

Санкт-Петербургский государственный университет  
Факультет математики и компьютерных наук

Дисциплина: Виртуализация и облачные технологии

## **Проектная работа**

**Выполнил студент:**  
**Андрей Сергеевич Хорохорин**

**Преподаватель:**  
**Егор Дмитриевич Кривоносов**

Санкт-Петербург

2024 г.

# Содержание

Постановка задачи.....	3
Предлагаемое решение.....	3
Предполагаемая реализация в “облаке” .....	4
Выбор облачного провайдера.....	5
Yandex cloud.....	5
Определение конкретных сервисов.....	5
Наличие пробного периода тарификации.....	5
Анализ стоимости.....	5
Субъективная оценка удобства.....	6
VK cloud.....	6
Определение конкретных сервисов.....	6
Наличие пробного периода тарификации.....	6
Анализ стоимости.....	7
Субъективная оценка удобства.....	7
Cloud.ru.....	7
Определение конкретных сервисов.....	7
Наличие пробного периода тарификации.....	7
Анализ стоимости.....	7
Субъективная оценка удобства.....	7
Вывод.....	8
Подробная спецификация продукта.....	9
Предполагаемые роли клиентов в продукте.....	9
Администратор сервиса.....	9
Хост сервиса.....	9
Пользователь сервиса.....	9
Подробная схема функциональности ролей.....	10
Предполагаемая нагрузка.....	13
Планируемая реализация спецификации.....	14
Архитектура сервиса.....	14
С точки зрения сети и программных компонентов.....	14
Выбор технологий для программных компонентов.....	15
Система контроля прав доступа.....	15
Описание RESTful API.....	15
Схема реляционной базы данных.....	16
Ограничения реализации.....	16
Возможности по масштабированию.....	16
Отклонения от плана во время реализации.....	17
Демонстрация работы.....	18
Клиентское приложение.....	18
Облачная инфраструктура.....	19
Серверное приложение.....	21
Вывод.....	22

## Постановка задачи

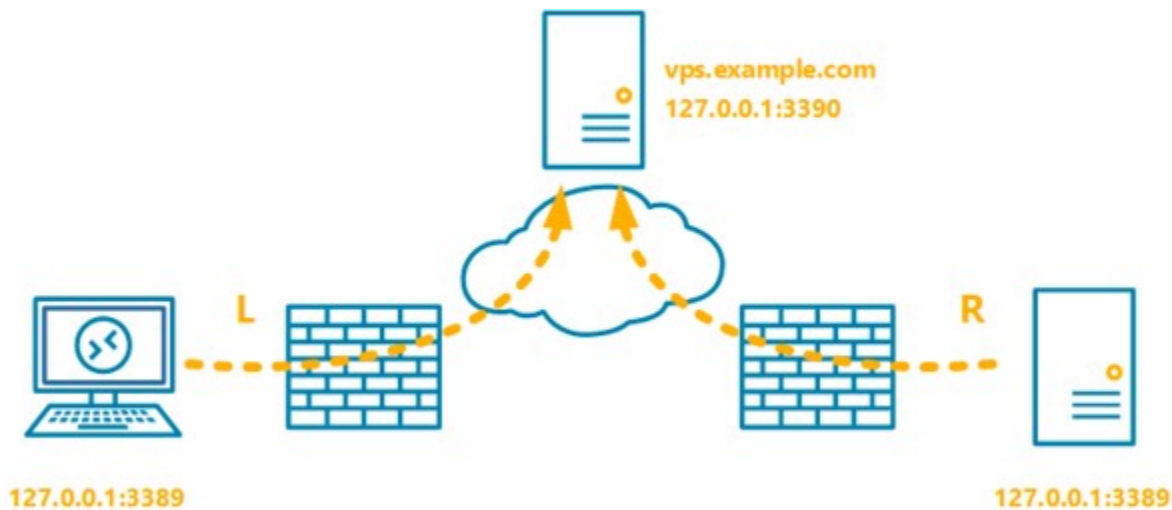
Современные компьютерные сети устроены достаточно сложно, из-за чего выполнение некоторых простых операций затруднено. Например одна из главных задач организации соединения между двумя хостами, сейчас требует некоторого ряда условий, наиболее труднодоступными из которых являются статический ip и отсутствие(или правильная настройка) вышестоящего NAT.

## Предлагаемое решение

К счастью есть достаточно простой способ организовать соединение лишь зная на соединяемых хостах (которые далее будут обозначены как A и B) сервера-посредника, через которого можно организовать соединение. Смысл заключается в организации двух туннелей:

- хост A – сервер-посредник
- хост B – сервер-посредник

и настройке сервера-посредника для связки этих двух туннелей.



Более того, сервер-посредник можно использовать для организации и более сложного сценария. Пусть у нас есть хоста-сервера A, B на обоих из которых предоставляется некоторый сервис, оба эти хоста находятся в неблагоприятных условиях: они за NAT и не имеют белого IP, что не даёт подключиться к ним напрямую. И есть некоторое количество

клиентов, которые хотят воспользоваться сервисом. Тогда сервер-посредник может заниматься динамическим балансом создания новых подключений от клиентов между хостами А и В.

Всё вышеописанное планируется предоставить в виде консольной утилиты, при помощи которой можно будет задавать логику приёма соединений для пользователей, которые хотят организовать некоторый сервис, и при помощи которой можно подключиться к развёрнутым таким образом сервисам.

## **Предполагаемая реализация в “облаке”**

Для реализации в облаке предполагается использовать следующие компоненты:

- Виртуальная машина, которая будет использоваться и как прокси-хост и как хост для серверной части консольного приложения.
- База данных активных хостов, готовые принять соединение и соответствующие им хендл и символьное имя.
- Ещё какой-либо сервис, упрощающий организацию сервиса и создание API. Например сервис управления ключами пользователя, который будет хранить ключи пользователей и предоставляющий API для консольной утилиты, позволяющий создать ключ. [Пример](#) сервиса в Яндекс. Облаке.

