

Содержание

Введение.....	4
Раздел 1. Основы облачных вычислений.....	6
1.1. Обзор облачной платформы «Яндекс.Облако»	6
1.2. Регистрация аккаунта на портале	16
Раздел 2. Виртуальные машины	21
2.1 Подготовка к созданию ВМ.....	24
2.2. Создание ВМ в «Яндекс.Облаке»	26
2.3. Запуск и работа в ВМ	32
Раздел 3. Базы данных	37
3.1. Модели баз данных.....	40
3.2. Языки запросов	44
3.3. СУБД.....	47
3.4. Создание БД в Яндекс.Облаке	51
Раздел 4. Создание сайтов.....	63
4.1.	6
Раздел 5. и	7
5.1.	7
Раздел 6. т	60
6.1.	68
Список литературы.....	72

Введение

Дисциплина «Облачные вычисления и системы» в учебном плане Ульяновского государственного технического университета относится к базовой части двух направлений подготовки:

- 09.03.02 «Информационные системы и технологии» профиля «Информационные системы и технологии»;
- 12.03.01 «Приборостроение» профиля «Индустриальный интернет».

В результате изучения данной дисциплины студенты на основе приобретенных знаний, умений и навыков достигают освоения компетенций на определенном уровне согласно учебным планам и рабочим программам выше перечисленных направлений:

- ✓ способность понимать принципы работы современных информационных технологий и программных средств, в том числе отечественного производства, и использовать их при решении задач профессиональной деятельности (ОПК-2 для направления 09.03.02);
- ✓ способность разрабатывать алгоритмы и программы, пригодные для практического применения в области информационных систем и технологий (ОПК-6 для направления 09.03.02);
- ✓ способность к анализу, проектированию, использованию и работе с нормативно-метрологической документацией контрольно-измерительных приборов, систем и комплексов, их элементов и узлов (ПК-1 для направления 12.03.01).

Освоение студентами этих компетенций предполагает использование при их обучении базовых знаний, полученных ими при изучении других дисциплин. В частности предполагается, что у студентов имеются знания по базам данных, системному администрированию, сетям и навыки программирования на языках высокого уровня.

В данное учебное пособие, помимо дополнительной теории к основным разделам дисциплины, входит лабораторный цикл, содержащий 6 лабораторных работ по изучению облачных технологий на платформе компании «Яндекс». Эти облачные технологии представлены в виде сервисов на портале **cloud.yandex.ru**, которые предоставляются под общим названием «Яндекс.Облако».

Выполнение цикла лабораторных работ на платформе компании «Яндекс» обусловлено тем, что данная компания предоставляет зарегистрированным пользователям бесплатно стартовый грант в размере 4000 рублей (для резидентов РФ) сроком на 60 дней. Данный грант условно делится на две части, которые распределяются на сервисы особым образом. Первая часть (1000 рублей) может использоваться для оплаты услуг сервиса **Yandex Compute Cloud** и инструментов **Yandex Cloud Marketplay**. Вторая часть (3000 рублей) может использоваться для оплаты услуг всех других сервисов «Яндекс.Облако».

В качестве альтернативы предлагаемого цикла лабораторных работ для получения зачета по дисциплине студенты могут выполнить 2-3 модуля на выбор общим объемом не менее 12 часов бесплатного курса от компании «Яндекс», который на портале «Яндекс.Облако» называется «Инженер облачных сервисов».

Обязательным требованием, как при выполнении цикла лабораторных работ, так и при альтернативном выполнении модулей курса «Инженер облачных сервисов», является содержание в имени аккаунта фамилии студента на латинице.

Основное назначение пособия – обеспечение необходимой теорией лабораторного практикума дисциплины «Облачные вычисления и системы» для студентов выше перечисленных направлений, однако оно может быть полезным и для самостоятельной подготовки студентов других направлений, которые планируют применять облачные технологии в своей дальнейшей работе.

Раздел 1. Основы облачных вычислений

Рассмотрим основные понятия и термины, которые применяются в настоящее время при описании облачных технологий. Любая облачная платформа представляет собой совокупность связанных сервисов, сдающих в аренду пользователю вычислительные мощности в том объеме, который он затребовал.

Главная отличительная особенность состоит в том, что доступ к этим вычислительным мощностям пользователь получает через сеть Интернет. Такой подход к потреблению вычислительных мощностей принято называть облачные вычисления.

По своей сути такой подход предполагает замену или дополнение традиционных вычислительных центров, расположенных на территории предприятий потребителей. При этом облачные провайдеры берут на себя все задачи, связанные с поддержанием работоспособности и высокой производительности аппаратного и программного обеспечения, предоставляемого в аренду.

Принято выделять три основные модели облачных вычислений, которые предоставляются провайдерами:

- IaaS (инфраструктура как сервис),
- PaaS (платформа как сервис),
- SaaS (программное обеспечение как сервис).

Однако в последние годы появились и другие модели оказания услуг, которые в настоящее время набирают все большую популярность. К этим моделям относятся DBaaS (база данных как сервис), Serverless (бессерверные вычисления) и FaaS (функция как сервис).

База данных (БД) – совокупность данных, хранение которых организовано по определенным правилам, предусматривающим общие принципы описания и манипулирования данными, независимыми от прикладных программ. Часто БД относятся к узкой предметной области и организованы для использования многими пользователями.

Рассмотрим характерные отличительные особенности основных и вспомогательных моделей облачных вычислений.

Модель *IaaS* предполагает получение доступа к виртуальным серверам, сетям, хранилищам и другим ресурсам на базовом уровне. Это имеет сходство с традиционным способом работы с инфраструктурой, осуществляемым отделом ИТ. Главное отличие состоит в том, что физическое оборудование расположено в дата-центре провайдера облачных услуг. При этой модели имеется возможность регулировать производительность ресурсов в зависимости от потребностей.

В модели *PaaS* пользователь не занимается серверами, хранилищами и приложениями. Он просто выбирает из доступного списка серверы и среды, необходимые для запуска, тестирования, развертывания, обновления и масштабирования его приложений.

Модель *SaaS* предполагает получение готовых программ, запускаемых и управляемых провайдером облачных услуг. Доступ к ним происходит через браузер, API или программу-клиент пользователя. Преимущество этой модели услуг – автоматическое обновление предоставляемых программ и защита от потери данных.

На рис. 1.1 представлена структурная схема, на которой сопоставлены три основные модели облачных вычислений.

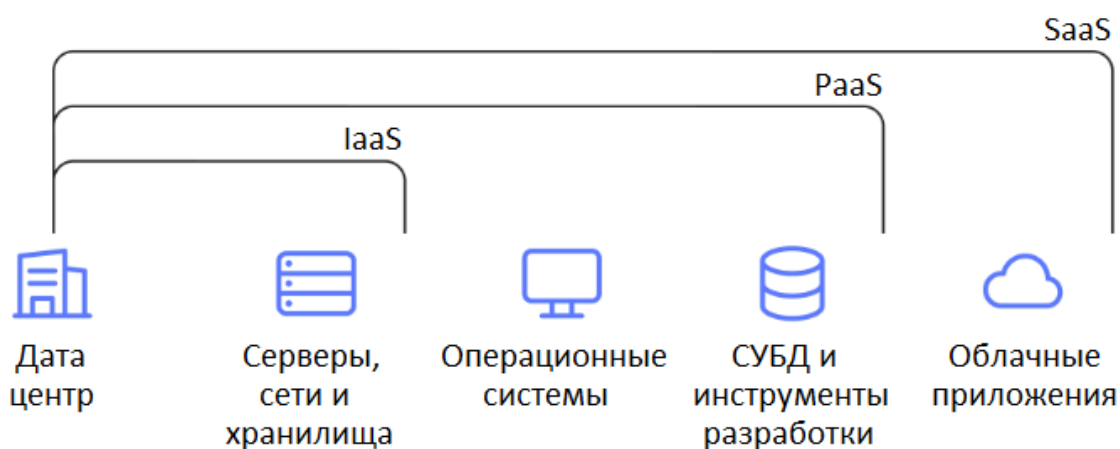


Рис. 1.1 Структура основных моделей облачных вычислений

Для модели *Serverless* все задачи по управлению облаком возлагаются на провайдера, а пользователь сосредотачивается на создании кода и логики разрабатываемых приложений. Он получает виртуальную машину (ВМ) для выполнения запроса, которая уничтожается после завершения работы. Таким образом, будут оплачиваться только использованные ресурсы.

В модели *FaaS* пользователь запускает некоторую функцию (часть кода приложения) в ответ на определенное событие, в качестве которого может выступать, например HTML-запрос, а провайдер предоставляет ресурсы для выполнения запуска. В этом случае проще масштабировать код и вводить микросервисы.

При модели *DBaaS* пользователь получает доступ к БД любого типа по запросу, не занимаясь вопросами управления БД и их поддержкой. Это позволяет существенно снизить нагрузку на отдел ИТ и уменьшить сроки запуска проектов.

В модели *KaaS* (Kubernetes как сервис) пользователь получает удобную управляемую систему развертывания контейнеров без необходимости обслуживания и администрирования ИТ-инфраструктуры. Автоматическое масштабирование нагрузки в Kubernetes позволяет обеспечить гарантированную доступность приложений для клиентов даже во время пиковых нагрузок.

Модель *IaC* (инфраструктура как код) предполагает, что пользователь настраивает свою инфраструктуру аналогично тому, как создается программное обеспечение (ПО). Это так называемая DevOps-практика, благодаря которой автоматизируется управление инфраструктурой и конфигурациями.

Современные облачные платформы обычно содержат несколько категорий ресурсов. Наиболее часто востребованы такие категории ресурсов как виртуальные машины, диски, базы данных. Рассмотрим более подробно наиболее востребованные категории ресурсов.

Виртуальные машины – это абстрактные вычислительные контейнеры, созданные программой, работающей на другой физической машине. Хотя ВМ физически не существуют, но они работают, как настоящий физический компьютер.

Другими словами можно сказать, что пользователь создаёт виртуальный компьютер внутри другого физического компьютера. Несколько ВМ могут работать параллельно на одном физическом компьютере, причем независимо друг от друга.

Тот физический компьютер, на котором создаются ВМ, называется хостом, а ВМ называются «гостевыми». На одном хосте может быть множество гостевых ВМ и множество виртуальных серверов.

Выше было сказано, что ВМ создается программно, однако она использует реальные физические ресурсы хоста, такие как процессор, оперативная память и место на диске (физическое дисковое пространство, выделяемое на жестком SSD-диске).

На хосте можно настроить столько ВМ, сколько необходимо, но физические аппаратные ресурсы придется разделить между ними всеми. Число ВМ, которые можно иметь одновременно, ограничено только ресурсами хоста. Однако нужно помнить, что эти ВМ будут работать медленнее, чем реальный физический компьютер.

1.1 Обзор облачной платформы «Яндекс.Облако»

Облачная платформа «Яндекс.Облако» была запущена в тестовом режиме российской интернет-компанией «Яндекс» в 2018 году. Главная цель данной облачной платформы – создать отечественную альтернативу аналогичным зарубежным платформам.

В настоящее время облачная платформа «Яндекс.Облако» позволяет получить полноценный доступ к современным облачным технологиям в виде различных сервисов. На сегодняшний день общее количество сервисов приближается к пятидесяти.

С помощью сервисов «Яндекс.Облака» можно осуществлять вычисления и хранение данных, проводить масштабирование, создавать новые проекты и приложения, а также выполнять и более сложные задачи. Работа с облачной платформой доступна, как корпоративным пользователям, так и частным лицам и может осуществляться по двум основным моделям предоставления услуг *PaaS* (платформа как сервис) и *IaaS* (инфраструктура как сервис). На рис. 1.2 представлена архитектура облачной платформы «Яндекс.Облако».

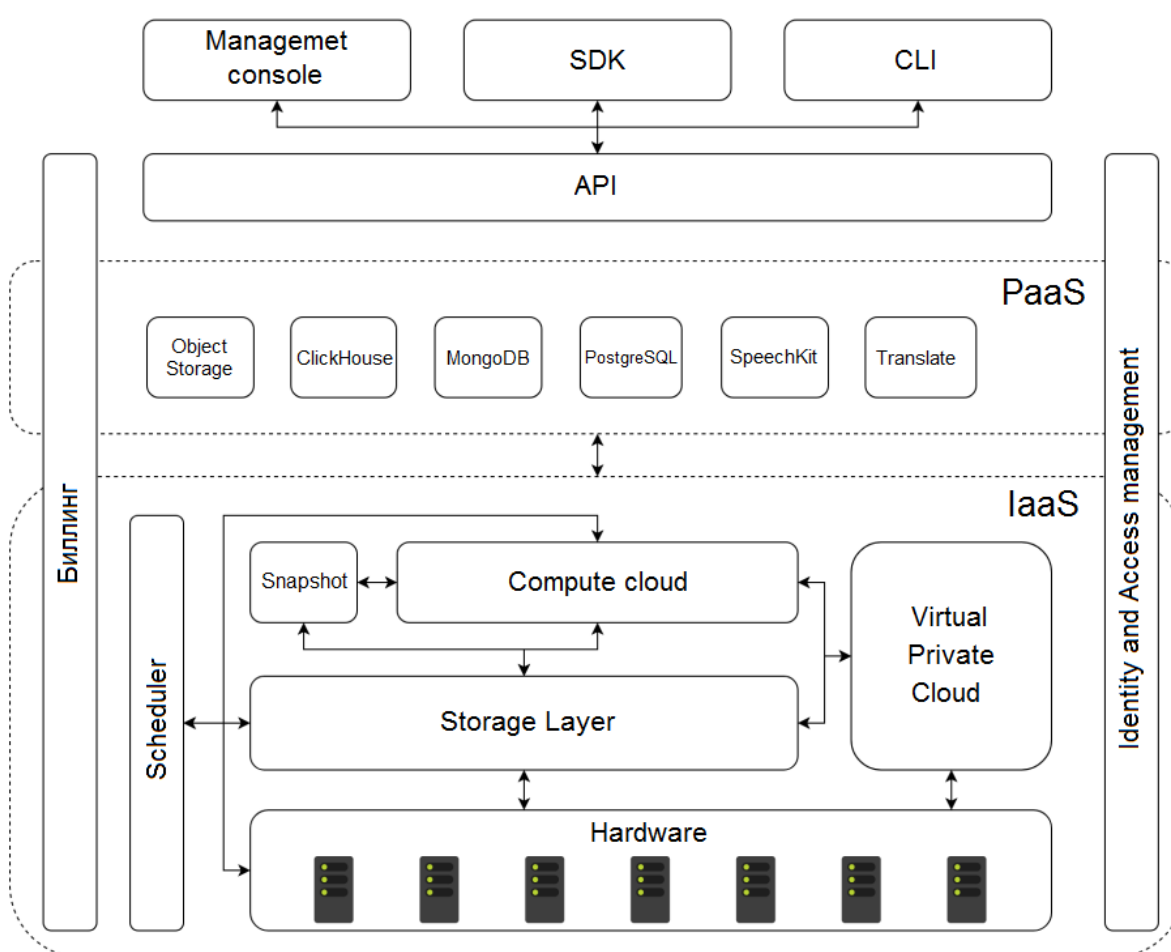


Рис. 1.2 Архитектура «Яндекс.Облака»

Управление инфраструктурой «Яндекс.Облака» может осуществляться с использованием графического интерфейса, командной строки либо с помощью инструментов разработки Python или Go.

