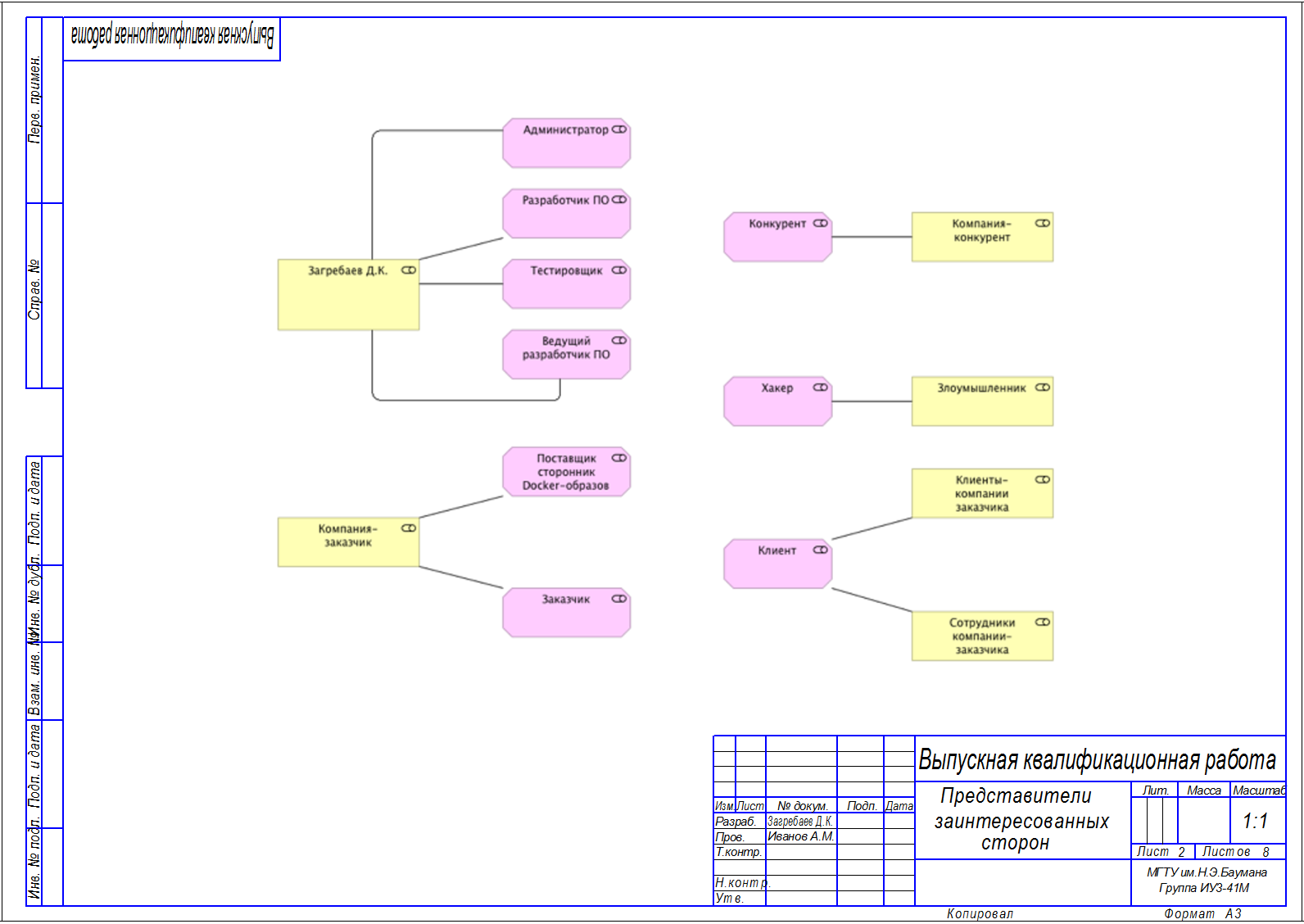


Рисунок 39 – Представители заинтересованных сторон

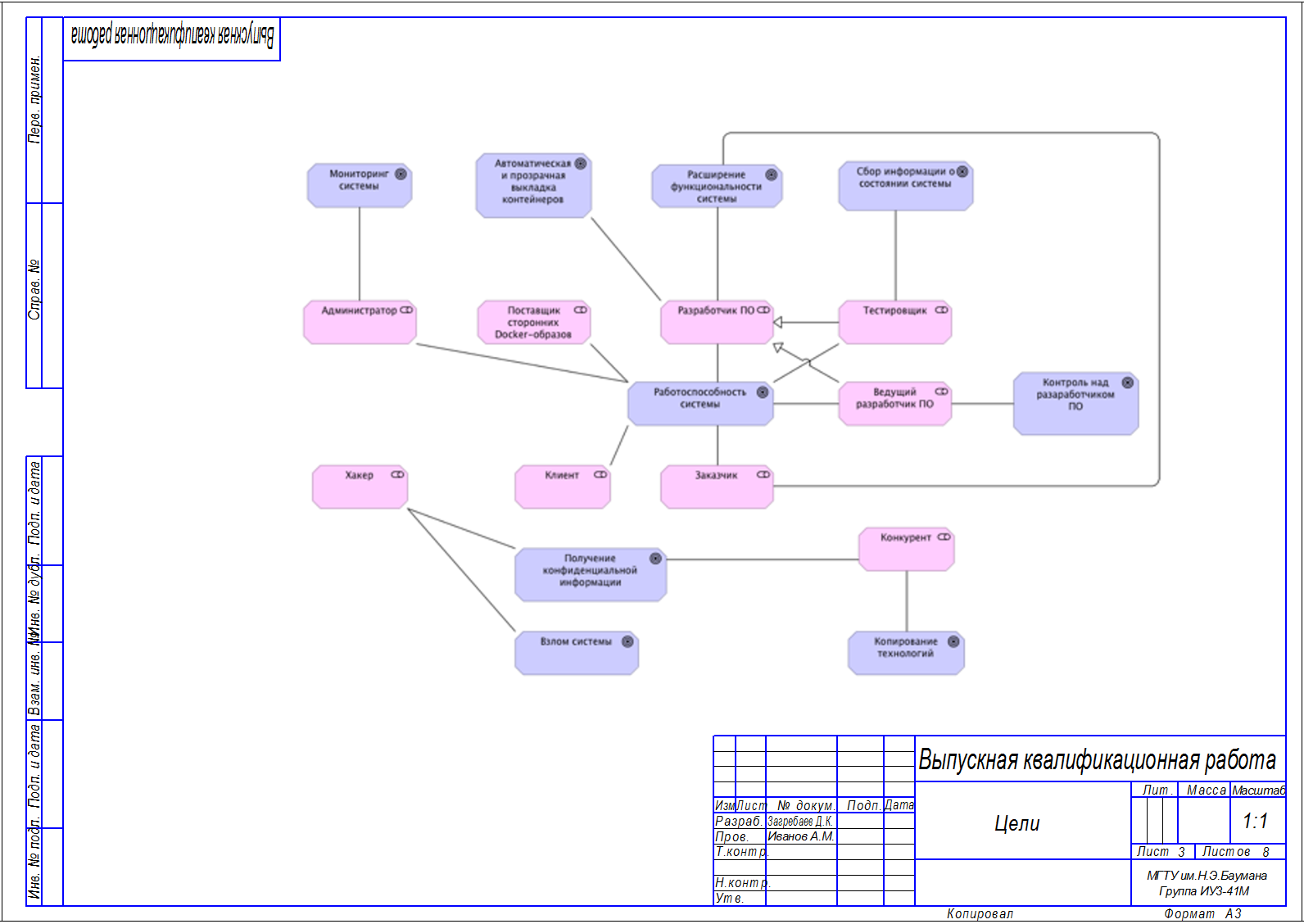


Рисунок 40 – Цели заинтересованных сторон

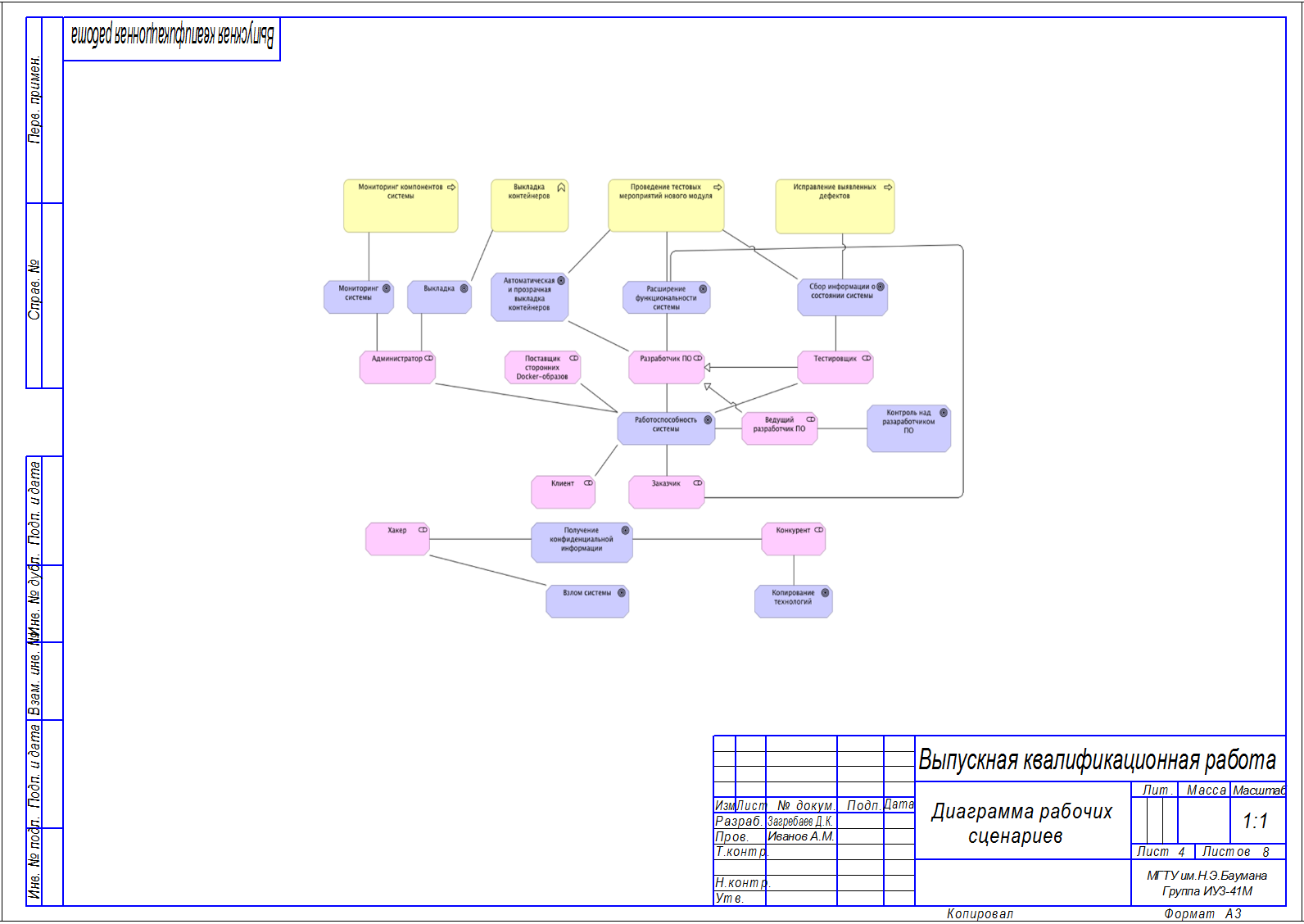


Рисунок 41 – Диаграмма рабочих сценариев

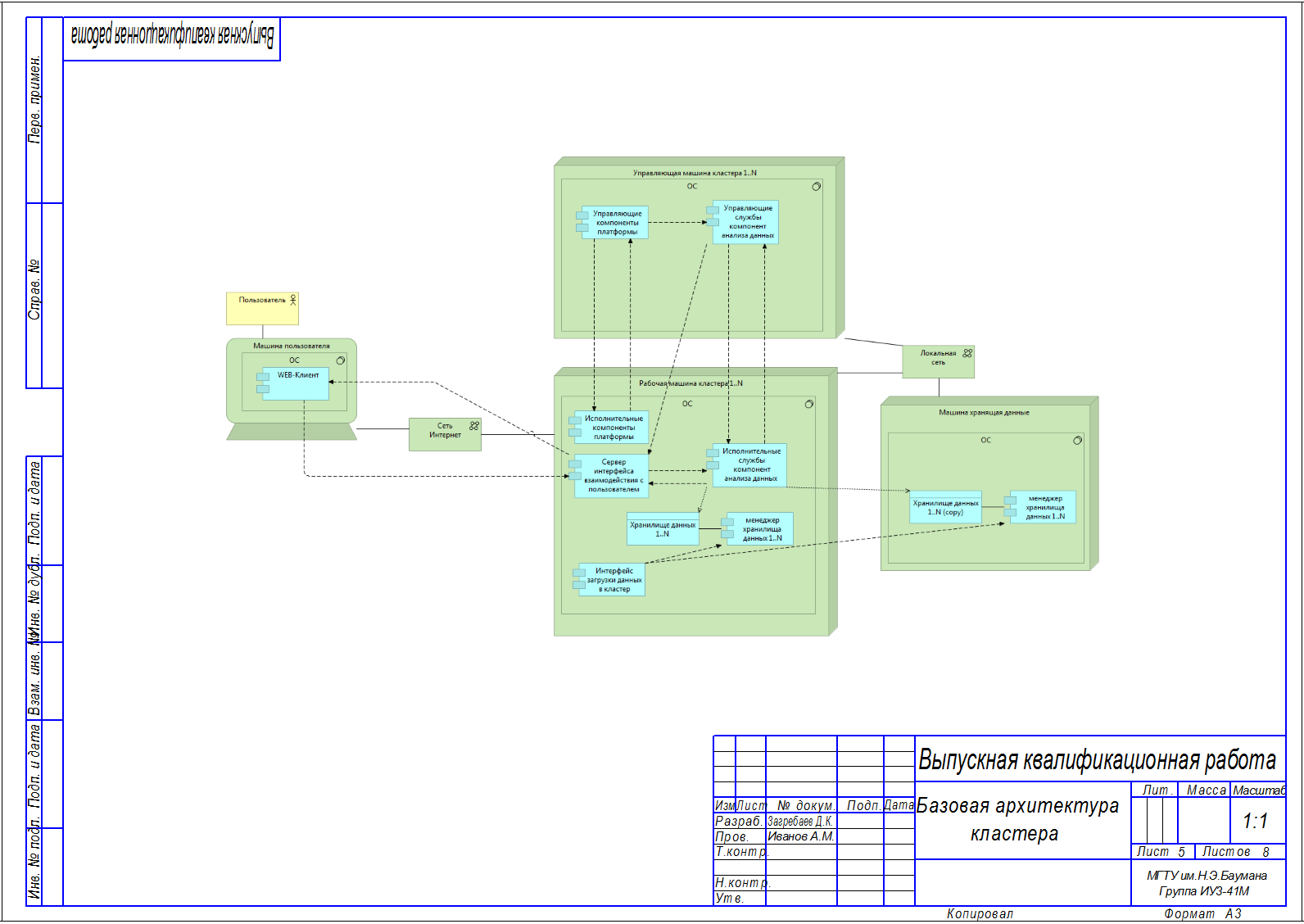


Рисунок 42 – Базовая архитектура кластера

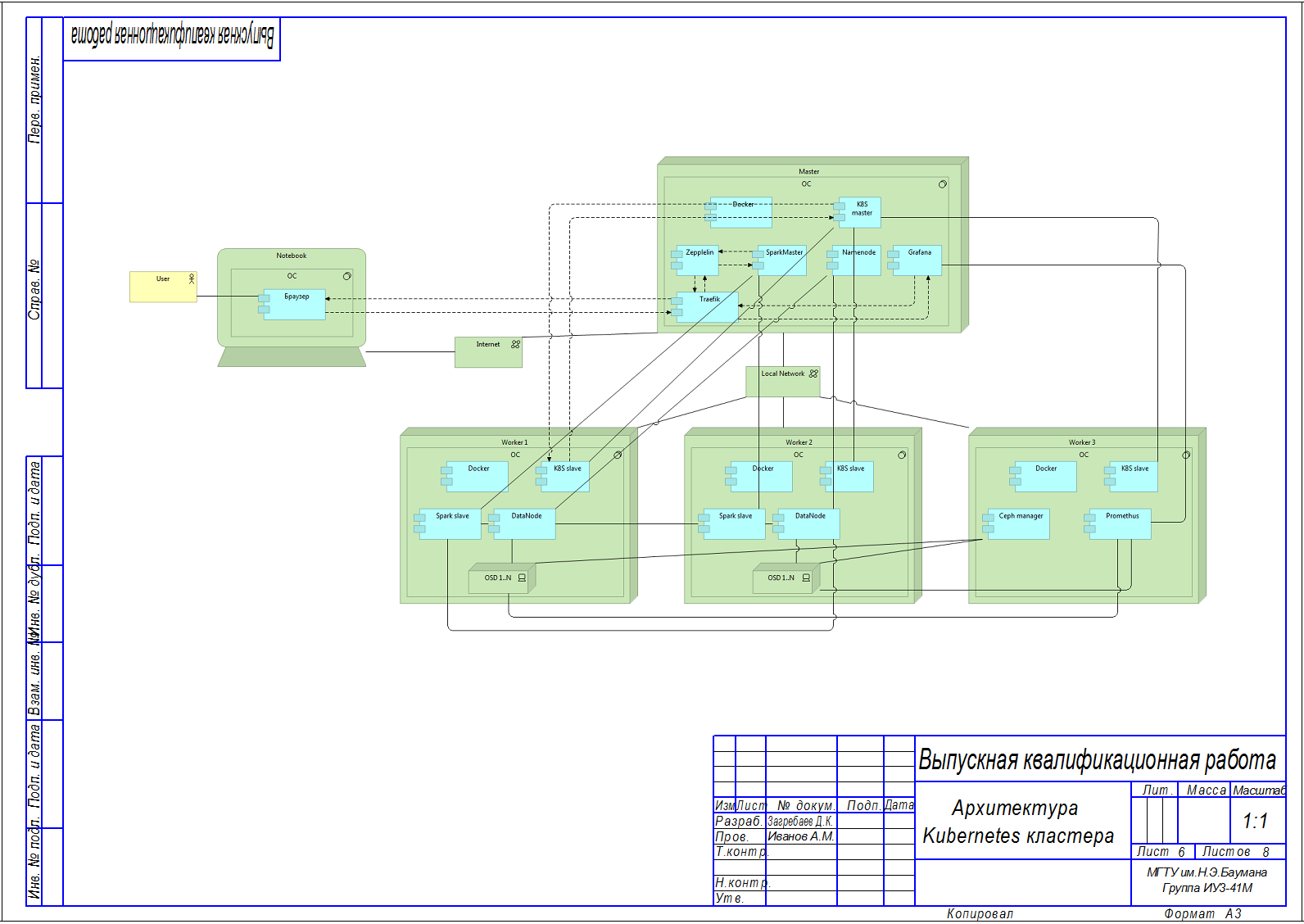


Рисунок 43 – Архитектура Kubernetes кластера