# Выбор облачного провайдера

К рассмотрению были выбраны следующие облачные провайдеры:

- [Yandex cloud](#)
- [VK cloud](#)
- [Cloud.ru](#)

Они были выбраны как самые известные провайдеры в РФ, чтобы избежать вероятность санкций от иностранных сервисов или отсутствия необходимых сервисов.

## Yandex cloud

### Определение конкретных сервисов

В соответствии с предполагаемой реализацией мною были выбраны следующие сервисы в Yandex cloud. (названия ниже соответствуют сервисам в Yandex cloud)

- [Compute Cloud](#) в качестве виртуальной машины
- [Yandex Managed Service for YDB](#) в качестве базы данных. Выбрана в целях экономии, в нашем на ней случае можно сэкономить, тк интенсивность обращения к БД незначительна. Подробнее см. «анализ стоимости»
- [Yandex API Gateway](#) для упрощения создания API, с которым будет взаимодействовать клиентское консольное приложение.
- Yandex Monitoring для мониторинга работы приложения.

### Наличие пробного периода тарификации

Платформа предлагает депозит в размере 4000 рублей на знакомство с платформой, которые можно тратить первые два месяца после начала работы. Однако на эту сумму накладывается ряд ограничений главным из которых является то, что на услуги Compute Cloud от этой суммы может быть потрачено не более 1000 рублей.

Более подробно с условиями пробного периода можно ознакомиться [по ссылке](#).

### Анализ стоимости

Для [Yandex Managed Service for YDB](#) каждый месяц первые миллион запросов не тарифицируются, при условии что база данных хранит менее одного гигабайта. В целом сложно себе представить что даже при активном использовании приложения этот лимит исчерпается.

[Yandex API Gateway](#) каждый месяц не тарифицируются первые 100 000 запросов к API-шлюзам.

Основную стоимость решения составит виртуальная машина. Её исходящий трафик свыше 100 гб в месяц будет стоить по 1,53 рубля за гигабайт. Не смотря на то, что в

процессе разработки данный трафик сложно будет израсходовать, то при дальнейшем использовании сервиса по назначению количеству проходящего исходящего трафика может быть весьма большим.

Если брать самую дешёвую виртуальную машину, то размера депозита хватит чуть менее чем на месяц. Ниже приведена предполагаемая конфигурация виртуальной машины:

Параметр	Значение
Платформа	Intel Cascade Lake
Гарантированная доля vCPU	5% 20% 50% 100%
Количество vCPU	2
Объём RAM	1 GB
Прерываемая VM	<input type="checkbox"/>
Публичный IP-адрес	<input checked="" type="checkbox"/>
Исходящий трафик	1 GB
Диск — 1 (Загрузочный)	network-ssd
Размер диска	5 GB

Компонент	Цена
Compute Cloud 1	1161,15 ₽
Виртуальная машина Compute Cloud 1	1161,15 ₽
Исходящий трафик в интернет	0,00 ₽
<b>Итого</b> (в месяц)	<b>1161,15 ₽</b>

### Субъективная оценка удобства

Во время анализа стоимости и подборки конкретных сервисов я отметил, что есть множество примеров развертки решений, документация достаточно полная.

## VK cloud

### Определение конкретных сервисов

В соответствии с предполагаемой реализацией мною были выбраны следующие сервисы в VK cloud.

- [vps/vds](#)
- [DaaS](#) PostgreSQL
- у vk не оказалось сервиса для организации API endpoint. Поэтому в качестве третьего сервиса при использовании VK cloud можно взять сервис [мониторинга](#)

### Наличие пробного периода тарификации

Платформа предлагает депозит в размере 3000 рублей на знакомство с платформой, которые выдаются при регистрации на корпоративный email.

Более подробно с условиями пробного периода можно ознакомиться [по ссылке](#).

### Анализ стоимости

Самая дешёвая база данных будет стоить чуть менее 4000 рублей в месяц.

Трафик виртуальной машины не тарифицируется. Сама виртуальная машина будет стоить 983 рубля в месяц, при этом её конфигурация чуть более мощная, чем в Yandex Cloud.

Cloud monitoring предоставляется бесплатно

В результате получается, что учебный проект можно будет хостить чуть более двух недель без внесения оплаты, что хуже чем предложение от Yandex Cloud, зато нет тарификации трафика, что при большой популярности приложения может делать предложение от VK cloud выгодней.

The screenshot shows a configuration interface for a cloud service. At the top, a blue button labeled 'basic-1-1-10' is visible. Below it, the price is listed as '983 ₽ / месяц' and '0,02 ₽ / мин'. The specifications section includes: vCPU (x1), RAM (1 ГБ), and Объем (10 ГБ). Under 'Поколение процессора', 'Intel Cascade Lake' is selected over 'Intel Ice Lake'. For 'Тип диска', 'HDD' is selected over 'SSD'. Finally, 'Операционная система' is set to 'Debian' with a dropdown arrow.

### Субъективная оценка удобства

Документация по предоставляемым сервисам присутствует, и на первый взгляд, весьма полная. Однако количество самих сервисов предоставляемых PaaS сильно меньше, чем в Yandex Cloud, а также web интерфейс тормозной и глючный.

## Cloud.ru

### Определение конкретных сервисов

В соответствии с предполагаемой реализацией мною были выбраны следующие сервисы

- [виртуальная машина Evolution compute](#)
- [DaaS PostgreSQL](#)
- [Сервис](#) хранения данных пользователей ключей для авторизации

### Наличие пробного периода тарификации

Платформа предоставляет депозит в 4000 рублей при привязке карты.

### Анализ стоимости

Виртуальная машина небольшой мощности предоставляется бессрочно и бесплатно. Однако публичный IP для неё предоставляется за 150 рублей в месяц.