Все сервисы «Яндекс.Облака» разделены на две группы в соответствии с двумя моделями предоставления услуг: *платформенные*, предоставляемые по модели *PaaS* и *инфраструктурные*, которые предоставляются по модели *IaaS*.

Платформенные сервисы позволяют создавать приложения, основываясь на управляемых базах данных, таких как MongoDB, PostgreSQL, ClickHouse, содержащих функции администрирования. Рассмотрим популярные платформенные сервисы более подробно.

Yandex Managed Service for PostgreSQL. Этот сервис разворачивает и поддерживает кластеры СУБД PostgreSQL и его помощью можно восстановить фрагмент БД за неделю, создать хосты для хранения копий, создать автоматически обновляемые реплики, подобрать способ хранения данных, создать новые БД и многое другое. При этом обеспечивается высокая безопасность информации и высокая скорость ее обработки даже при работе с большими объемами.

Yandex Managed Service for ClickHouse. Поддерживает и разворачивает кластеры БД, основанные на ClickHouse. Данный тип БД позволяет обрабатывать более миллиарда строк или десятков гигабайт информации за одну секунду и находит применение для обработки аналитических запросов в режиме реального времени на структурированных больших данных. Этот сервис позволяет выбрать тип хранилища, автоматически настраивать ZooKeeper (приложение для управления кластером), дает возможность настроить, восстановить БД, создавать хосты резервного копирования в разных зонах доступности и имеет высокую степень надежности благодаря современным методам шифрования.

Yandex Managed Service for MongoDB. Этот сервис разворачивает и поддерживает БД MongoDB. Основные достоинства этого типа БД: высокая устойчивость к различным сбоям, возможность выбора типа хранилища, легкое обслуживание, копии в разных зонах доступности,

современные технологии для изоляции и шифрования, а также ускоренный принцип обработки информации.

Yandex Translate. Позволяет интегрировать переводчик Яндекса в приложения и веб-проекты для конечных пользователей. Распознает около 90 языков, а также может переводить как тексты целиком, так и отдельные слова. Основные достоинства: автоматическое определение языка, точный перевод за счет статистической схемы трансляции, а также улучшение качества перевода за счет машинного обучения.

Yandex SpeechKit. Используется как основа для голосового помощника Алисы и является уникальной технологией по синтезу и распознаванию речи. Основные возможности: работает с тремя языками (русский, английский и турецкий) в текстовом и аудиальном режимах, максимально естественно воспроизводит речь, производит быстрый синтез текста в реальном времени, а также выполняет обмен данных с помощью удобного HTTP API.

Yandex Managed Service for MySQL. Разворачивает и поддерживает БД MySQL. Основные функции: непрерывное создание снимков БД (позволяет восстановить копию последней недели), хранение копий в разных зонах доступа, наличие read-реплик (позволяют строить каскадные и произвольные топологии), выбор локального или сетевого хранилища, наличие доступа к настройкам БД, восстановление резервных дублей, просмотр графиков статистики и логов, высокий уровень шифрования с полной изоляцией от других БД.

Yandex Managed Service for Redis. Поддерживает и разворачивает БД, основанные на СУБД Redis. Основные возможности: сохранение информации в формате ключ-значение, указание в настройках временного диапазона хранения, генерация автоматических копий при создании хостов, высокая скорость обработки данных, изоляция БД друг от друга и дополнительная система шифрования.

Инфраструктурные сервисы предназначены для обеспечения любого проекта пользователя основными ресурсами: безопасными хранилищами информации, обработчиками данных, а также системами безопасного доступа и обмена трафиком. Общая инфраструктура «Яндекс.Облака» разделена на три независимые зоны доступности «А» (Владимирская область), «В» (Рязанская область) и «С» (Московская область). Именно благодаря наличию этих зон и достигается высокая надежность, устойчивость к отказам и возможность сохранять копии. Пользователи вправе самостоятельно определить зону доступности для своего облака. Чтобы обеспечить еще более высокую надежность и устойчивость к различным видам отказов планируется построение нового четвертого дата-центра в Калуге.

Yandex Compute Cloud. Данный сервис предоставляет необходимые масштабируемые вычислительные мощности для решения разнообразных задач пользователя. Управление можно осуществлять с помощью консоли, SKD, API либо CLI (командная строка). ПО можно устанавливать самостоятельно либо заказать предустановку на этапе первичной настройки. Из операционных систем (ОС) доступны Windows Server и Linux, а из ПО доступны DSVM (программы для машинного обучения и анализа данных) и LAMP (набор утилит Apache, MySQL, PHP). Важные понятия сервиса: *диск* – виртуальный аналог реального SSD-диска, *снимок диска* – копия всей файловой системы. Снимок диска позволяет сделать резервную копию исходной и всех промежуточных версий, а также перенести данные с одного диска на другой. Это позволяет выполнить миграцию файловой системы из одной зоны доступности в другую. Основные возможности: выбор числа ядер процессора, объема ОЗУ, размера и количества дисков. Также выбираются: нужная ОС, протестированное ПО и число ВМ (для каждой ВМ определяется ее зона доступности). Можно подключить несколько сетевых SSD-дисков для хранения информации.

Yandex Object Storage. Этот сервис предназначен для масштабируемого хранения данных. Максимальный размер одного файла не должен превышать 5 Тб. Обычно используется серверами с высокой нагрузкой для быстрого и надежного доступа к данным. Предоставляемые возможности: неограниченное расширяемое место для хранения, резервное копирование в разные зоны доступности, объектное хранилище 2-х типов (стандартное – для часто используемых файлов и холодное – для редко используемых). Обмен данными можно осуществлять различными средствами: с помощью командной строки, графического клиента, Python SDK, HTTP API или Java.

Yandex Virtual Private Cloud. С помощью этого сервиса можно быстро создать облачную сеть для передачи данных между облачными ресурсами, а также для их связи с интернетом. Предоставляются следующие возможности: регулировка пропускной способности, снижение уровня задержек, изоляционный обмен данными, управление подсетями с возможностью выбора внутренних адресов.

Yandex Identity and Access Management. Сервис предназначен для управления доступами к облачным ресурсам. Он позволяет назначать участников, их роли и создавать специальные аккаунты с настройкой аутентификации. Основные функции: аутентификация через учетную запись Яндекса, настройка полномочий участников команды, возможность создания сразу нескольких сервисных аккаунтов.

Yandex Resource Manager. Этот сервис позволяет структурировать имеющиеся в облаке ресурсы. С его помощью можно осуществлять управление доступом к различным ресурсам, создавать новые каталоги и назначать роли участникам. Дополнительные возможности: объединение ресурсов в один или несколько каталогов для группы участников, сделать открытое облако или присвоить участникам свои роли, управление облаком и каталогами с помощью API, графического интерфейса или командной строки.

Yandex Instance Groups. Это новый компонент, созданный для помощи пользователям в развертывании и горизонтального масштабирования ВМ. С его помощью пользователь может создавать в облаке группы однотипных ВМ. Основные функции: настройка числа ВМ и политики, автоматическое масштабирование количества ВМ, подключение к сетевому балансировщику нагрузки.

Yandex Load Balancer. Данный сервис отвечает за отказоустойчивость приложения, помогая создать и настроить балансировщик с распределением трафика. Основные возможности: обработка любых объемов входящего трафика, обработка сетевых пакетов без задержки, автоматическое определение готовности ресурса к приему нового трафика (при отсутствии готовности – отмена запроса), правильное распределение нагрузки на имеющиеся ресурсы (распределение по адресу), хранение IP-адреса в нужном сетевом пакете запроса. Этот сервис хорошо справляется с правильным распределением веб-трафика и обеспечивает нужный уровень отказоустойчивости.

Yandex Message Queue. Этот сервис предназначен для быстрой и надежной передачи сообщений между различными приложениями. Основные функции: очереди двух типов (FIFO и стандартные), копирование и хранение нескольких дубликатов, сохранение на твердых носителях. Для работы используются интерфейсы командной строки, HTTP API, библиотеки различных языков программирования.

Yandex Container Registry. Данный сервис позволяет управлять и развертывать, а также хранить в облачной инфраструктуре Docker-образы (исполняемый пакет, содержащий все необходимое для запуска приложения). Этот сервис используют для улучшения скорости скачивания, хранения Docker-образов в отказоустойчивом хранилище Yandex Object Storage, а также для определения, кому именно будет предоставлен доступ для работы с образом. Сервис позволяет легко и быстро скачать нужный образ в любой момент времени.

1.2 Регистрация аккаунта на портале

ЛР1. *Цель:* Осуществить регистрацию аккаунта на облачной платформе «Яндекс.Облако» и создать платежный аккаунт для получения бесплатного стартового гранта.

Пример выполнения лабораторной работы №1 описан с помощью пронумерованных шагов. Каждый шаг при этом сопровождается рисунком экрана (скрином), соответствующим этому шагу.

Задание. Выполните самостоятельно все шаги аналогично приведенному примеру. В качестве варианта задания на рисунках всегда должна быть представлена *персональная информация* о студенте, выполнившем лабораторную работу. Это может быть логин входа на портал Яндекса или фамилия студента. В примере выполнения эта персональная информация выделена красной рамкой.

Шаг 1. Откройте браузер и наберите адрес облачной платформы «Яндекс.Облако» www.cloud.yandex.ru. После входа на портал вам будет предложено создать свое облако. Для этого в соответствующем окошке необходимо указать название своего облака и нажать кнопку «Создать», как показано на рисунке 1.3.

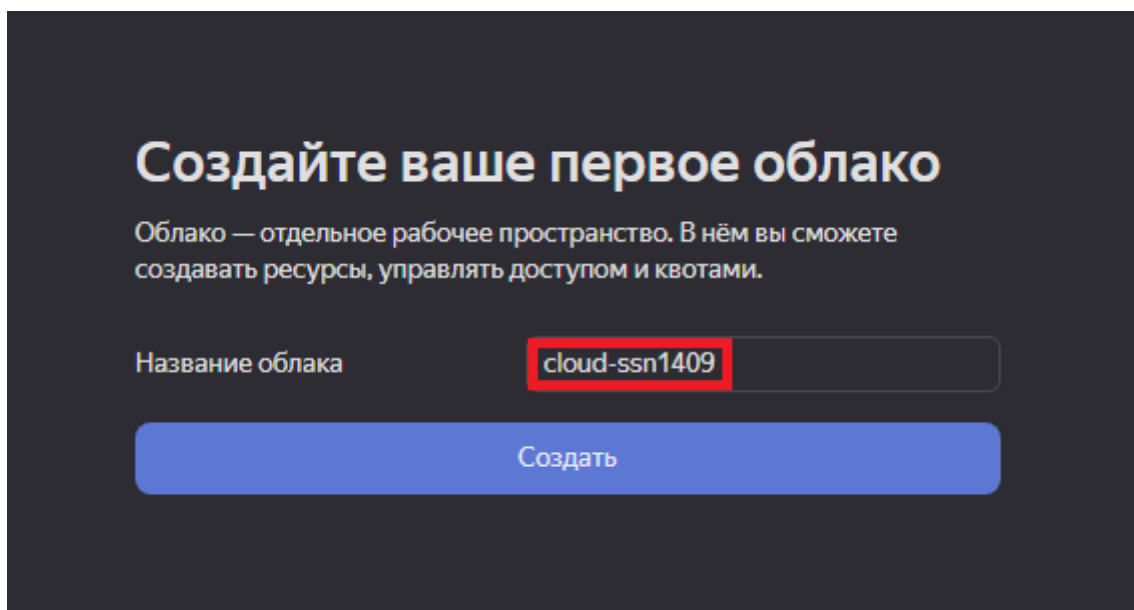


Рис. 1.3 Ввод названия создаваемого «Яндекс.Облака»

Шаг 2. После создания облака откроется окно портала на вкладке «Дашборд каталога», как показано на рис. 1.4, где внизу приводится информация о сервисах, которые в настоящее время имеются на платформе «Яндекс.Облако».

В средней области окна приведена информация о том, что было создано виртуальное частное облако Virtual Private Cloud, которое уже содержит одну виртуальную сеть и три подсети.

Вверху открытой вкладки окна приведена информация и предложение о создании платежного аккаунта. В этом случае пользователю предоставляется стартовый гранд на тестирование в размере 4000 руб. Этот гранд действует в течение 60 дней и за этот срок студенты должны успеть выполнить все лабораторные работы. Чтобы воспользоваться стартовым грантом нажмите на кнопку «Создать аккаунт».