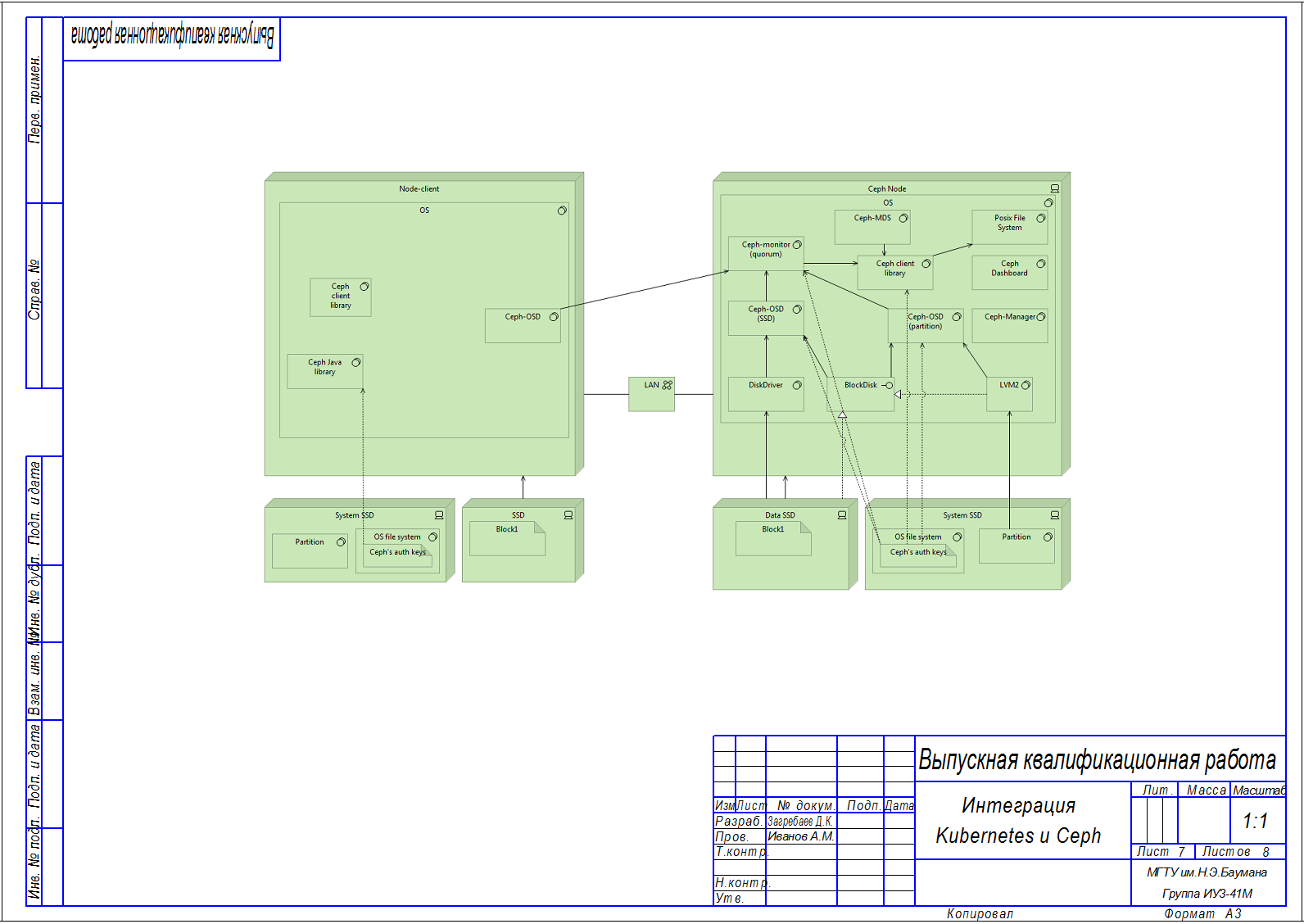


Рисунок 44 – Интеграция Kubernetes и Ceph

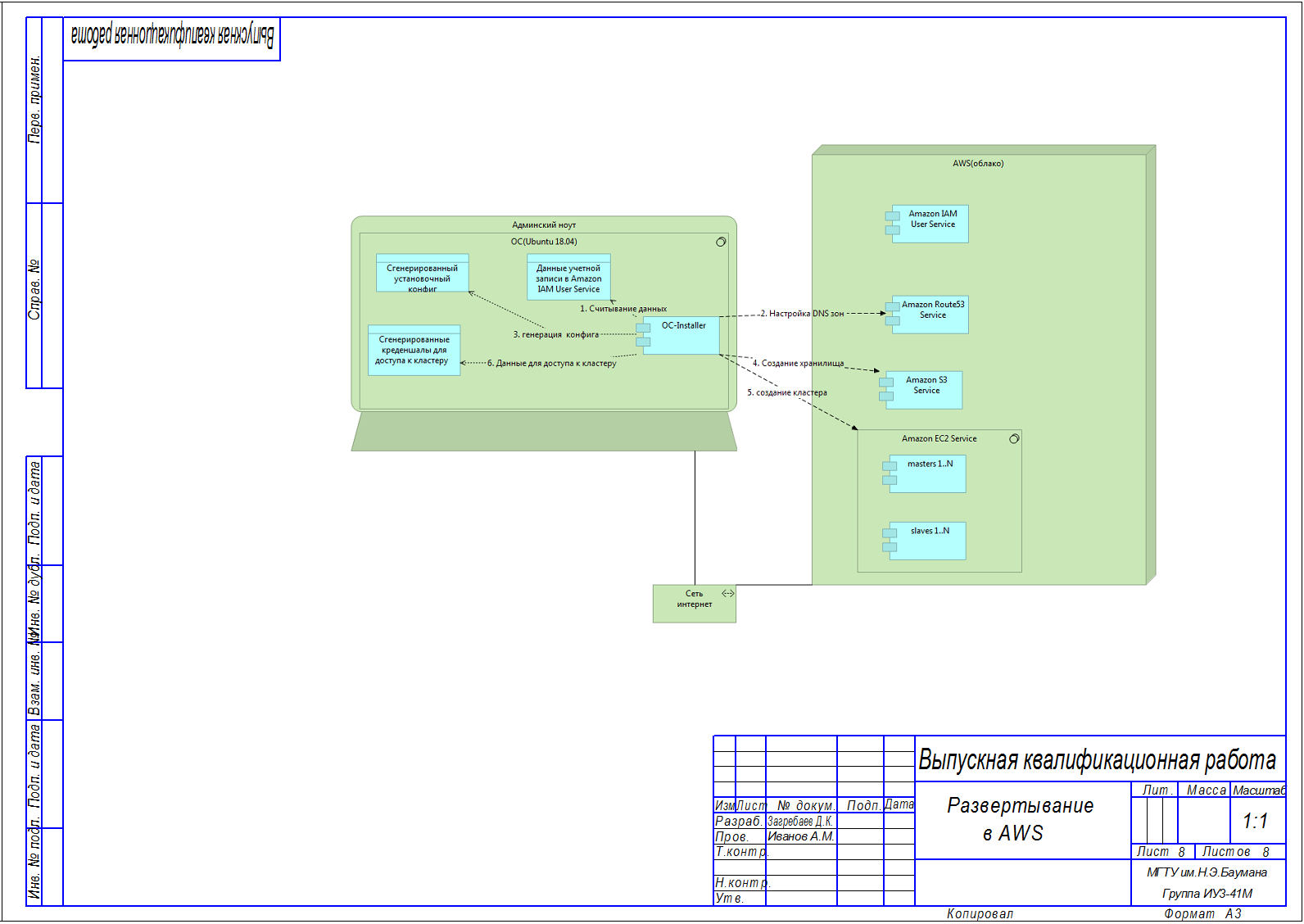


Рисунок 45 – Развертывание в AWS

ПРИЛОЖЕНИЕ 1  
Техническое задание

* 1. Общие сведения

В представленной выпускной квалификационной работе разрабатывается платформа анализа больших данных на использовании Kubernetes с использованием Apache Spark.

Дополнительное ограничения:

* платформа на основе Kubernetes;