Стоимость сервиса с PostgreSQL составляет 1900 рублей за месяц. Система мониторинга предоставляется бесплатно. В результате выданного депозита хватит на работу проекта в течении двух месяцев.

### Субъективная оценка удобства

Документация присутствует, однако Cloud.ru подразделяется на ещё несколько это облачных платформ (Evolution, Advanced, Vmware), что немного мешает, тк документация представлена к каждому отдельно. Есть обучалка по панели управления.

## Вывод

Наиболее выгодным с точки зрения того, сколько сервис может проработать на депозите является cloud.ru, но у него есть значительный минус в виде отсутствия сервиса с базой данных в Evolution облаке.

Наиболее функциональным с точки зрения количества сервисов выглядит Яндекс облако, при этом на нём есть возможность уложиться в выдаваемый депозит, поэтому эта кандидатура выглядит наиболее подходящей под цели моего образовательного проекта.



# Подробная спецификация продукта

Для упрощения дальнейшего понимания ролей стоит упомянуть, что данный проект относится к модели B2B, т.е. основными клиентами сервиса являются отдельные лица или сообщества, которые хотят развернуть сервис не приобретая выделенные вычислительные мощности, а предоставляя его на своих устройствах.

## Предполагаемые роли клиентов в продукте

### Администратор сервиса

Данный вид пользователя является владельцем сервиса, именно он создаёт сервис, определяет настройки его доступности и имеет привилегию по его удалению. При развитии сервиса его привилегии будут расширяться, например, определением логики балансировки нагрузки между хостами. Идентификатором сервиса в дальнейшем будем называть пару состоящую из имени администратора и названия сервиса.

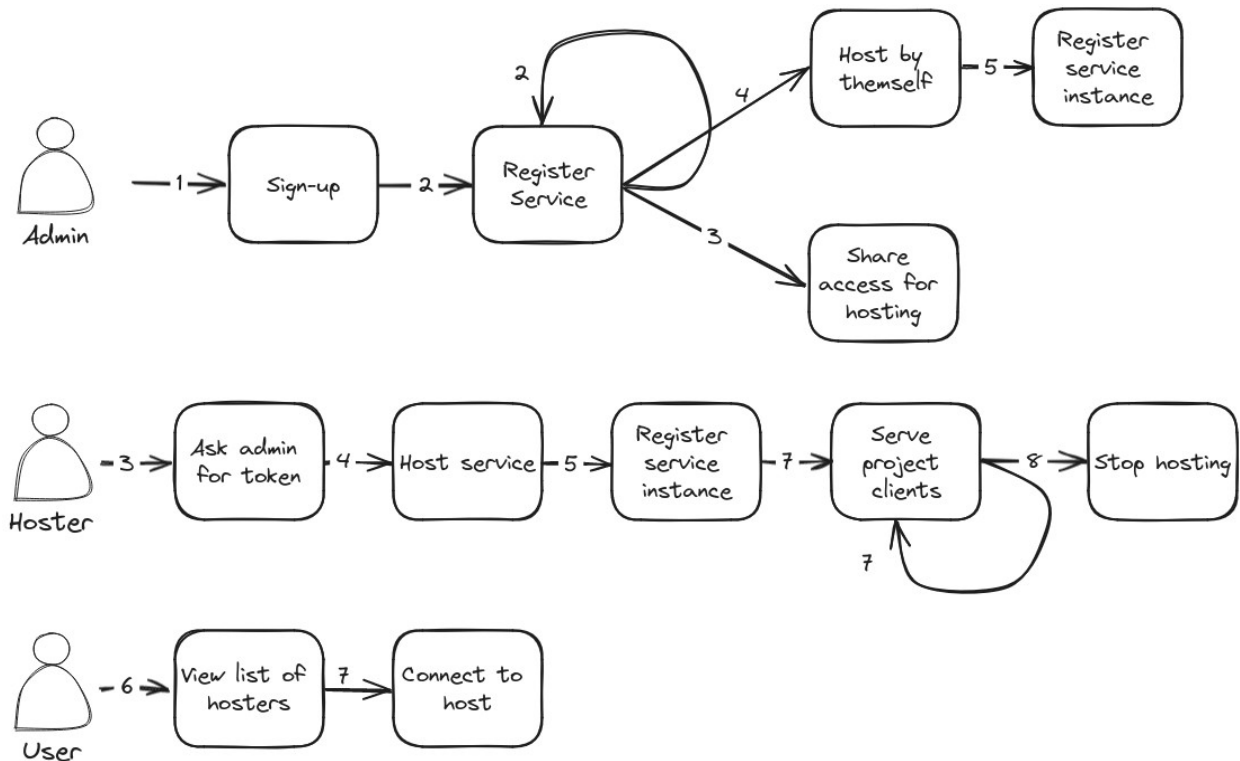
### Хост сервиса

Хостом сервиса можно стать по приглашению администратора. Наличие приглашения даёт возможность принимать участие в обслуживании клиентов сервиса, владельцем которого является администратор.

### Пользователь сервиса

Лицо, обращающееся по, обычно, публично опубликованному идентификатору сервиса, которого обслужит один из хостов сервиса.