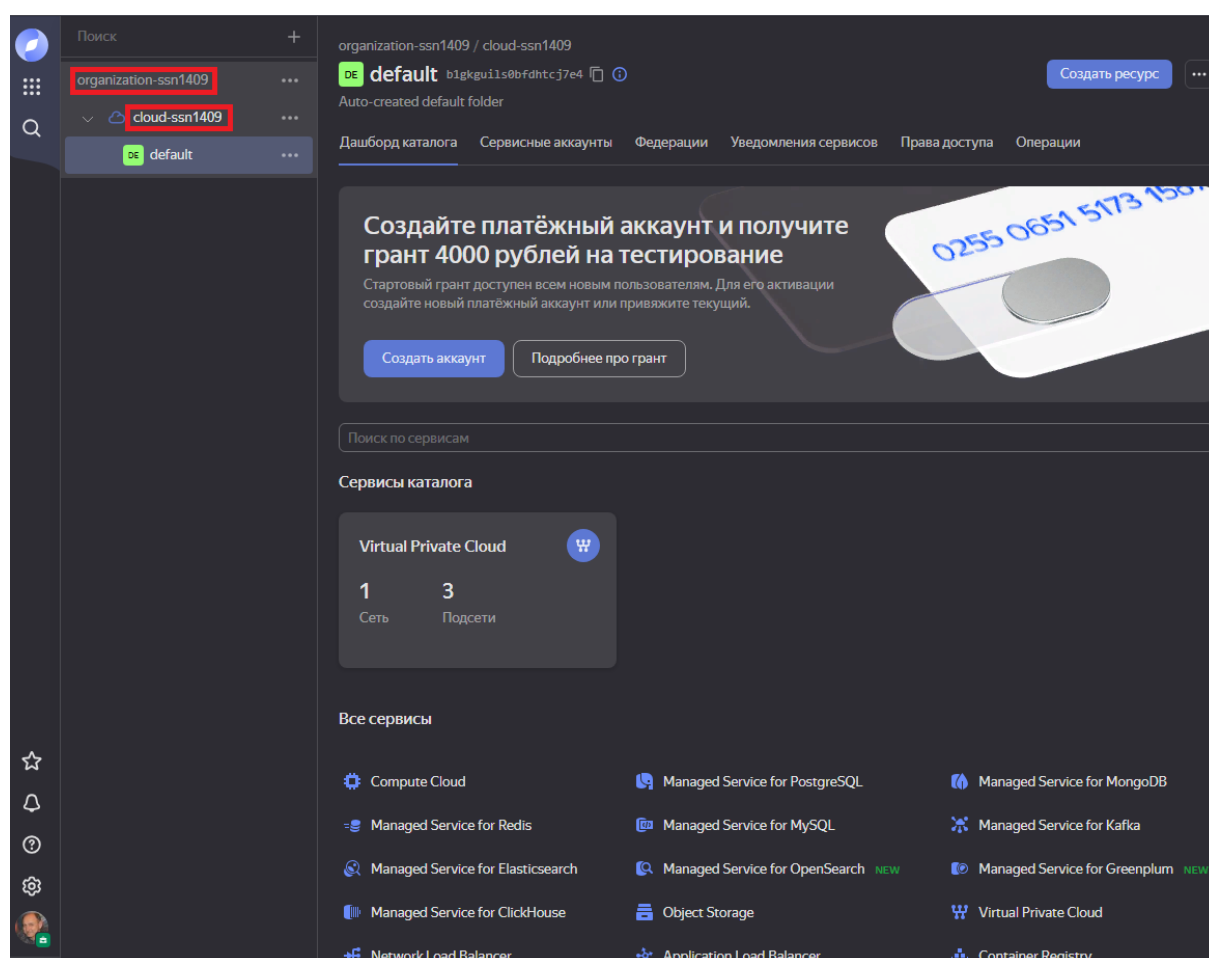


Рис. 1.4 Окно портала «Яндекс.Облако» на вкладке «Дашборд каталога»

Шаг 3. В окне, показанном на рис. 1.5, нужно ввести требуемую информацию. В имени аккаунта должна быть ваша фамилия. При добавлении банковской карты на ней должна быть минимальная сумма 11 руб. Данные карты на рисунке в отчете не нужно показывать.

Создание платежного аккаунта [X]

Расчёты будут вестись в рублях. При выборе оплаты банковской картой используйте карты, выпущенные российскими банками.

Страна* ? Россия [v]

Имя аккаунта* teacher-sazonov

Тип плательщика

Физическое лицо | Юридическое лицо или ИП

Данные плательщика

Фамилия* Сазонов

Имя* Сергей

Отчество* Николаевич

Почтовый адрес г. Ульяновск, [redacted]

Банковская карта ?

MIR 2200 [redacted]

Добавить карту

Контактные данные

Электронная почта* ssn1409@yandex.ru [edit]
На нее мы будем отправлять отчеты об использовании сервисов Yandex Cloud и важные уведомления о платежном аккаунте

Телефон* 790 [redacted] [edit]

Рис. 1.5 Окно создания платежного аккаунта

Шаг 4. После правильного ввода всех данных и нажатия кнопки «Создать» должно появиться окно, как на рис. 1.6, в котором статус платежного аккаунта должен быть «Active».

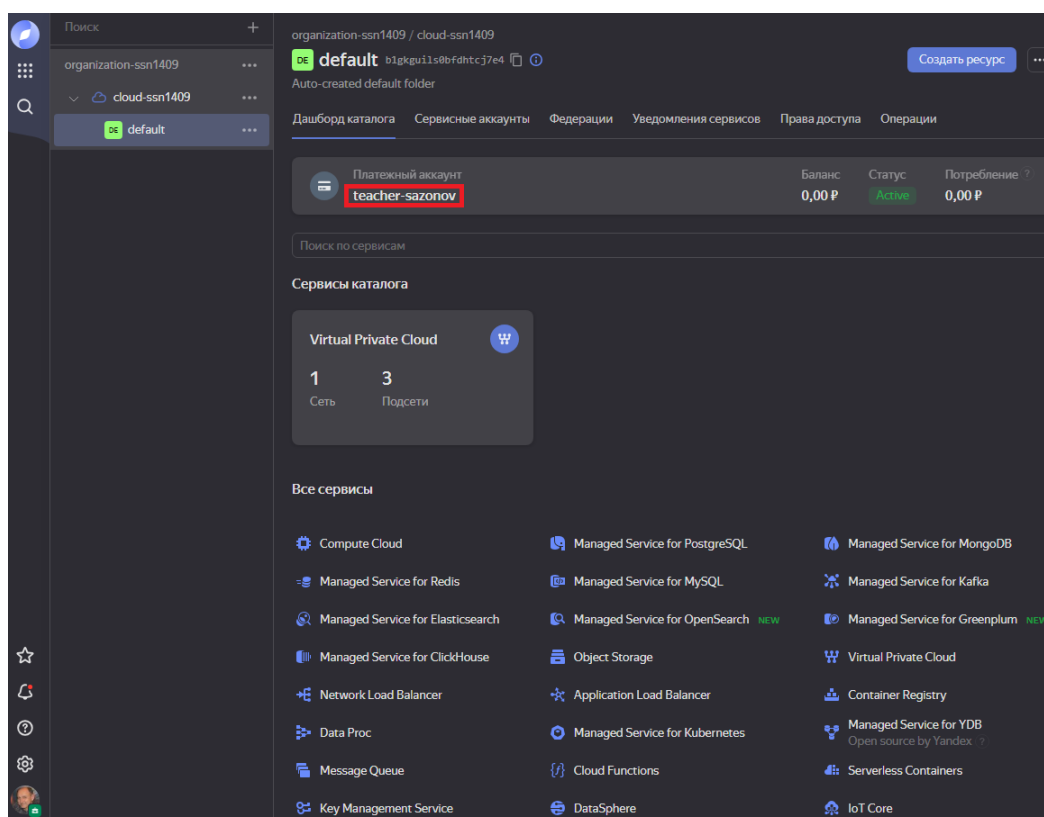


Рис. 1.6 Информация о статусе платежного аккаунта

Шаг 5. При нажатии кнопки «Сообщения» должно появиться окно, как на рис. 1.7, в котором сообщается о завершении регистрации.

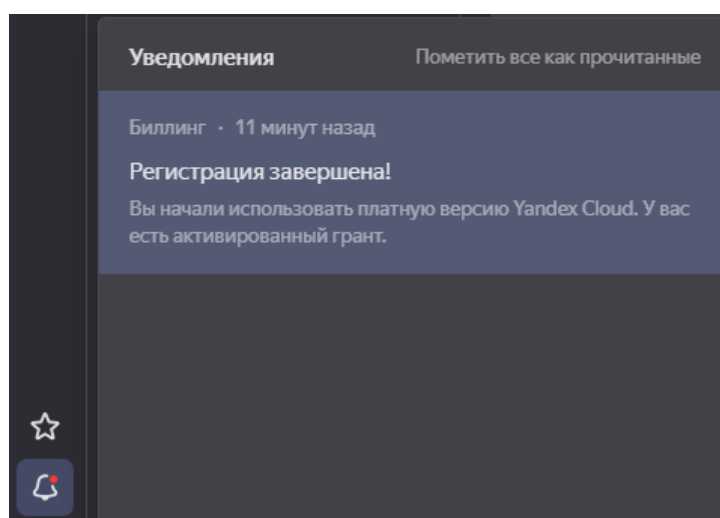


Рис. 1.7 Окно с информацией о завершении регистрации

Шаг 6. Далее необходимо убедиться в активном статусе созданных сетей и подсетей. При нажатии кнопки «Сеть» должно появиться окно, показанное на рисунке 1.8, в котором приводится информация о статусе и параметрах созданной виртуальной сети.

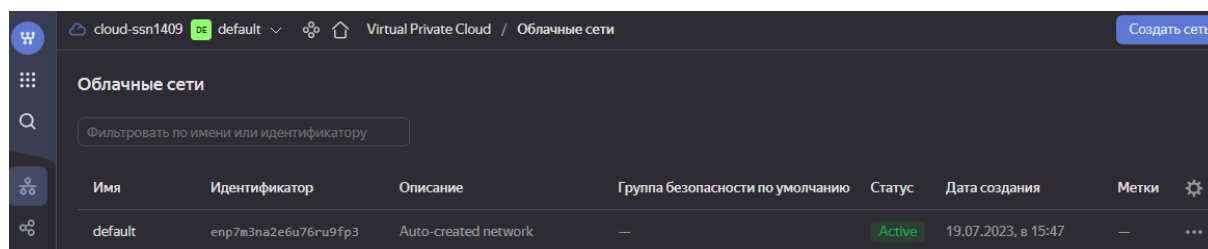


Рис. 1.8 Окно с информацией о созданной виртуальной сети

Шаг 7. Аналогично при нажатии кнопки «Подсети» должно появиться окно, показанное на рисунке 1.9, в котором приводится информация о статусе и параметрах созданных виртуальных подсетей.

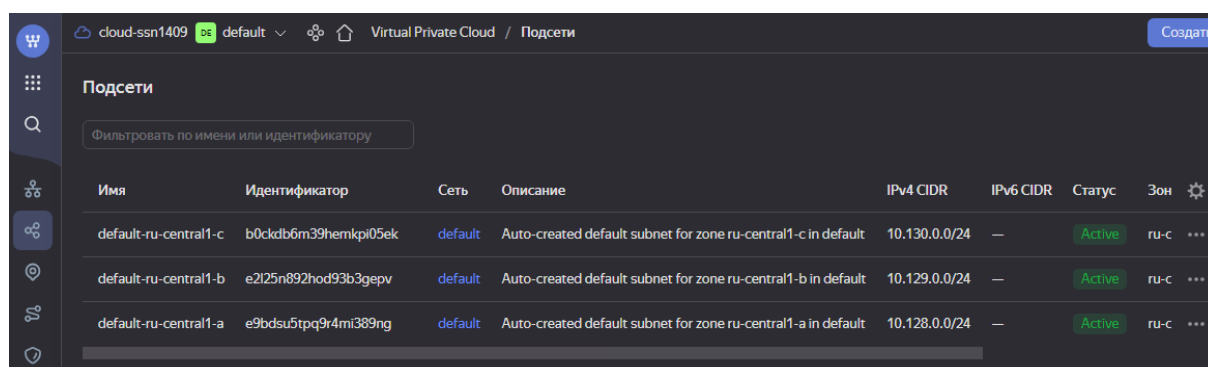


Рис. 1.9 Окно с информацией о созданных виртуальных подсетях

На этом этапе выполнение лабораторной работы №1 «Регистрация аккаунта на портале» считается завершенной.

Содержание отчета:

1. Титульный лист с названием работы и данными исполнителя.
2. Рисунки (скрины экранов) всех шагов выполнения работы.
3. Выводы по проделанной лабораторной работе.

Раздел 2. Виртуальные машины

Упрощая определение виртуальной машины (ВМ), данное ранее можно сказать, что ВМ – это программа, которая имитирует работу другой программы, то есть, по сути, она имитатор компьютера внутри реального физического компьютера.

Понятие виртуализации применяют по отношению к системам и процессам. В первом случае ВМ эмулируют операционные системы (ОС), хранилища данных, сервера или компьютерные сети. Во втором случае ВМ нужна для правильной работы приложения.

Достоинства ВМ:

1. Возможности ВМ не зависят от хоста, а определяются возможностями виртуальной гостевой системы. Например, ВМ с ОС Windows позволяет запускать программы на компьютерах с ОС Linux.
2. ВМ могут эмулировать работу устаревших ОС, позволяя избежать необходимости техподдержки устаревшего оборудования.
3. На одном многозадачном серверном хосте одновременно могут быть запущены несколько ВМ для работы разных сервисов.
4. ВМ легко архивировать, переносить и настраивать. Физически ВМ представляет собой файл-образ, хранимый на любом носителе.

Недостатки ВМ:

1. Каждая ВМ потребляет ресурсы хоста. Несколько ВМ на одном сервере приводят к снижению его производительности, поэтому повышаются требования к выбору хостов.
2. Производительность ВМ ниже производительности реального ПК с теми же характеристиками, так как дополнительно расходуются ресурсы хоста на поддержание работы среды виртуализации.

В связи с этим типовое использование ВМ – предоставление сервисных услуг клиентам при оптимальной загрузке сервера. Например, на одном сервере одновременно могут работать почтовый сервер, хранилище данных, СУБД и система тестирования стороннего ПО.

ВМ настраивают с помощью гипервизоров. Упрощенное сравнение традиционной аппаратно-программной архитектуры и виртуального комплекса представлено на рисунке 2.1.