* факто-ориентированное описание архитектуры платформы

Срок выполнения работы – 12 недель, с 7 февраля 2019 года.

* 1. Назначение и цели создания платформы

Платформа анализа больших данных на основе Kubernetes представляет собой легко развертываемую платформу на физических или облачных серверах. Платформа должна поддерживать распределенное развертывание.

В результате создания платформы должна быть достигнута цель развертывания кластера на нескольких физических или виртуальных машинах.

* 1. Требования к платформе

В данной платформе требуется производить успешное развертывания и хранение данных на нескольких машинах.

Основные показатели качества, которые предъявляются к системе представлены в таблицах 16-21.

Таблица 16 – Показатели производительности системы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Источник стимула** | **Стимул** | **Условия** | **Артефакт** | **Реакция** | **Количественная мера** |
| Администратор | Обновление сервисов | Нормальные условия | Платформа | Последовательное обновление сервисов в кластере, относящихся к одному приложению | Менее 1 секунды |
| Администратор | Добавление нового сервиса в кластер | Нормальные условия | Платформа | Создание экземпляров сервисов | Реакция платформы менее 1 секунды |
| Платформа | Детектирование сбоя приложения | Нормальные условия | Платформа | Платформа пытается перезапустить приложение | Менее 30 секунд |
| Администратор | Загрузка данных | Нормальные условия | Платформа | Платформа начинает загрузку данных в распределенную файловую систему | Менее 1 секунд |

Таблица 17 – Показатели модифицируемости платформы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Источник стимула** | **Стимул** | **Условия** | **Артефакт** | **Реакция** | **Количественная мера** |
| Разработчик платформы | Написание плагинов для платформы | В период разработки | Исходный код плагина платформы | Не потребовалось существенных изменений и доработок ранее разработанных частей платформы | В течение часа |
| Разработчик  платформы | Подключение плагинов к платформе | В период разработки | Исходный код плагина платформы | Не потребовалось существенных изменений и доработок ранее разработанных частей платформы | В течение часа |