## Подробная схема функциональности ролей



Ниже приведена подробная спецификация для каждого из действий на схеме выше:

Номер действия	1
Название	Регистрация аккаунта администратора
Кратное описание	Создание нового аккаунта администратора, для последующей регистрации сервиса
Акторы	Администратор
Предусловия	Отсутствуют
Основной поток	Пользователь вводит желаемое имя аккаунта, система проверяет свободно ли оно, и при положительном результате регистрирует пользователя и возвращает ему токен администратора. Пример команды: <i>client.py --sign-up handle</i>

Номер действия	2
Название	Регистрация сервиса
Кратное описание	Создание нового сервиса, привязанного к создающему его аккаунту администратора
Акторы	Администратор
Предусловия	1

Основной поток	Администратор вводит токен, подтверждая свою роль, и желаемое имя сервиса. Система проверяет, не заводил ли уже данный сервис этот администратор и в случае успеха, создаёт сервис и возвращает токен, позволяющий предоставлять сервис. Пример команды: <i>client.py --register-project handle --token «adm_token»</i>
----------------	---

Номер действия	3
Название	Наделение правами предоставления сервиса
Кратное описание	Администратор каким-либо удобным ему способом передаёт токен, разрешающий публикацию сервиса, доверенным лицам, которые публикуют
Акторы	Администратор, доверенное лицо(будущий хост)
Предусловия	2
Основной поток	Не специфицирован, т. к. выполняется не в рамках нашего приложения.

Номер действия	4
Название	Развёртывание сервиса
Кратное описание	Актор разворачивает сервис
Акторы	Администратор или доверенное лицо
Предусловия	3
Основной поток	Не специфицирован, т. к. выполняется не в рамках нашего приложения и полностью зависит от реализации предоставляемого сервиса.

Номер действия	5
Название	Регистрация экземпляра сервиса
Кратное описание	Актор сообщает о своей готовности обслуживать клиентов.
Акторы	Администратор или доверенное лицо
Предусловия	4
Основной поток	Актор предоставляет ключ от сервиса, который был получен при его создании, и порт, на котором развёрнут сервис. Сервер проверяет предоставленный ключ, и если он корректен, то развёрнутый актором экземпляр сервиса начинает участвовать в обслуживании клиентов. Пример команды: <i>client.py --start-hosting handle --token «hst_token» --port 8008</i>

Номер действия	6
Название	Поиск экземпляров сервиса

Кратное описание	Актор запрашивает список портов, на которых предоставляется сервис
Акторы	Любой пользователь
Предусловия	Отсутствуют
Основной поток	Актор зная, имя администратора и имя сервиса запрашивает список активных на данный момент экземпляров сервиса. Пример команды: <i>client.py --list --service admin/service</i>

Номер действия	7
Название	Предоставление сервиса
Кратное описание	Пользователь сервиса подключается к одному из экземпляров сервиса, после чего происходит непосредственное предоставление сервиса (полезная нагрузка)
Акторы	Конечный пользователь и администратор или доверенное лицо, развернувшее экземпляр сервиса
Предусловия	5, 6
Основной поток	Пользователь, ранее выполнивший запросивший список экземпляров сервиса (см действие 5), подключается к случайному экземпляру. Пример команды: <i>client.py --connect --service admin/service</i>

Номер действия	8
Название	Завершение предоставления сервиса
Кратное описание	Актор выключает экземпляр сервиса, после чего происходит автоматическая deregистрация данного сервиса.
Акторы	Администратор или доверенное лицо, развернувшее экземпляр сервиса
Предусловия	5
Основной поток	Не специфицирован, т. к. выполняется не в рамках нашего приложения и полностью зависит от реализации предоставляемого сервиса.

## Предполагаемая нагрузка

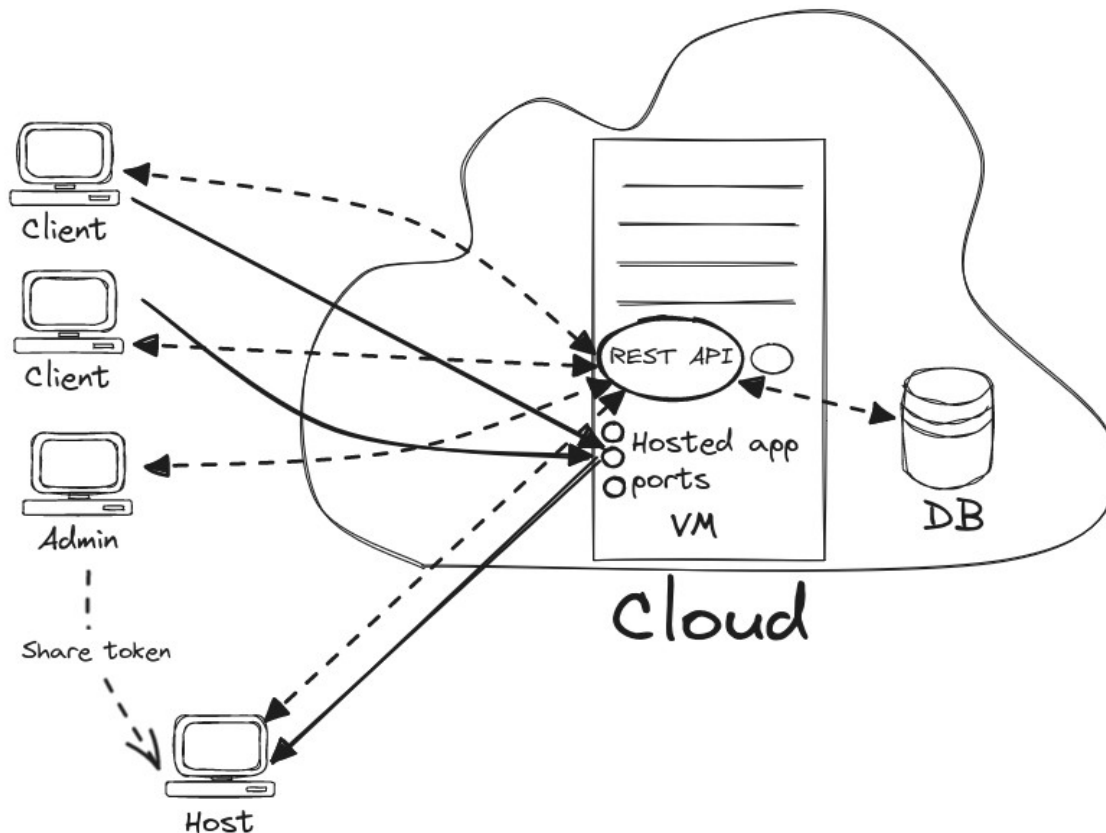
Большая нагрузка на сервис не ожидается, т. к. предполагаемая целевая аудитория это небольшие сообщества или организации, которые занимаются некоммерческой деятельностью и хотят сэкономить на хостинге путём использования локальных вычислительных мощностей. На мой взгляд, данная категория довольно нишевая, поэтому нагрузка будет не большой.