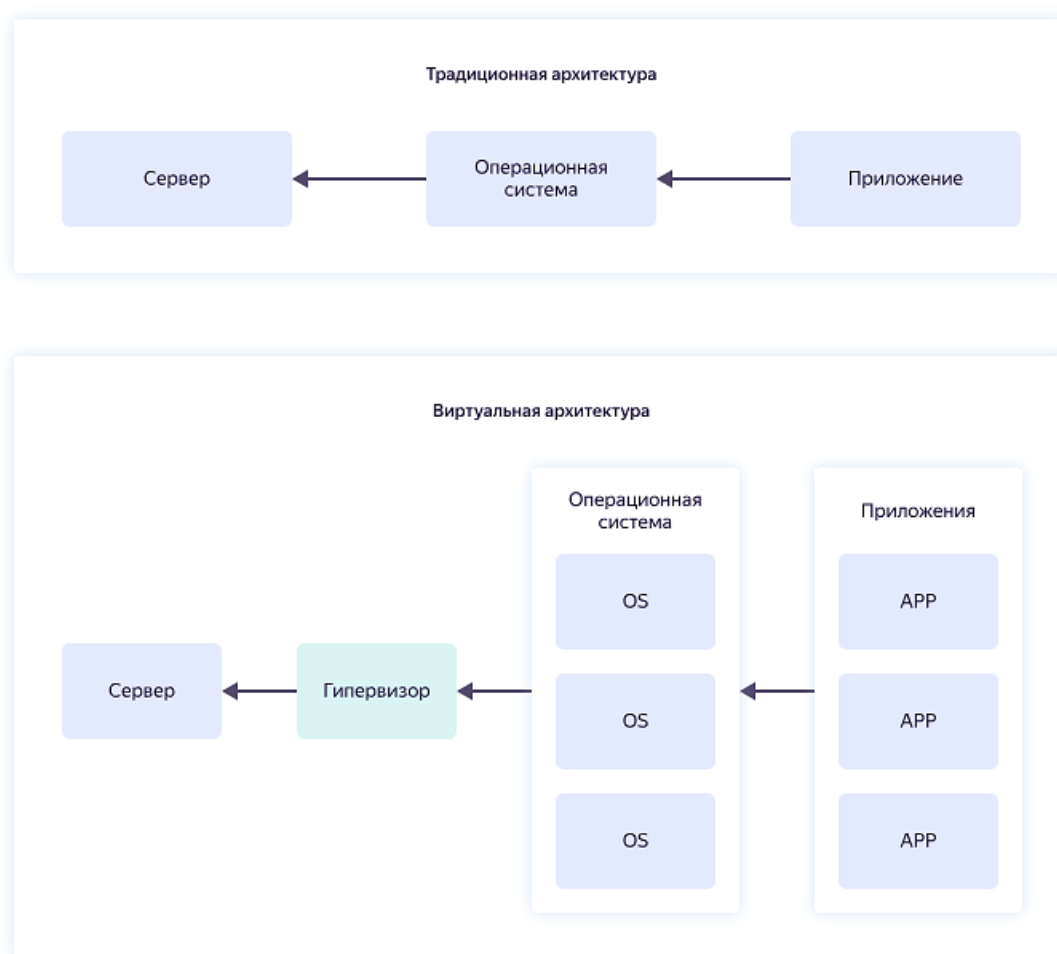


Рис. 2.1 Сравнение традиционной и виртуальной архитектур

Гипервизор отвечает не только за создание, но и за управление ВМ: ее запуск и остановку, распределение доступа к аппаратным и программным ресурсам хоста, архивирование и быстрое развертывание. Кроме организации доступа клиентов к ВМ, гипервизор выполняет функции защиты хоста от отклонений в режимах работы на самих ВМ (аналогично прокси-серверу для доступа к ресурсам сети).

Этапы настройки ВМ зависят от аппаратного и программного обеспечения хоста, типа и производителя гипервизора и определяются при планировании структуры виртуальной системы.

Общие рекомендации по настройке:

- Заранее выберите и настройте систему хранения данных (выбор контроллеров дисков, типов интерфейса, объема, времени чтения и записи, а также типа RAID-массива).
- Выделите больше ресурсов для работы ВМ (память «с запасом», а также разрешение ВМ отключать ее при отсутствии необходимости использования ускоряет работу всех ВМ хоста).
- Уменьшите количество фоновых программ (использование хоста в качестве сервера негативно сказывается на производительности системы в целом, например, отключение сканирования хоста антивирусом дает значительный прирост производительности).
- Используйте различные инструменты мониторинга и сбора метрик для управления производительностью ВМ.

Эти меры позволят эффективно планировать выделяемые ВМ ресурсы и своевременно выявлять ВМ, имеющие регулярные проблемы с производительностью. Настройка и оптимизация работы ВМ – трудоемкий процесс, требующий специалистов высокой квалификации.

Особенности ВМ от Yandex Cloud

Очевидное преимущество облачной ВМ – это удобство настройки и работы с ней, благодаря использованию упрощенного интуитивно-понятного интерфейса. Примером подобной системы является сервис *Yandex Compute Cloud* – облачная платформа, использующая инфраструктуру и уникальные технологии Яндекса.

Этот сервис позволяет создавать ВМ и отказоустойчивые группы ВМ в инфраструктуре Yandex Cloud, защищенной в соответствии с Федеральным законом РФ «О персональных данных» №152-ФЗ.

ВМ сервиса *Yandex Compute Cloud* поддерживают работу со снимками дисков пользователя для быстрого развертывания системы. Кроме того, в *Yandex Cloud Marketplace* доступны образы дисков с самыми распространенными ОС семейства Linux.

2.1 Подготовка к созданию ВМ

ЛР2. *Цель:* Создать в облачной платформе «Яндекс.Облако» ВМ и рассмотреть основные моменты работы с ней.

Пример выполнения лабораторной работы №2 также описан с помощью пронумерованных шагов. При этом требования к каждому шагу аналогичны требованиям лабораторной работы №1.

Задание. Выполните самостоятельно все шаги аналогично приведенному примеру. В качестве варианта задания на рисунках также должна присутствовать *персональная информация* о студенте, которая в примере также выделена красной рамкой.

Шаг 1. В документации от Яндекса по созданию ВМ в облаке приводятся рекомендации организации доступа к ней с помощью предварительно созданного SSH-ключа. Поэтому вначале целесообразно рассмотреть процесс создания этого ключа в рекомендованном Яндексом программном комплексе PuTTY. Скачайте с официального сайта файл с расширением ***.msi**. Установите этот комплекс на свой ПК. При этом вы получите приложения, показанные на рис. 2.2.

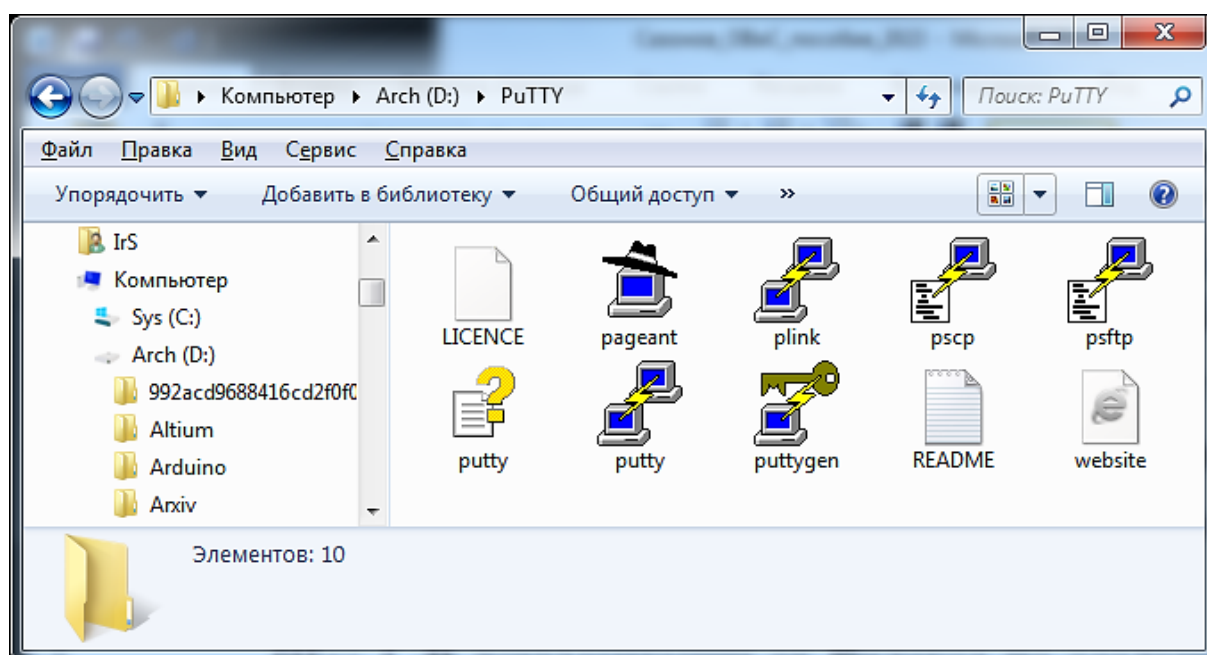


Рис. 2.2 Состав программного комплекса PuTTY

Шаг 2. Убедитесь, что директория PuTTY прописана в системной переменной PATH (**Компьютер → Свойства → Дополнительные параметры системы → Переменные среды → Системные переменные → PATH**). Если она там не прописана, то необходимо добавить.

Шаг 3. Запустите приложение **puttygen**, выберите в качестве генерируемой пары **EdDSA** и нажмите **Generate** (см. рис. 2.3).

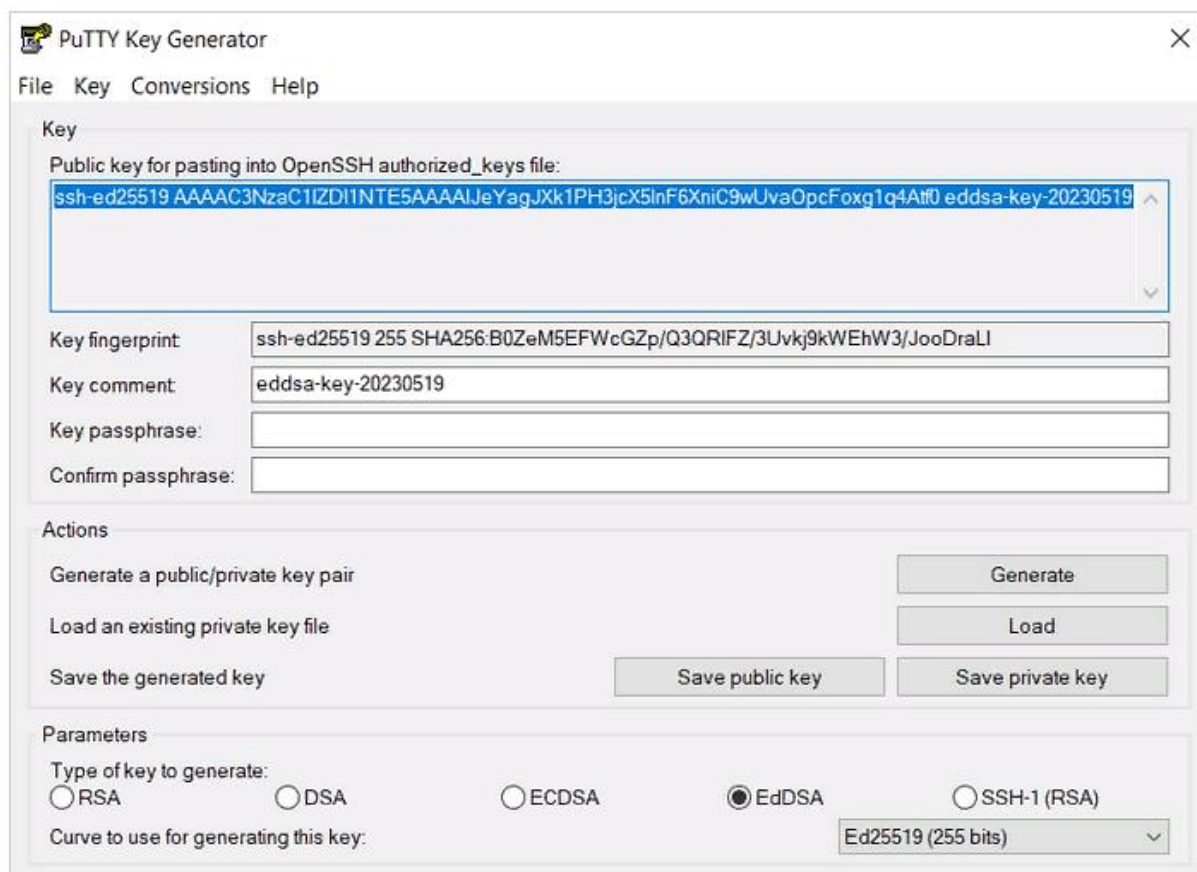


Рис. 2.3 Генерация SSH-ключа

В поле **Key passphrase** необходимо ввести пароль для ключа и повторно ввести его в поле ниже. Далее нажмите кнопку **Save private key** и сохраните закрытый ключ. Этот ключ не рекомендуется кому-либо передавать. Кроме того рекомендуется сохранить ключ в текстовом файле. Для этого скопируйте открытый ключ *одной строкой* из текстового поля в текстовый файл с названием **id_ed25519.pub**.

2.2 Создание VM в «Яндекс.Облаке»

Шаг 4. В консоли управления выберите каталог для VM (можно оставить по умолчанию), а в списке сервисов выбрать **Compute Cloud**. После этого откроется окно, пример которого показан на рис. 2.4.

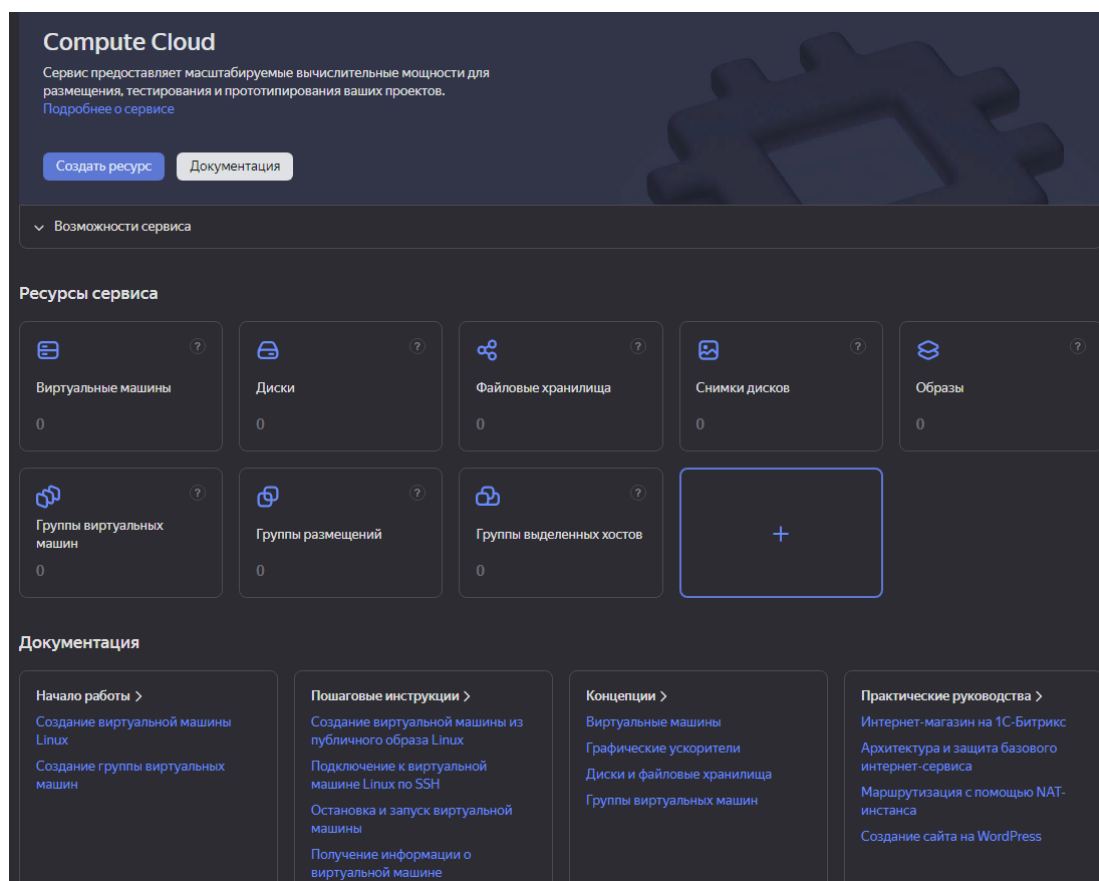


Рис. 2.4 Пример окна **Compute Cloud**

Выберите ресурс «Виртуальные машины», как показано на рис.2.5.