Таблица 18 – Показатели готовности платформы

| **Источник стимула** | **Стимул** | **Условия** | **Артефакт** | **Реакция** | **Количественная мера** |
| --- | --- | --- | --- | --- | --- |
| Платформа | При развёртывании приложения произошла ошибка | Нормальные условия | Платформа | Журналирование ошибки, выдача ошибки | Платформа продолжает работать |
| Платформа | Длительное бездействие | Нормальные условия | Платформа | Платформа продолжает работу | Платформа работает в штатном режиме |
| Платформа | Аварийная ситуация | Нормальные условия | Платформа | Выдача сообщения об ошибке работы платформы, перезапуск компонента платформы, журналирование ошибки | Платформа работает в штатном режиме |

Таблица 19 – Показатели контролепригодности системы

| **Источник стимула** | **Стимул** | **Условия** | **Артефакт** | **Реакция** | **Количественная мера** |
| --- | --- | --- | --- | --- | --- |
| Тестировщик платформы | Тестирование функций платформы | В момент тестирования платформы | Платформа | Выполнение команд, пришедших от тестировщика | Задействованы весь необходимый функционал платформы |
| Тестировщик платформы | Нахождение ошибок в работе платформы | В момент тестирования платформы | Платформа | Платформа работает без ошибок | Задействовано не менее 90% функциональных возможностей Платформы |
| Тестировщик платформы | Контроль соблюдения принятых соглашений | В момент тестирования платформы | Платформа | Конфигурация платформы соответствует принятым соглашениям | 100% конфигураций платформы принятым соглашениям |
| Тестировщик платформы | Добавление новых модулей | В момент тестирования системы | Платформы | Новый модуль добавляется в платформу без возникновения ошибок | 100% модулей, удовлетворяющих интерфейсным требованиям |

Таблица 20 – Показатели практичности системы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Источник стимула** | **Стимул** | **Условия** | **Артефакт** | **Реакция** | **Количественная мера** |
| Конечный пользователь | Использование возможности платформы | В момент пользования платформы | Платформа | Повторное использование конфигураций | 80% объёма конфигурации можно использовать повторно |
| Конечный пользователь | Легкое и понятное пользование платформой | В момент пользования платформы | Платформа | Платформа имеет понятный и хорошо задокументированный интерфейс | Интерфейс взаимодействия с платформой понятен для конечного пользователя |
| Конечный пользователь | Сведение к минимуму количества ошибок при использовании платформы | В момент пользования платформы | Платформа | Саморазвертываемость платформы | Функционирование платформы не требует взаимодействия с конечным пользователем |

Таблица 21 – Атрибуты безопасности

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Источник стимула** | **Стимул** | **Условия** | **Артефакт** | **Реакция** | **Количественная мера** |
| Взломщик | Кража данных пользователя | В момент использования | Данные в платформе | Система проверяет доступность данных пользователю, основываясь на RBAC | Данные пользователя находятся под защитой от неавторизованного доступа |
| Взломщик | Получение доступа к управлению платформой | В момент использования | База знаний | Система проверяет доступность данных и механизмов управления для пользователя, основываясь на RBAC | Данные пользователя находятся и элементы управления платформой находятся под защитой от неавторизованного доступа |

* 1. Состав и содержание работ по созданию платформы

При создании платформы необходимо провести следующие работы:

* выявить заинтересованные стороны;
* выявить функции платформы
* выявить показатели качества платформы;
* разработать архитектуру платформы;
* выбрать базовую архитектуру платформы;
* реализовать несколько прототипов платформы;
* провести отладку платформы;
* провести тестирование платформы.

Платформа должна обладать следующей функциональностью:

* Платформа автоматический развертывается с минимальным участие человека;
* Платформа имеет систему мониторинга;
* Платформа позволяет надёжно хранить большие объемы данных;
* Платформа позволяет разворачивать в себе в виде Docker контейнеров приложения;
* Платформа поддерживает описание конфигураций;
* Платформа поддерживает ввод текста с клавиатуры;
  1. Порядок контроля и приемки платформы

Для контроля работоспособности платформы должен быть проведен экспериментальный запуск платформы, подтверждающий работу платформы.

* 1. Требования к документированию

Расчетно-пояснительная записка

* 1. Источники разработки

Задание на ВКР, НИ