Далее в расчётах не будет учитываться количество служебных запросов от хостеров и администраторов, т. к. количество запросов от них пренебрежимо мало относительно запросов от обычных пользователей. Оценим сверху количество организаций, которые будут подключены к нашему сервису сотней. При этом, если в каждой из организации в среднем 50 человек и каждый из них 50 раз в день ищет нового хоста, что тоже весьма пессимистичная оценка. Тогда в сутки будет поступать 250 тыс запросов или же чуть более чем 2 запроса в секунду. Именно на эту нагрузку будем рассчитывать дальше, при рассмотрении возможностей по масштабированию.

# Планируемая реализация спецификации

## Архитектура сервиса

С точки зрения сети и программных компонентов



На схеме выше штрихом обозначены соединения, обеспечивающие служебную работу моего сервиса, а сплошным — трафик пользовательского приложения. Круг внутри виртуальной машины обозначает на ней порт.

В результате можно выделить следующие программные компоненты:

- Клиентское приложение, общее для всех ролей: клиента, администратора и хостера.
- Сервер, реализующий REST-аpi для клиентского приложения и взаимодействующий с базой данных.
- Хост система виртуальной машины. Стандартные образы не подходят, т. к. для приложения необходимы специфичные настройки OpenSSH.
- Реляционная база данных, в которой приложение хранит информацию для авторизации пользователей и информацию о подключённых хостах для каждого из приложений.

## Выбор технологий для программных компонентов

- Клиентское приложение реализовано на Python для ускорения разработки и кроссплатформенности, т. к. на нём есть отличная реализация OpenSSH и сокетов, которая работает на большинстве систем.
- Серверное приложение будет реализовано на Python, с использованием библиотеки Flask, для упрощения построения API.
- Образ для виртуальной машины будет собран при помощи технологии NixOS, т. к. это интересная технология для реализовать подхода IaC (Infrastructure as Code)., упрощающая развёртывание, тестирование, A/B обновления хостов в кластер.
- В качестве базы данных будет использован DbaaS (Managed service for YDB) по причинам, описанным в выборе облака и оценок стоимости решения.
- Для большей гибкости также используется API Gateway, предоставляемый облачным провайдером бесплатно при небольшой нагрузке.

## Система контроля прав доступа

Разделение ролей планируется сделать путём выдачи токенов.

Всего будет вида токенов:

- Admin token -- выдаётся при регистрации.
- Service token — разрешает подключение к сервису в качестве хоста для определённого проекта.
- Host token — необходим для деинициализации хоста.

Организацией хранения токенов должен озаботиться сам пользователь. В отличие от паролей, отсутствие шифрования для них не так страшно, т. к. токены решают проблему использования пользователем одних и тех же паролей в различных сервисах, поэтому их утечка не так критична.

Например, можно просто сохранять их в незашифрованных файлах на машине, где развёрнут экземпляр приложения, тогда кража токена значит наличие доступа к машине, но тогда наличие токена ничего не даёт, т. к. всё что можно с ним сделать, это отключить текущий хост(что бесполезно, т. к. он уже под контролем злоумышленника) или подключить к обслуживанию новый(что не ухудшает ситуацию, т. к. злоумышленник уже имеет доступ к приложению, которое обслуживает клиентов).

## Описание RESTful API

- **GET /admin/register (handle) → admin token**
- **POST /project/register (admin token, project handle) → project token**
- **POST /host (project token) → allocated port, host token**
- **DELETE /host (host, token) →**
- **GET /service → service hosted ports**

## Схема реляционной базы данных



## Ограничения реализации

- Из-за того, что соединение между клиентами и прокси-сервером установлено через протокол SSH, то пробрасываемый порт работает исключительно по протоколу TCP.
- Ограничение в количестве портов (что ограничивает количество клиентов-хостов) и объеме трафика, проходящего через прокси сервер.

## Возможности по масштабированию

В предлагаемой реализации проекта не все части могут масштабироваться горизонтально без значительных изменений. Масштабироваться горизонтально может база данных и REST сервис, если завернуть его в контейнер и запустить несколько экземпляров на одной более мощной виртуальной машине. Не масштабируется горизонтально количество свободных портов и сетевой трафик, приходящий на машину. Для решения этих проблем необходимо создавать и менять архитектуру, создавая несколько виртуальных машин и добавляя перед ними балансировщик нагрузки.



## Отклонения от плана во время реализации

На этапе планирования предполагалось, что для хоста и клиента будут завершать свои сессии повторным вызовом клиентской утилиты. В процессе реализации стало понятно, что этим и не удобно пользоваться и это усложняет реализацию. Поэтому было принято решение пересылать пакеты, пока одна из сторон не закрывает соединение. Если по какой-либо причине требуется завершать пересылку до завершения приложения, можно запускать клиентское приложение без флага `--detach` и принудительно прерывать его по требованию.

# Демонстрация работы

## Клиентское приложение-

Работа клиентского приложения тестировалось на GNU Linux дистрибутиве NixOS и Windows 10. В обоих случаях использовалась версия python 3.11.