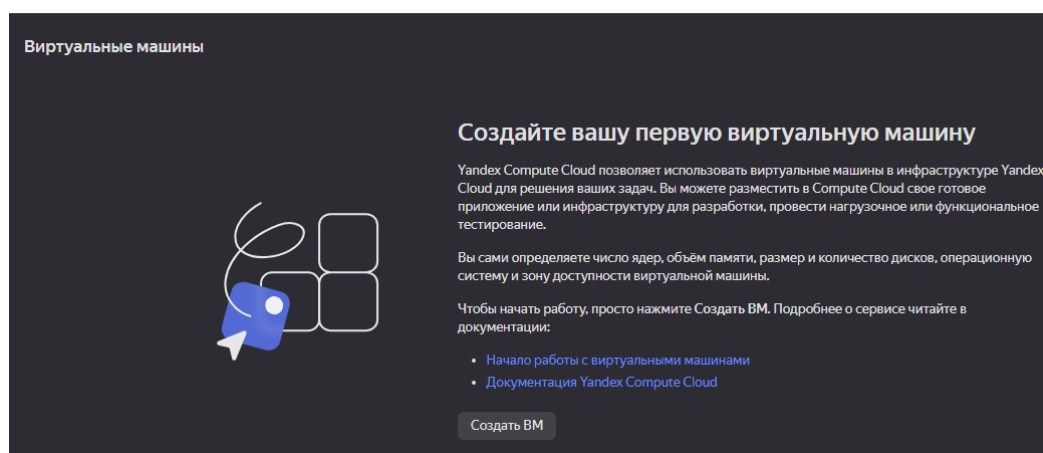


Рис. 2.5 Окно ресурса «Виртуальные машины»

Шаг 5. Нажатие на кнопку **Создать ВМ** дает окно параметров, первая половина которого показана на рис.2.6.

Создание виртуальной машины

Базовые параметры

Имя ?

Описание ?

Зона доступности ?

Выбор образа/загрузочного диска

Операционные системы Container Solution Cloud Marketplace Пользовательские

Фильтр по операционной системе

Ubuntu	22.04	
CentOS	7	
Debian	11	
Fedora	35	
openSUSE Leap	15.3	
Fedora CoreOS	35	

[Показать все продукты](#)

Диски и файловые хранилища

Диски 1 Файловые хранилища

Имя диска	Тип	Размер
openSUSE Leap 15.3	Загрузочный	HDD
		15 ГБ

5 ГБ 8192 ГБ

605,40 ₽ в месяц ?

[Тарифы и цены](#)

- Intel Ice Lake. 50% vCPU — прерываемые ВМ 288,00 ₽
- Intel Ice Lake. RAM — прерываемые ВМ 100,80 ₽
- Публичный IP-адрес 172,80 ₽
- Стандартное сетевое хранилище (HDD) 43,80 ₽

Сэкономьте до 22%

Зарезервируйте ресурсы для Compute на полгода или год и платите меньше

[Подробнее](#)

Рис. 2.6 Первая половина окна параметров ВМ

В блоке **Базовые параметры** введите имя и описание ВМ. Длина имени должна быть от 3 до 63 символов и может содержать строчные буквы латинского алфавита, цифры и дефисы. Первый символ имени должен быть буквой, а последний символ не должен быть дефисом. Зону доступности можно оставить по умолчанию или выбрать.

В блоке **Выбор образа/загрузочного диска** выберите один из образов и версию ОС на базе Linux. В блоке **Диски и файловые хранилища** на вкладке **Диски** настройте загрузочный диск (выберите тип и нужный размер). Файловое хранилище в данной работе не настраивается.

The screenshot displays the configuration interface for a virtual machine, specifically the second half of the window. It includes sections for network settings, security groups, and access. On the right, there is a pricing summary and a promotional offer.

Гарантированная доля vCPU: 20%, 50%, 100%. Для несложных приложений, которые не требуют постоянно 100% vCPU.

RAM: 2 ГБ (слайдер от 1 ГБ до 8 ГБ).

Дополнительно: ☒ Прерываемая.

Сетевые настройки

Подсеть: default / default-ru-central1-a

Публичный адрес: Автоматически, Список, Без адреса.

Дополнительно: ☐ Защита от DDoS-атак.

Внутренний IPv4-адрес: Автоматически, Вручную.

Настройки DNS для внутренних адресов: [выпадающий список]

Группы безопасности: Не выбрано. В указанной сети не найдена ни одна группа безопасности.

Доступ

Сервисный аккаунт: Создать аккаунт.

Логин: [маскированный текст]

SSH-ключ: ssh-ed25519 [маскированный текст]

Дополнительно: ☒ Разрешить доступ к серийной консоли.

Прайс-сумма: 605,40 ₽ в месяц. Тарифы и цены.

- Intel Ice Lake. 50% vCPU — прерываемые VM: 288,00 ₽
- Intel Ice Lake. RAM — прерываемые VM: 100,80 ₽
- Публичный IP-адрес: 172,80 ₽
- Стандартное сетевое хранилище (HDD): 43,80 ₽

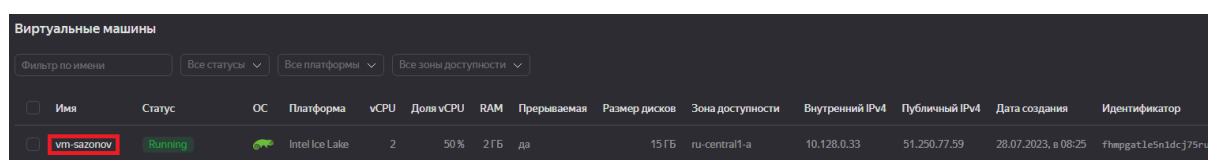
Промо-баннер: Сэкономьте до 22%. Зарезервируйте ресурсы для Compute на полгода или год и платите меньше. Подробнее.

Рис. 2.7 Вторая половина окна параметров VM

В блоке **Вычислительные ресурсы** выбирают одну из трех платформ (можно оставить по умолчанию), гарантированную долю загрузки процессора (рекомендуется выбрать 50% для экономии средств) и

объем ОЗУ (по умолчанию 2 ГБ). Дополнительно можно сделать VM прерываемой (рекомендуется для экономии средств). В блоке **Сетевые настройки** можно все оставить по умолчанию. В блоке **Доступ** указываются данные для доступа на VM. Сервисный аккаунт в данной работе создавать не требуется. В поле **Логин** введите свою фамилию на латинице. В поле **SSH-ключ** вставьте содержимое файла открытого ключа, который предварительно был создан в пункте 2.1. Разрешать доступ к серийной консоли не обязательно, но потребуется для настройки, если получить доступ описываемым методом не удалось. Остальные параметры VM оставьте по умолчанию. После указания всех параметров VM необходимо нажать на кнопку **Создать VM**.

Шаг 6. Далее нужно подождать некоторое время, пока не появится окно с информацией о статусе VM, которое изображено на рис. 2.8.



Имя	Статус	ОС	Платформа	vCPU	Доля vCPU	RAM	Прерываемая	Размер дисков	Зона доступности	Внутренний IPv4	Публичный IPv4	Дата создания	Идентификатор
vm-sazonov	Running		Intel Ice Lake	2	50 %	2 ГБ	да	15 ГБ	ru-central1-a	10.128.0.33	51.250.77.59	28.07.2023, в 08:25	fhpqgatl5n1dcj75ru8

Рис. 2.8 Окно статуса созданной VM

Если выделить созданную VM, то внизу окна появится строка (см. рисунок 2.9), предлагающая совершить возможные действия над VM.

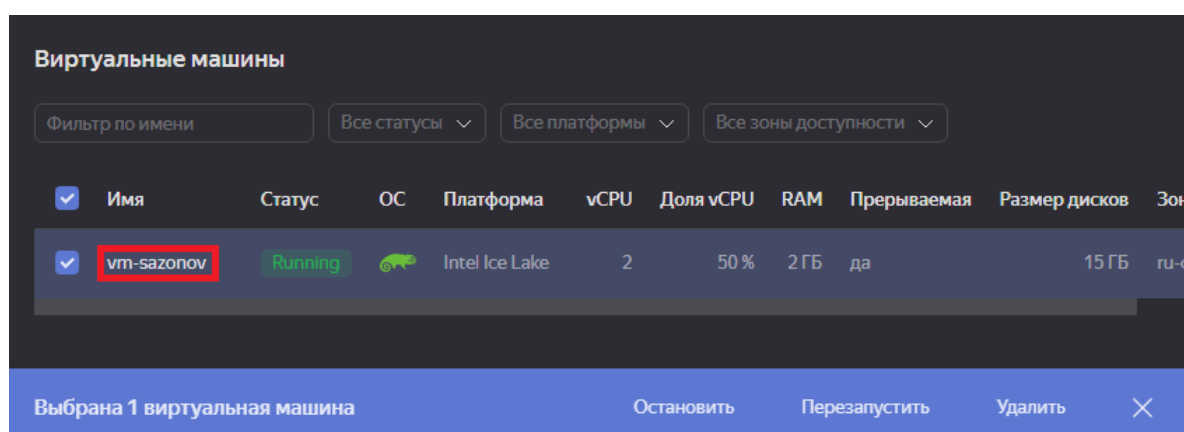


Рис. 2.9 Окно со строкой действий над созданной VM

После завершения работы с VM не забывайте ее остановить для экономии выделенных вам средств за счет стартового гранта.

В левой части окна расположено подокно с инструментами. Если, например, щелкнуть «Обзор», то появится окно, как на рисунке 2.10.

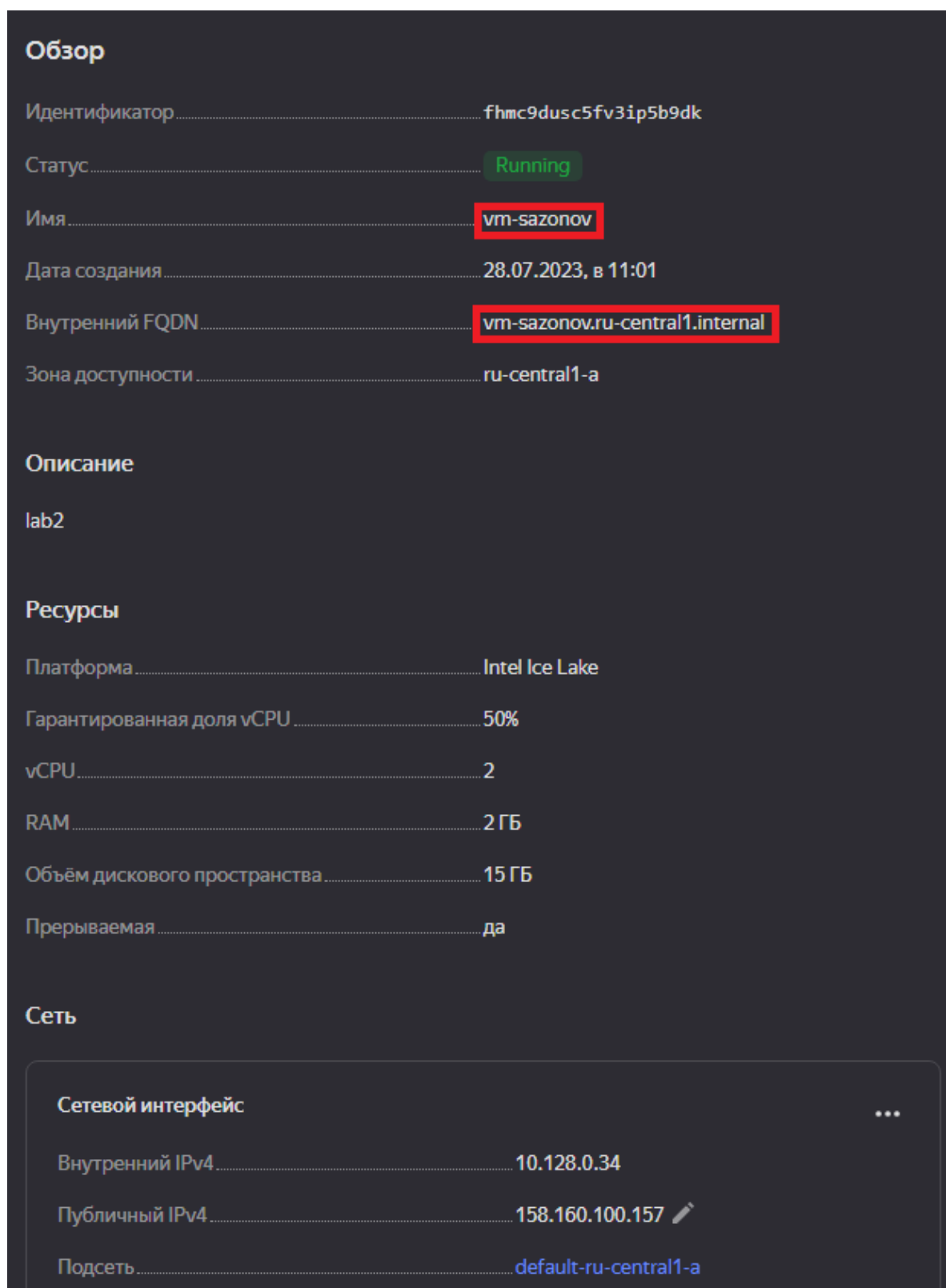


Рис. 2.10 Первая часть окна «Обзор» созданной VM

Вторая часть окна «Обзор» представлена на рисунке 2.11. В целях безопасности информация о ключах на данном рисунке отсутствует. С этой же целью рекомендуется всем студентам в отчете по данной лабораторной работе эту информацию также не приводить.

The screenshot displays the 'Overview' window for a virtual machine, organized into several sections:

- Настройки DNS для внутренних адресов**: A table with columns 'Зона', 'FQDN', and 'TTL'. The table is currently empty, showing 'Нет данных'.
- Резервное копирование**: A toggle for 'Cloud Backup' is shown as 'Не подключён'.
- Мониторинг**: A toggle for 'Агент для поставки метрик' is shown as 'нет'.
- Дополнительно**: A toggle for 'Доступ к серийной консоли' is shown as 'разрешен'.
- Метаданные**: A section with a dropdown arrow, containing several configuration items:
 - serial-port-enable**: Set to '1'.
 - ssh-keys**: A text area containing a redacted SSH key.
 - user-data**: A text area containing a cloud-config script:

```
#cloud-config
datasource:
  Ec2:
    strict_id: false
ssh_pwauth: no
users:
- name: [redacted]
  sudo: ALL=(ALL) NOPASSWD:ALL
  shell: /bin/bash
ssh_authorized_keys:
- [redacted]
```
 - install-unified-agent**: Set to '0'.

Рис. 2.11 Вторая часть окна «Обзор» созданной VM

Инструмент «Мониторинг» предназначен для получения информации об интенсивности использования различных созданных средств. Окно «Мониторинг» содержит информацию о загрузке процессора, дисков

и созданных виртуальных сетях. Информация (см. рисунок 2.12) представлена в режиме реального времени с изменяемым масштабом.

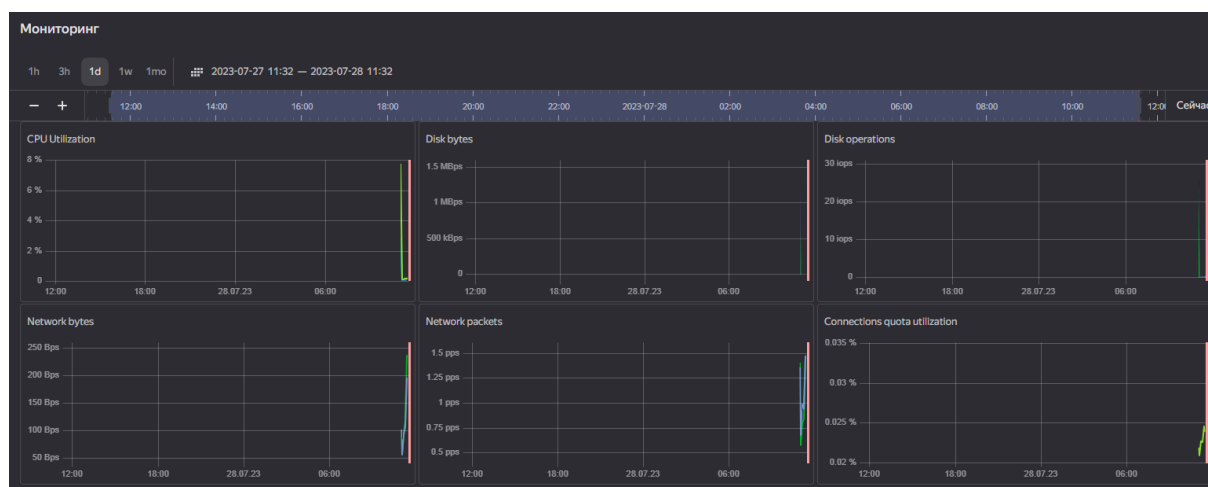


Рис. 2.12 Окно «Мониторинг» созданной VM

2.3 Запуск и работа с VM

Выполним подключение к VM, находящейся в статусе **Running**, по протоколу **SSH**, используя программу PuTTY. Для подключения необходимо указать публичный IP-адрес VM. Этот публичный адрес можно определить через консоль управления. Он располагается в поле **Публичный IPv4** блока **Сеть** на странице VM.

Шаг 7. Запуск приложения Pageant.

- Нажмите ПКМ на значок Pageant на панели задач.
- В контекстном меню выберите пункт **Add key**.
- Выберите сгенерированный PuTTY приватный ключ в формате **.ppk**. Если для ключа задан пароль, введите его.

Шаг 8. Запуск приложения PuTTY.

- В поле **Host Name (or IP address)** введите публичный IP-адрес VM, к которой вы хотите подключиться.
- Укажите порт **22** и тип соединения **SSH**.

Окно конфигурации программы PuTTY после выполнения шагов 7 и 8 показано на рисунке 2.13.

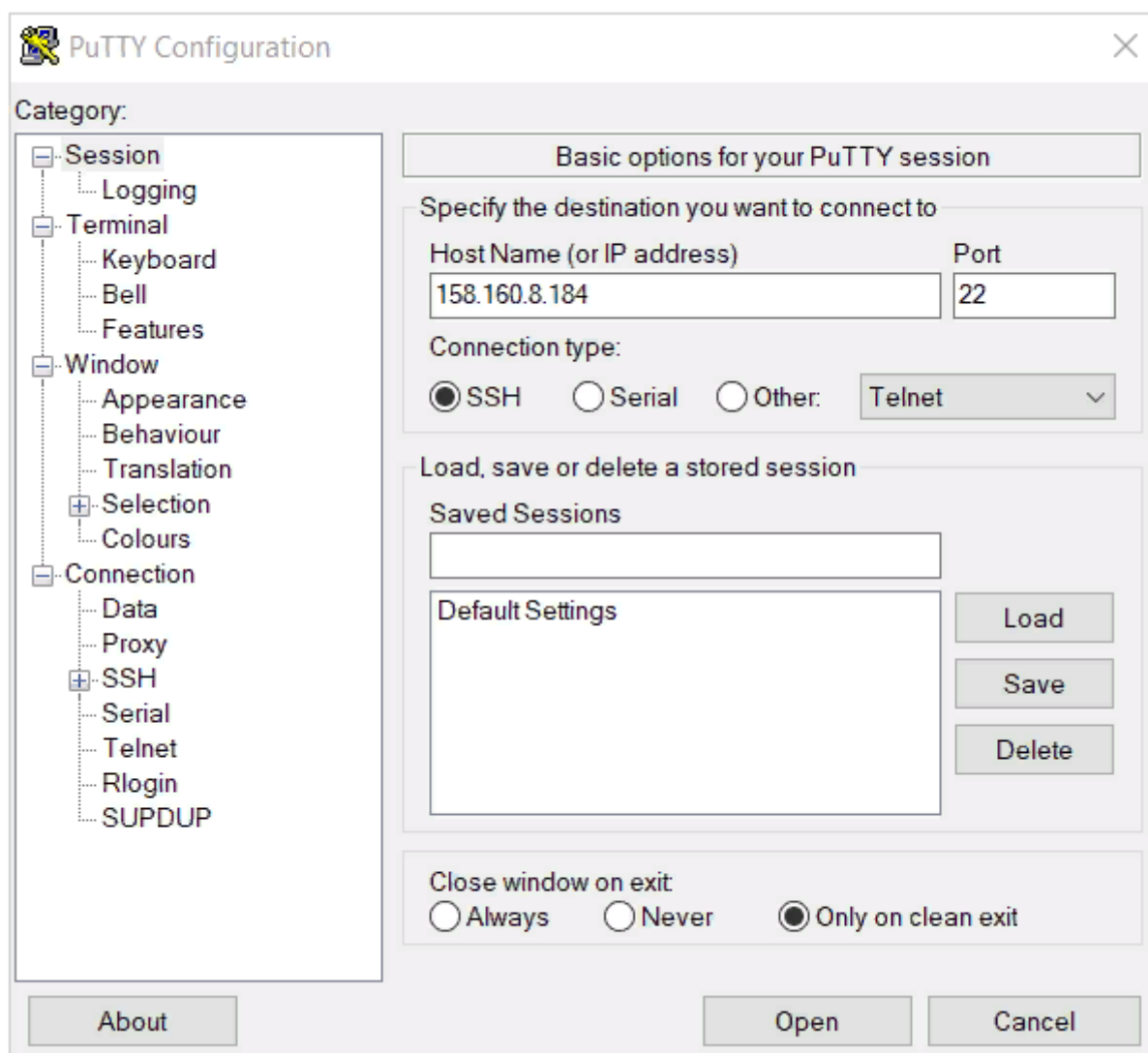


Рис. 2.13 Окно конфигурации PuTTY после выполнения шагов 7 и 8

Шаг 9. Далее необходимо открыть в дереве **Category**, которое расположено слева, пункт **Connection** → **SSH** → **Auth** и установить флаг **Allow agent forwarding**. Окно этого пункта после выполнения данной операции показано на рисунке 2.14.

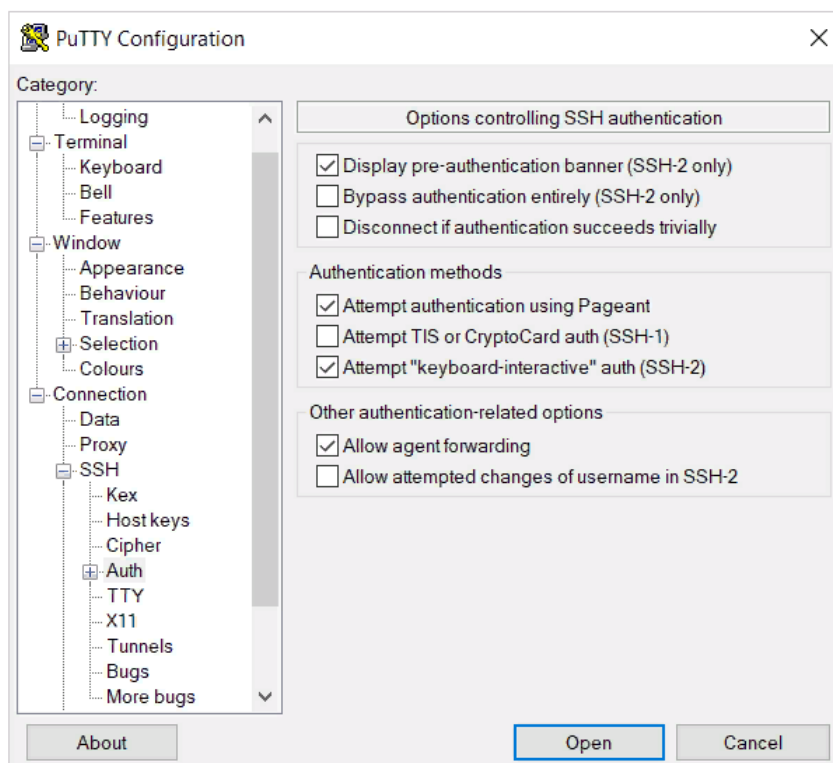


Рис. 2.14 Окно конфигурации PuTTY после выполнения шага 9

Шаг 10. Откройте пункт **Connection** → **SSH** → **Auth** → **Credentials** и в поле **Private key file for authentication** выберите файл **.ppk**.

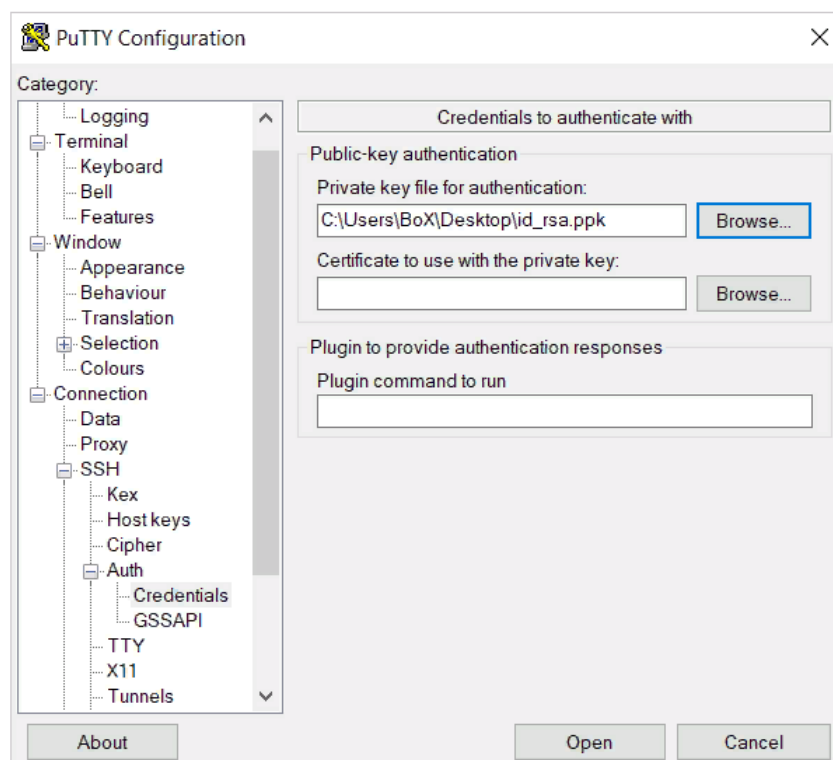


Рис. 2.15 Окно конфигурации PuTTY после выполнения шага 10

Шаг 11. Вернитесь в меню **Sessions**. В поле **Saved Sessions** введите имя сессии и нажмите **Save** (см. рис. 2.16). Этот профиль сессии можно использовать для подключения с помощью Pageant.

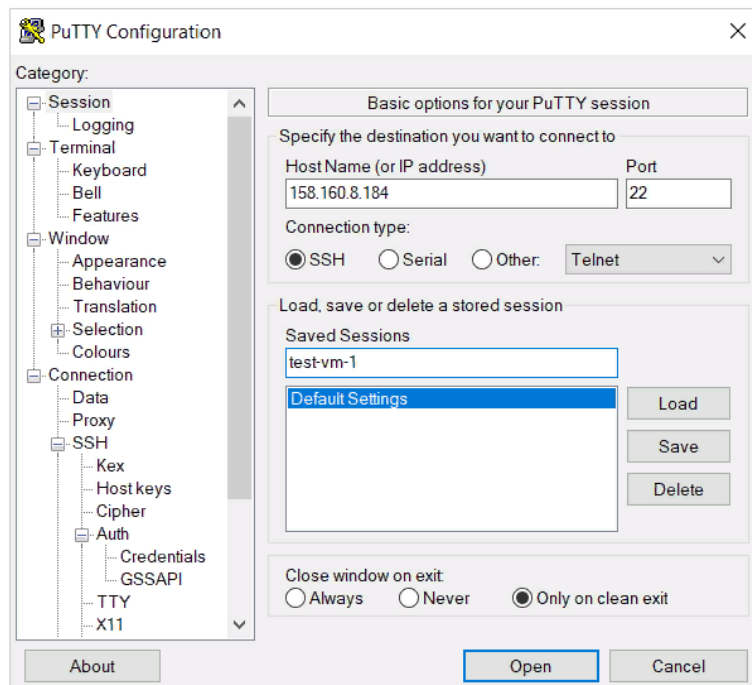


Рис. 2.16 Окно конфигурации PuTTY после выполнения шага 11

Шаг 12. Нажмите кнопку **Open**. При первом подключении к ВМ появляется сообщение о неизвестном хосте, как показано на рис. 2.17.