Список опций клиентского приложения:

```
> ./client.py -h
usage: kdispatch [-h] [--token TOKEN] [--quiet] [--port PORT] [--detach] [--service handle/service-name]
                (--connect | --list | --start-hosting | --sign-up HANDLE | --register-project PROJ_NAME)

options:
  -h, --help            show this help message and exit
  --token TOKEN, -t TOKEN
                        access token
  --quiet, -q           prevent hints, may be usable for scripting
  --port PORT, -p PORT  local port for project sharing (if you are host) or receiving (if you are client)
  --detach              detach application from console after '--start-hosting'
  --service handle/service-name, -s handle/service-name
                        specify service for '--connect' and '--list' actions
  --connect, -c         connect local port 'port' to project
  --list, -l           show available hosts for specified project
  --start-hosting       start share project hosted on local 'port'
  --sign-up HANDLE      sign-up as administrator to get token allows project operations
  --register-project PROJ_NAME, -r PROJ_NAME
                        register new service with 'proj-name'
```

Ниже приведён типичный пример использования.

```
> ./client.py --sign-up ivanpetrov
adm_JDjvwENid9DwDcVMYFo5sA
Above you can see you administrator token. Save it and specify
for project-managing operations (e.g. --register-project) using --token argument
> ./client.py --register-project service --token adm_JDjvwENid9DwDcVMYFo5sA
prj_m4jOXjokLsgwYovVqQ1HedQ
Above you can see service hoster token. Save it and share with peoples, who
should be able to host your service with '--start-hosting'.
Of course, you can use that token by yourself
> ./client.py --start-hosting -p 8891 --token prj_m4jOXjokLsgwYovVqQ1HedQ
Hosting will be stopped automatically when specified service port is closed
DEBUG: host_token hst_OflGuU-xyLjWgeY_csr9-w, local_port 8891, remote_port 57109
Connect request registred

> nc -kl 127.0.0.1 8891
ping
pong

> ./client.py --connect -s 'ivanpetrov/service' --port 3388 --detach
57109
> nc 127.0.0.1 3388
ping
pong
```







В примере выше произошло следующее:

1. Регистрируется новый администратор
2. Создаётся новый проект

3. Запускается утилита netcat на порту 8891 (в левом нижнем углу), которая выступает в качестве сервера пользовательского сервиса
4. Экземпляр сервиса регистрируется для приёма клиентов на порт 8891
5. Клиент подключает локальный порт 3388 к сервису (в правом нижнем углу)
6. Запускается netcat на порту 3388, симулирующий пользовательское приложение сервиса.
7. По результату два netcat успешно подключаются к друг другу

## Облачная инфраструктура

Таблицы из YDB

Корневая директория	
Фильтр по имени	Фильтр по владельцу
Имя	Владелец
 admins	 ydb-user
 hosts	 ydb-user
 projects	 ydb-user

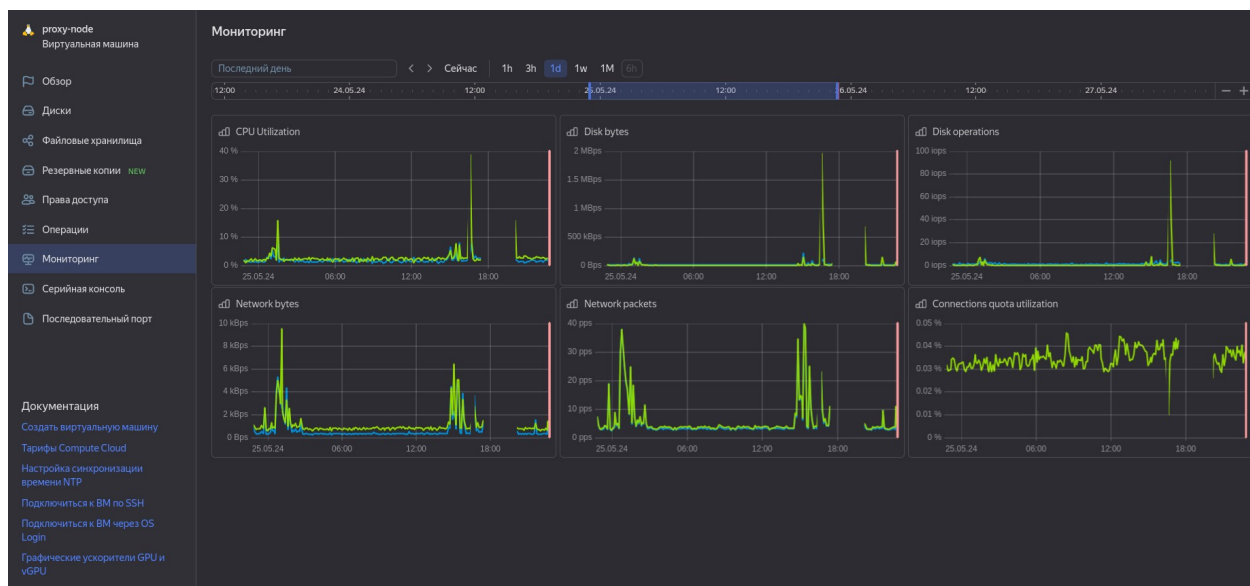
Вывод содержимого таблицы admins после демонстрации выше:

#	handle ▲	token ▲	▲
0	"ivan_petrov"	"adm_ndvWgd-LJJKq-zwKrb8I-w"	...
1	"ivanpetrov"	"adm_JDjvwENid9DwDcVMYFo5sA"	...
2	"khaser"	"adm_NEAQ9p1w2DN_7VRvWnltAA"	...

Мониторинг для YDB:

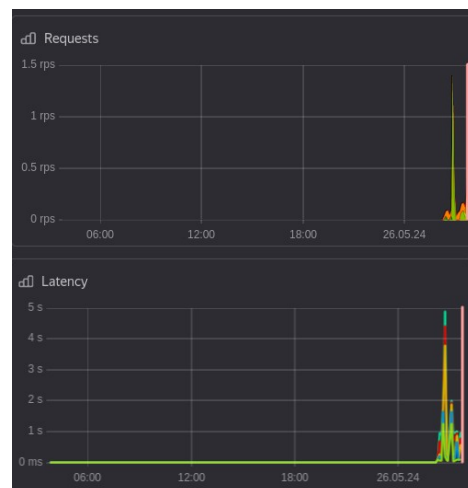


## Мониторинг для виртуальной машины:

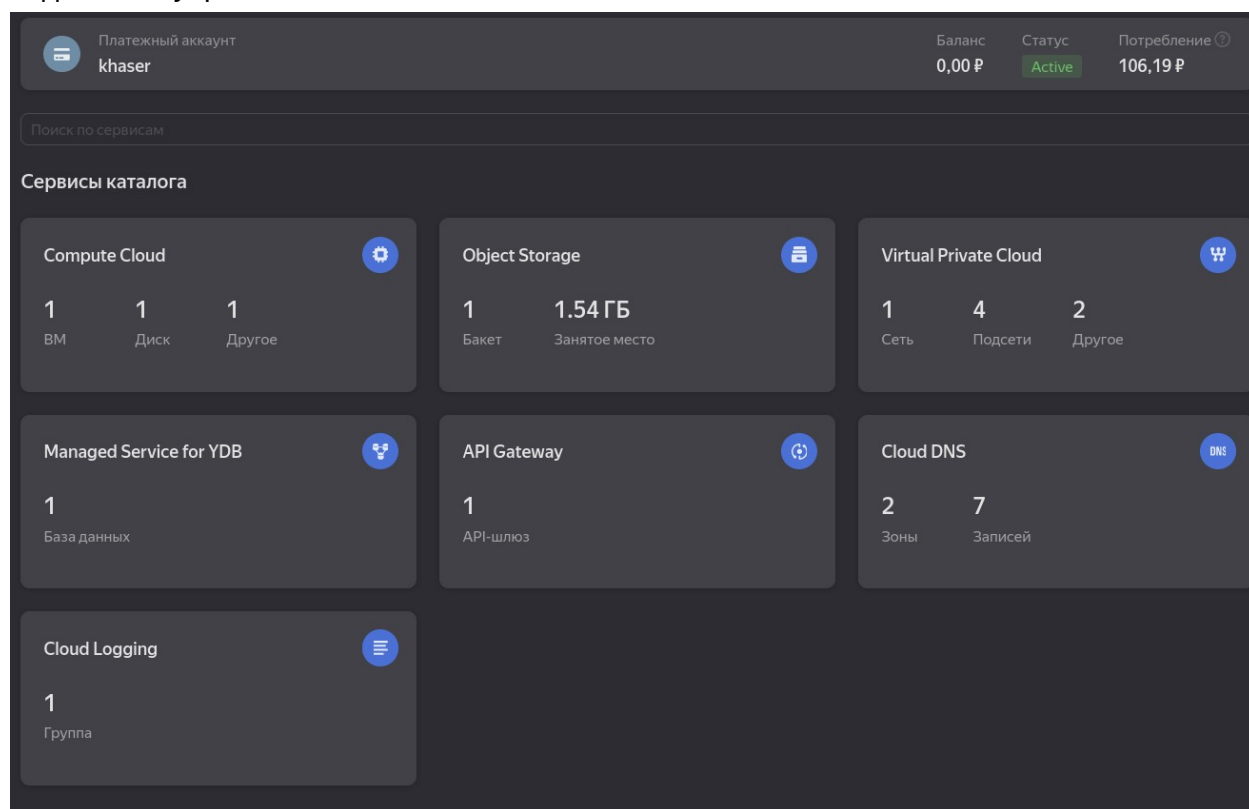


## Настройки API Gateway:

```
openapi: 3.0.0
info:
  title: kdispath server API
  version: 1.0.0
servers:
  - url: https://d5da41kf778qvb6k83as.apigw.yandexcloud.net
paths:
  /api/{path+}:
    x-yc-apigateway-any-method:
      x-yc-apigateway-integration:
        type: http
        url: http://130.193.38.230:8080/api/{path}
        query:
          '*': '*'
        headers:
          '*': '*'
          Host: 130.193.38.230:8080
        parameters:
          - name: path
            in: path
            required: false
            schema:
              type: string
```



Вид панели управления:



## Серверное приложение

В выводе ниже можно видеть как мои запросы, которые были получены во время демонстрации клиентского приложения выше (полученные с IP:81.89.176.2), так и запросы от сканеров уязвимостей третьих лиц.

```
-- Boot 520c35c37f584711a99f6a64894d16ac --
May 25 17:09:44 nixos systemd[1]: Started kdispatch-server.service.
May 25 17:10:41 nixos kdispatch-server[627]: * Serving Flask app 'server'
May 25 17:10:41 nixos kdispatch-server[627]: * Debug mode: on
May 25 17:10:42 nixos kdispatch-server[627]: WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI
May 25 17:10:42 nixos kdispatch-server[627]: * Running on all addresses (0.0.0.0)
May 25 17:10:42 nixos kdispatch-server[627]: * Running on http://127.0.0.1:8080
May 25 17:10:42 nixos kdispatch-server[627]: * Running on http://10.128.0.10:8080
May 25 17:10:42 nixos kdispatch-server[627]: Press CTRL+C to quit
May 25 17:10:42 nixos kdispatch-server[627]: * Restarting with stat
May 25 17:10:45 nixos kdispatch-server[945]: * Debugger is active!
May 25 17:10:45 nixos kdispatch-server[945]: * Debugger PIN: 108-660-552
May 25 17:30:39 nixos kdispatch-server[945]: 159.65.11.244 - - [25/May/2024 17:30:39] "CONNECT www.google.com:443 HTTP/1.1" 404 -
May 25 17:49:39 nixos kdispatch-server[945]: 87.121.69.27 - - [25/May/2024 17:49:39] "CONNECT api.rev.pm:443 HTTP/1.1" 404 -
May 25 17:50:29 nixos kdispatch-server[945]: 141.98.11.15 - - [25/May/2024 17:50:29] "CONNECT google.com:443 HTTP/1.1" 404 -
May 25 18:04:10 nixos kdispatch-server[945]: 38.222.47.20 - - [25/May/2024 18:04:10] "POST /goform/set_LimitClient_cfg HTTP/1.1" 404 -
May 25 18:38:31 nixos kdispatch-server[945]: 81.89.176.2 - - [25/May/2024 18:38:31] "GET /api/service/hosts?handle=khaser&name=svr HTTP/1.1" 404 -
May 25 18:42:10 nixos kdispatch-server[945]: 81.89.176.2 - - [25/May/2024 18:42:10] "GET /api/admin/register?handle=khaser HTTP/1.1" 308 -
May 25 18:42:10 nixos kdispatch-server[945]: 81.89.176.2 - - [25/May/2024 18:42:10] "GET /api/admin/register/?handle=khaser HTTP/1.1" 200 -
May 25 18:42:33 nixos kdispatch-server[945]: 81.89.176.2 - - [25/May/2024 18:42:33] "POST /api/service/register HTTP/1.1" 200 -
May 25 18:42:53 nixos kdispatch-server[945]: 81.89.176.2 - - [25/May/2024 18:42:53] "GET /api/service/hosts?handle=khaser&name=svr HTTP/1.1" 200 -
May 25 19:20:45 nixos kdispatch-server[945]: 197.53.188.30 - - [25/May/2024 19:20:45] "POST /goform/set_LimitClient_cfg HTTP/1.1" 404 -
May 25 19:20:46 nixos kdispatch-server[945]: 197.53.188.30 - - [25/May/2024 19:20:46] "POST /goform/set_LimitClient_cfg HTTP/1.1" 404 -
May 25 19:23:27 nixos kdispatch-server[945]: 81.89.176.2 - - [25/May/2024 19:23:27] "GET /api/admin/register?handle=ivan_petrov HTTP/1.1" 308 -
May 25 19:23:27 nixos kdispatch-server[945]: 81.89.176.2 - - [25/May/2024 19:23:27] "GET /api/admin/register/?handle=ivan_petrov HTTP/1.1" 200 -
May 25 19:23:52 nixos kdispatch-server[945]: 81.89.176.2 - - [25/May/2024 19:23:52] "GET /api/admin/register?handle=ivan_petrov HTTP/1.1" 308 -
May 25 19:23:53 nixos kdispatch-server[945]: 81.89.176.2 - - [25/May/2024 19:23:53] "GET /api/admin/register/?handle=ivan_petrov HTTP/1.1" 409 -
May 25 19:24:02 nixos kdispatch-server[945]: 81.89.176.2 - - [25/May/2024 19:24:02] "GET /api/admin/register?handle=ivanpetrov HTTP/1.1" 308 -
May 25 19:24:02 nixos kdispatch-server[945]: 81.89.176.2 - - [25/May/2024 19:24:02] "GET /api/admin/register/?handle=ivanpetrov HTTP/1.1" 200 -
May 25 19:25:20 nixos kdispatch-server[945]: 81.89.176.2 - - [25/May/2024 19:25:20] "POST /api/service/register HTTP/1.1" 200 -
May 25 19:27:14 nixos kdispatch-server[945]: 81.89.176.2 - - [25/May/2024 19:27:14] "POST /api/service/hosts HTTP/1.1" 200 -
May 25 19:29:24 nixos kdispatch-server[945]: 87.236.176.226 - - [25/May/2024 19:29:24] "GET / HTTP/1.1" 404 -
May 25 19:29:55 nixos kdispatch-server[945]: 81.89.176.2 - - [25/May/2024 19:29:55] "GET /api/service/hosts?handle=ivanpetrov&name=service HTTP/1.1" 200 -
May 25 19:30:33 nixos kdispatch-server[945]: 198.199.111.41 - - [25/May/2024 19:30:33] "GET / HTTP/1.1" 404 -
May 25 19:30:51 nixos kdispatch-server[945]: 81.89.176.2 - - [25/May/2024 19:30:51] "GET /api/service/hosts?handle=ivanpetrov&name=service HTTP/1.1" 200 -
lines 12018-12056/12056 (END)
```

## Вывод

Ссылка на github репозиторий проекта: <https://github.com/khaser/kdispatch/>

Проект сделан на [devops экосистеме Nix](#), которая существенно упростила построение CI, деплой решения, написание интеграционных тестов, которые запускаются на кластере виртуальных машин. Для улучшения воспроизводимости используется экспериментальное расширение Nix Flakes, добавляющее lock файл для абсолютно всех зависимостей. Однако использование Nix не является обязательным, допустима и ручная настройка образа, опираясь на *configuration.nix* и установка в него Python пакетов, которые были описаны через *setuptools*.

Проект помог мне лучше понять как услуги PaaS платформы могут упростить разработку и будущую поддержку приложения и какие типовые проблемы при этом могут возникать. Также это мой первый опыт с YDB, которая на данный момент мне показалась очень плохо приспособленной к работе с языками, отличными от C++. Для Python нет ORM библиотеки, хорошо поддерживающей YDB(есть только [проекты](#) находящиеся на раннем этапе разработки), поэтому пришлось писать на SQL, что при такой простой схеме БД является неоправданным усложнением.

Для проекта есть множество направлений развития:

- сделать клиентский интерфейс в виде python библиотеки, это позволит python приложениям использовать наш сервис настолько прозрачно, что пользователь даже не будет знать, об использовании приложением данного проекта. Это позволит выйти проекту на следующий уровень, став уже более похожим на фреймворк для децентрализованных приложений.
- предоставить возможность пользователю выбирать подключение к конкретному хосту, а не подключаться к случайному
- организовать работу приложения таким образом, чтобы весь трафик между клиентом и хостом шёл в обход сервера.