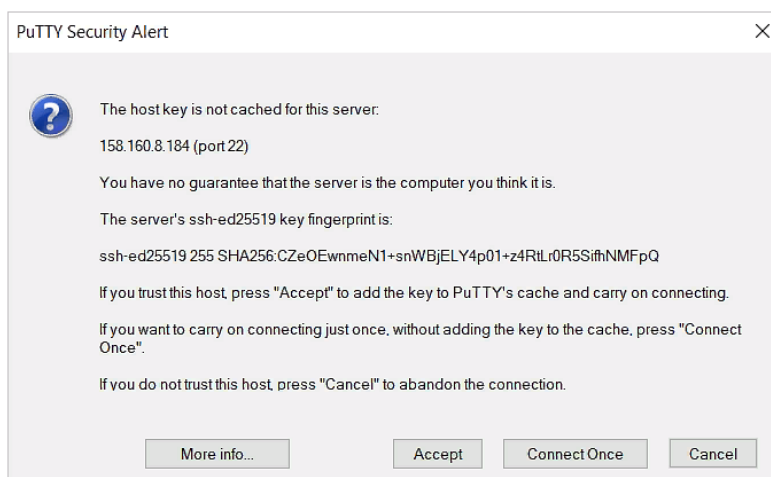
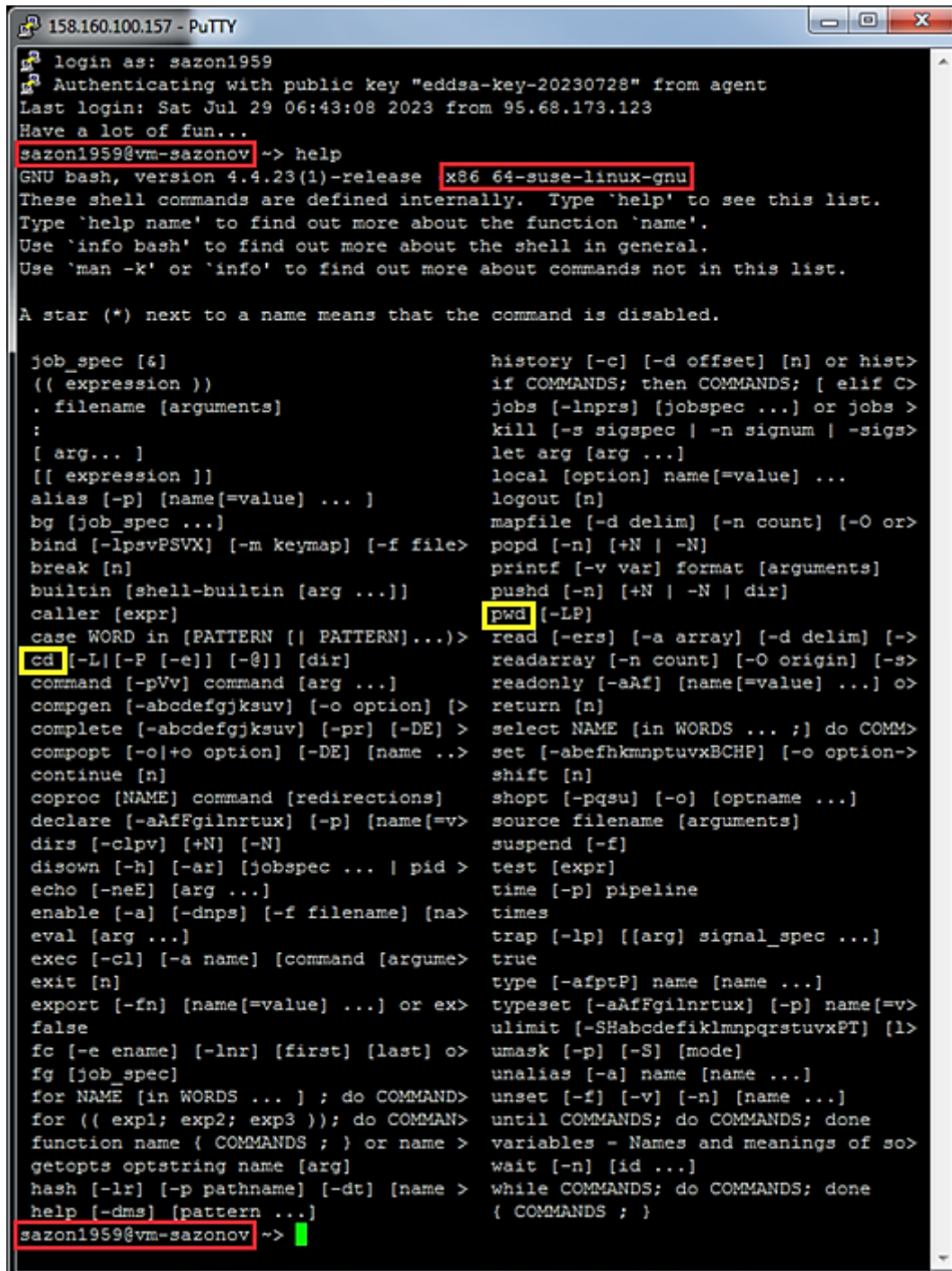


Рис. 2.17 Окно сообщения о неизвестном хосте

Нажмите кнопку **Accept**. В окне терминала введите логин пользователя и нажмите **Enter**. Если настройка была осуществлена корректно, то

будет установлено соединение с сервером. Проверьте правильность работы ВМ вводом 3-4 команд Linux, как показано на рисунке 2.18.



```
158.160.100.157 - PuTTY
login as: sazon1959
Authenticating with public key "eddsa-key-20230728" from agent
Last login: Sat Jul 29 06:43:08 2023 from 95.68.173.123
Have a lot of fun...
sazon1959@vm-sazonov ~-> help
GNU bash, version 4.4.23(1)-release x86_64-suse-linux-gnu
These shell commands are defined internally. Type 'help' to see this list.
Type 'help name' to find out more about the function 'name'.
Use 'info bash' to find out more about the shell in general.
Use 'man -k' or 'info' to find out more about commands not in this list.

A star (*) next to a name means that the command is disabled.

job_spec [&]
(( expression ))
. filename [arguments]
:
[ arg... ]
[[ expression ]]
alias [-p] [name[=value] ... ]
bg [job_spec ...]
bind [-lpsvPSVX] [-m keymap] [-f file]
break [n]
builtin [shell-builtin [arg ...]]
caller [expr]
case WORD in [PATTERN] [PATTERN]...)
cd [-L] [-P] [-e] [-@] [dir]
command [-pVv] command [arg ...]
compgen [-abcefgjksuv] [-o option]
complete [-abcefgjksuv] [-pr] [-DE]
compropt [-o] [+o option] [-DE] [name ...]
continue [n]
coproc [NAME] command [redirections]
declare [-aAfFgIlNrtux] [-p] [name[=v]
dirs [-clpv] [+N] [-N]
disown [-h] [-ar] [jobspec ... | pid]
echo [-neE] [arg ...]
enable [-a] [-dnps] [-f filename] [na]
eval [arg ...]
exec [-cl] [-a name] [command [argume]
exit [n]
export [-fn] [name[=value] ...] or ex
false
fc [-e ename] [-lnr] [first] [last] o
fg [job_spec]
for NAME [in WORDS ... ] ; do COMMAND
for (( exp1; exp2; exp3 )); do COMMAND
function name { COMMANDS ; } or name >
getopts optstring name [arg]
hash [-lr] [-p pathname] [-dt] [name >
help [-dms] [pattern ...]
sazon1959@vm-sazonov ~->
```

Рис. 2.18 Проверка правильности работы ВМ

Покажите текущую директорию командой **pwd**. Перейдите в корневой каталог и выполните две команды **ls** и **lspci**. Отчет должен содержать пункты аналогичные лабораторной работе №1.

Раздел 3. Базы данных

Очень многие приложения работают с большими объемами данных, причем большинство из них занимается обработкой этих данных без математических расчетов. По экспертным оценкам около 90% разработки приложений связаны с обработкой данных, которые хранятся в базах данных (БД), поэтому системы управления базами данных (СУБД) являются основой очень многих приложений.

Данные могут быть любыми, поэтому способ их обработки будет зависеть от их природы. Иногда данные представляются только одним значением, например числом π , которое используется в программе, рисующей окружности и сферы. Это значение задается в самой программе, как отношение длины окружности к ее диаметру. Подобные данные называются *константами*, так как они не изменяются.

В качестве примера меняющихся данных рассмотрим некоторые наиболее востребованные курсы валют ЦБ РФ на 05.08.2023 по отношению к российскому рублю, представленные в таблице 3.1. В мировой практике принято, чтобы величины курсов валют были зафиксированы с точностью не менее шести значащих цифр.

Таблица 3.1

Название валюты	Код	Единиц	Курс ЦБ РФ
Доллар США	USD	1	94.8076
Евро	EUR	1	103.8379
Белорусский рубль	BYN	1	30.4270
Дирхам (ОАЭ)	AED	1	25.8120
Индийская рупия	INR	10	11.5273
Китайский юань	CNY	1	13.1755
Турецкая лира	TRY	10	35.1476
Узбекский сум	UZS	10000	81.3954
Фунт стерлингов	GBP	1	120.3772
Японская иена	JPY	100	66.5083

Если нужно создать приложение для конвертации валют, то оно должно будет использовать таблицу, в которой содержались бы названия валют, валютный курс и количество национальных единиц валюты в одном рубле. Валютные курсы постоянно изменяются, и количество строк таблицы валют также может изменяться. Например, новые страны могут заключать договор с ЦБ РФ или имеющиеся страны расторгать договор. Все это должно фиксироваться в таблице.

То есть получается, что приложение конвертор валют требуется изменять после каждого изменения встроенной таблицы. Удобнее сделать так, чтобы приложение читало файл, содержащий некоторую простую информацию о валютах, например название, обозначение и курс обмена. Тогда можно будет при необходимости изменения таблицы модифицировать только файл, не изменяя само приложение.

Файл с данными таблицы 3.1 не имеет какой-то особой структуры; это просто несколько строк текста, которые имеют значение только для конкретного приложения, читающего этот файл. Поскольку такому файлу не свойственна определенная структура, в технической литературе его принято называть «плоским» (**flat**).

Многие системы и приложения и в настоящее время используют плоские файлы для хранения данных или обмена информацией. Например, файл паролей имеет именно такую структуру.

Исходя из таблицы 3.1, плоский файл содержит некоторое число *атрибутов*, которые вместе составляют более сложный информационный элемент, получивший название *записи*. Атрибуты являются столбцами в таблице, а записи строками. Таким образом, плоский файл предназначен для сбора связанные записи вместе. Однако бывают случаи, когда плоский файл плохо подходит для поддержания работы приложения, и тогда вводят дополнительные возможности.

Предположим, что в приложение для конвертации валют нужно дополнительно ввести страны, государственный язык и численность

населения. В плоском файле одна строка соответствует одной записи, и каждый ее атрибут всегда находится на одном и том же месте (например, код валюты всегда является вторым атрибутом). Это дает возможность просматривать данные по столбцам. То есть добавление государственного языка можно понимать как добавление еще одного столбца во все имеющиеся строки.

Но при введении этого столбца возникает затруднение, поскольку в некоторых странах несколько официальных государственных языков (например, в Бельгии французский и фламандский языки). Такую проблему принято называть *повторяющиеся группы*. В целом можно сказать, что ситуация выглядит так: некоторый атрибут (язык) может повторяться в записи несколько раз, поэтому будет повторяться не только запись (строка), но и данные в ней.

Плоские файлы не справляются с такой задачей, поскольку нельзя определить, где заканчивается атрибут язык и начинается следующий атрибут. Задачу можно решить введением в файл некоторой структуры, но тогда он перестает быть плоским. Именно эта проблема стимулировала разработку более сложных СУБД. Причем нужен универсальный способ хранения и извлечения данных, а не частное решение конкретной задачи, придумываемое для обхода проблем, возникающих в системах обработки данных, то есть нужна БД.

По наиболее употребительному определению БД представляет собой большой набор данных, который организуется специальным образом для того, чтобы в дальнейшем обеспечить быстрый поиск и извлечение нужных данных.

Структурно СУБД это комплект библиотек, приложений и утилит, освобождающих разработчика от забот, касающихся хранения и управления данными. Кроме того СУБД предоставляет средства поиска и обновления записей. За несколько десятков лет для решения проблем хранения данных было создано множество СУБД.

3.1 Модели баз данных

При проектировании первых СУБД они разрабатывались так, чтобы тем или иным способом решить проблему повторяющихся групп. Эти методы разработки потребовали введения нового понятия – *модели СУБД*.

Одним из главных факторов проектирования ранних СУБД была эффективность. Легко манипулировать записями БД, имеющими фиксированное число атрибутов (столбцов). Так можно избежать проблемы повторяющихся групп. В этом случае можно прочитать каждую запись БД в простую структуру. Однако реально такие ситуации встречаются редко, поэтому программистам приходится обрабатывать весьма часто неструктурированные данные.

Сетевая модель вводит в БД **указатели (pointer)** – записи, которые содержат ссылки на другие записи. Например, так можно хранить запись для каждого заказчика, который в течение некоторого времени сделал множество заказов. Запись заказчика содержит указатель на одну запись заказа, которая содержит данные по этому заказу, а также указатель на другую запись заказа. Если применить этот принцип для рассмотренного ранее примера приложения конвертора валют, то можно использовать структуру, представленную на рисунке 3.1.

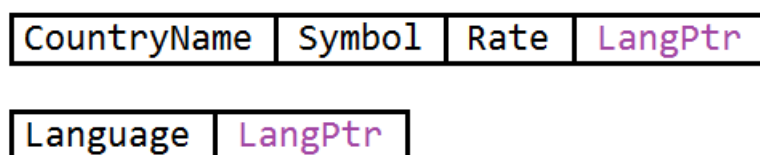


Рис. 3.1 Структура записей приложения конвертора валют

После загрузки данных получается связанный (отсюда и название модели – *сетевая*) список для языков, как показано на рисунке 3.2.

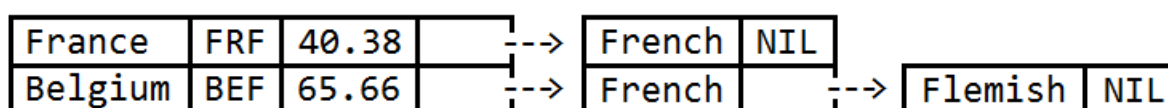


Рис. 3.2 Связанный список для государственных языков

Два разных типа записей, которые представлены на рисунке 3.2, будут храниться отдельно друг от друга, каждый тип записи – в своей собственной таблице.

Более целесообразно было бы, если бы названия языков не повторялись в БД. Можно ввести третью таблицу, содержащую языки и идентификатор (часто это просто целое число), который будет использоваться для ссылки на запись таблицы языков из записей другого типа. Такой идентификатор принято называть *ключом*.

В сетевой модели БД есть несколько преимуществ. Если нужно найти записи одного типа, относящиеся к определенной записи другого типа (например, государственные языки страны), то это можно сделать быстро, следуя по указателям, начиная с указанной записи.

Однако есть и недостатки. Если нужен перечень стран, в которых говорят по-французски, то придется пройти по ссылкам всех записей стран, и для больших БД такая операция займет много времени. Это можно исправить, создав другие связанные списки указателей специально для языков, но такое решение сложно и не универсально, так как нужно заранее решить, как будут организованы ссылки.

Кроме того, писать приложение для сетевой модели БД весьма сложно, так как ответственность за создание и поддержание указателей по мере обновления и удаления записей лежит на приложении.

Иерархическая модель появилась в конце 60-х годов прошлого столетия. В этой модели проблема повторяющихся групп решается за счет представления одних записей через множество других.

Сущность этой модели отражает спецификация материалов, применяемая для описания сложного продукта. Например, автомобиль состоит из шасси, кузова, двигателя и колес. Каждый из этих компонентов состоит из других деталей, которые опять-таки состоят из других более мелких деталей. В итоге мы доходим до простейших элементов автомобиля, которыми комплектуются его составляющие.

Иерархическая модель БД применяется до сих пор. СУБД на ее основе способна оптимизировать хранение данных, описывающих сложные составные объекты. Например, она позволяет легко определить, в каком автомобиле используется какая-то конкретная деталь.

Реляционная модель появилась в начале 70-х годов прошлого столетия. В этой модели вводится понятие *отношений* (**relation**) и правила использования таблиц для представления фактов (хранимых данных), устанавливающих отношения с реальными объектами.

В этот период времени стало очевидно, что эффективность хранения данных менее важна, чем целостность данных. В связи с этим реляционная модель придает гораздо большее значение целостности данных, чем любая другая ранее применявшаяся модель.

Реляционную СУБД определяет набор правил. Запись таблицы носит название «кортеж», который представляет собой упорядоченную группу атрибутов, принадлежащих определенному типу.

Кортежи строятся по одному шаблону, то есть у них одинаковое число компонентов одинаковых типов. Пример набора кортежей:

{"France", "FRF", 40.38}

{"Belgium", "BEF", 65.66}

Каждый из этих кортежей состоит из трех атрибутов: названия страны (строковый тип), валюты (строковый тип) и валютного курса (тип с плавающей точкой). В реляционной БД добавляемые записи (кортежи) должны следовать этой структуре шаблона, поэтому записи, которые представлены ниже, не могут быть добавлены:

{"Germany", "DEM"}

– недостаточно атрибутов

{"Switzerland", "CHF", "French", "German", "Italian", "Romans"}

– слишком много атрибутов

{13.68, "ITL", "Italy"}

– неправильный тип атрибутов (неправильный порядок)

Кроме этого, кортеж не может повторяться ни в одной таблице. Такая мера может выглядеть жесткой, поскольку может показаться, например, что для системы заказы-клиенты один клиент не сможет заказать какой-то продукт дважды. На практике этот запрет легко обойти, введя дополнительный атрибут.

Каждый атрибут записи должен быть *атомарным*, то есть быть простой информацией, а не записью или списком. Выше было показано, что типы соответствующих атрибутов записи должны совпадать. Практически это означает, что они должны быть либо строками, либо целыми числами, либо числами с плавающей точкой, или же принадлежать какому-то другому типу, поддерживаемому СУБД.

Атрибут, по которому различают записи, во всем остальном идентичные, называется *ключом*. Иногда в качестве ключа может использоваться комбинация из нескольких атрибутов.

Чтобы выделить одну запись таблицы от всех остальных записей (то есть сделать запись уникальной) используют атрибут, который называется *первичным ключом*. В реляционной БД каждое отношение (таблица) должно иметь первичный ключ.

Последнее правило, которое определяет структуру реляционной БД, представляет собой *ссылочную целостность*. Такое требование объясняется тем, что в любой момент времени все записи БД должны иметь смысл. Разработчик приложения, взаимодействующего с БД, должен быть крайне внимателен. Он обязан убедиться, что его код не нарушает целостности базы. Например, в системе заказы-клиенты при удалении клиента из таблицы CUSTOMER необходимо удалить и все его заказы из таблицы ORDERS. Иначе останутся записи о заказах, которым не сопоставлен ни один из клиентов.

Краткий итог — реляционная модель построена на математических понятиях множеств и отношений, поэтому при создании приложений следует придерживаться определенных правил.

3.2 Языки запросов

Главное достоинство реляционных СУБД заключается в том, что они дают возможность пользователю задавать вопросы о хранимых данных на специальном языке запросов. Ранние БД проектировались так, чтобы отвечать на определенные типы вопросов по содержащейся в них информации. Реляционные БД более гибки, так как отвечают на вопросы, которые еще не были известны при создании базы.

В реляционной модели отношения определяют множества, а множества обрабатываются математически. Например, в запросах применяется раздел теоретической логики – исчисление предикатов. Такой подход обеспечивает высокую производительность поиска и выборки данных.

Исторически первым был реализован язык запросов QUEL, который использовался в БД Ingres. Далее был реализован язык запросов QBE (Query By Example – запрос по примеру). Чуть позднее в исследовательском центре IBM был разработан язык SQL (Structured Query Language – язык структурированных запросов).

SQL предназначен для описания, изменения и извлечения данных, хранимых в реляционных БД. Базовым стандартом считается SQL-92, однако его модификации делаются постоянно. Например, в наиболее поздний стандарт SQL:2023 добавлен тип данных JSON и добавлена часть 16 «Свойства запросов по графам».

Существуют три уровня соответствия стандартов SQL: Entry SQL, Intermediate SQL и Full SQL. Наиболее распространен уровень «Entry», и PostgreSQL очень к нему близок. Разработчики PostgreSQL с каждой новой версией делают его все ближе к стандарту.

В языке SQL существуют команды трех типов:

- **Data Manipulation Language (DML)** – язык манипуляции данными. Эта часть SQL используется в 90% случаев и состоит из команд добавления, удаления, обновления и выборки данных из БД.

- **Data Definition Language (DDL)** – язык определения данных. Этот тип команд используют для создания таблиц и управления другими аспектами БД, структурированными на более высоком уровне.
- **Data Control Language (DCL)** – язык управления данными. Команды этого типа позволяют контролировать права доступа к данным. Такие команды обычно применяет администратор БД, в функции которого входит контроль над правами доступа.

Язык SQL практически признан стандартом запросов и почти каждая СУБД поддерживает SQL. Это способствует унификации, так как приложение, написанное с применением SQL, можно перенести с малыми затратами и использовать в другой БД.

Однако в условиях рынка производители БД создают отличающиеся друг от друга продукты. Так появилось несколько диалектов SQL, поскольку в стандарте языка не определены команды для многих задач администрирования БД. Поэтому есть различия между диалектами SQL, принятыми в Oracle, SQL Server и PostgreSQL.

Рассмотрим несколько примеров, которые иллюстрируют работу с языком SQL. В первом примере создается новая таблица для предлагаемых на продажу товаров, которые затем войдут в заказ:

```
CREATE TABLE item
(
    item_id          serial,
    description      char(64)          not null,
    cost_price       numeric(7,2),
    sell_price       numeric(7,2)
);
```

Здесь определено, что таблице нужен идентификатор, действующий как первичный ключ, который должен автоматически генерироваться СУБД. Идентификатор имеет тип **serial**, означающий, что при добавлении нового элемента **item** в последовательности будет создан

новый уникальный **item_id**. Описание (**description**) – это текстовый атрибут, состоящий из 64 символов. Себестоимость (**cost_price**) и цена продажи (**sell_price**) определяются как семизначные числа с плавающей точкой, с двумя знаками после запятой.

Во втором примере показано, как можно использовать язык SQL для заполнения только что созданной таблицы:

```
INSERT INTO item(description, cost_price, sell_price)
values('Fan Small', 9.23, 15.75);
INSERT INTO item(description, cost_price, sell_price)
values('Fan Large', 13.36, 19.95);
INSERT INTO item(description, cost_price, sell_price)
values('Toothbrush', 0.75, 1.45);
```

Основой языка SQL является оператор **SELECT**. Он применяется для создания результирующих множеств – групп записей (или атрибутов), соответствующих некоторому критерию, который может быть достаточно сложными. Результирующие множества могут использоваться в качестве целевых объектов для изменений с помощью оператора **UPDATE** или удалений оператором **DELETE**.

Примеры использования оператора **SELECT**:

```
SELECT * FROM customer, orderinfo
WHERE orderinfo.customer_id = customer.customer_id GROUP BY customer_id
SELECT customer.title, customer.fname, customer.lname,
COUNT(orderinfo.orderinfo_id) AS "Number of orders" FROM customer, orderinfo
WHERE customer.customer_id = orderinfo.customer_id
GROUP BY customer.title, customer.fname, customer.lname
```

Эти операторы **SELECT** перечисляют все заказы клиентов в указанном порядке и подсчитывают количество заказов, сделанных каждым клиентом персонально.

СУБД PostgreSQL предоставляет пользователю несколько способов доступа к данным, а именно:

- использовать консоль для выполнения операторов SQL;
- непосредственно встроить SQL в приложение;
- использовать вызовы функций API для подготовки и выполнения операторов SQL, просмотра результирующих множеств и обновления данных из различных языков программирования;
- прибегнуть к опосредованному доступу к данным базы PostgreSQL с применением драйвера ODBC, JDBC или библиотеки DBI.

3.3 СУБД

Весь арсенал СУБД (программы, утилиты и библиотеки) нужен для построения БД и их использования. В обязанности СУБД входит:

- **Создание БД.** Некоторые системы создают одну или несколько БД внутри одного файла, другие задействуют несколько файлов ОС или реализуют низкоуровневый доступ к диску. Пользователи не должны заботиться о структуре таких файлов, поскольку весь необходимый доступ обеспечивает СУБД.
- **Организация выполнения запросов и обновлений.** Обеспечивает возможность запросов, удовлетворяющий некоторому критерию, например, выбор всех заказов, сделанных клиентом, но еще не доставленных. До широкого распространения SQL способы выражения таких запросов менялись от системы к системе.
- **Многозадачность.** Если с БД работают несколько приложений или к ней одновременно осуществляют доступ несколько пользователей, то СУБД должна гарантировать, что обработка запроса каждого пользователя не влияет на работу остальных. Если какой-то пользователь записывает данные тогда, когда другим тоже нужно прочитать (или записать) данные, то эти другие ожидают. Одновременно может происходить несколько считываний данных.

- **Ведение журнала.** Все изменений данных за некоторый период времени отображаются в журнале, который используется для отслеживания ошибок, а также для восстановления данных в случае сбоя системы. Производится резервное копирование данных и ведется журнал транзакций. Резервную копию можно будет использовать для восстановления БД в случае повреждения диска.
- **Гарантия безопасности БД.** Только зарегистрированные пользователи могут манипулировать данными и структурой БД. Для каждой БД определяется иерархия пользователей. Во главе этой структуры стоит *суперпользователь*, который может изменять все, далее идут пользователи, которые могут добавлять и удалять данные, а внизу находятся те, кто имеет право только на чтение.
- **Сохранение ссылочной целостности.** СУБД имеет свойства, способствующие поддержанию ссылочной целостности (корректности данных). Если запрос или обновление нарушает правила реляционной модели, то СУБД выдает сообщение об ошибке.

Большинство современных СУБД используют реляционную модель и язык запросов SQL. Их рейтинг представлен на рисунке 3.3. Яндекс.Облако не поддерживает СУБД Oracle и Microsoft SQL Server.

Rank			DBMS
Aug 2023	Jul 2023	Aug 2022	
1.	1.	1.	Oracle +
2.	2.	2.	MySQL +
3.	3.	3.	Microsoft SQL Server +
4.	4.	4.	PostgreSQL +
5.	5.	5.	MongoDB +
6.	6.	6.	Redis +

Рис. 3.3 Рейтинг СУБД за 2022-2023 годы по данным сайта **db-engines.com**

В нашей стране все большую популярность приобретает СУБД PostgreSQL, которая предоставляет много различных возможностей. Она достаточно надежна, имеет хорошую производительность и ра-

ботает практически на всех платформах. Помимо этого PostgreSQL свободно распространяется и имеет открытый исходный код.

PostgreSQL обладает практически всеми возможностями, которые есть в других СУБД, а также имеет некоторые свои дополнительные возможности. Перечень функциональных возможностей PostgreSQL:

- Транзакции
- Вложенные запросы
- Представления
- Ссылочная целостность – внешние ключи
- Сложные блокировки
- Типы, определяемые пользователем
- Наследственность
- Правила
- Проверка совместимости версий

Одним из важных преимуществ PostgreSQL является ее структурное построение или архитектура. Как и другие СУБД PostgreSQL может применяться в среде клиент-сервер. Это дает массу преимуществ и пользователям, и разработчикам.

Основу PostgreSQL составляет серверный процесс БД, который выполняется на одном сервере. Доступ из приложений к данным осуществляется через этот процесс. Программы-клиенты не могут получить доступ к данным самостоятельно, даже если они работают на компьютере, выполняющем серверный процесс.

Такое разделение клиентов и сервера позволяет организовать распределенную систему. Можно отделить клиентов от сервера посредством сети и разрабатывать клиентские приложения в среде, удобной для пользователя. Например, можно реализовать БД под Linux и создать клиентские приложения, которые будут работать в операционной системе Windows. На рисунке 3.4 приведена схема типичной модели распределенного приложения PostgreSQL.

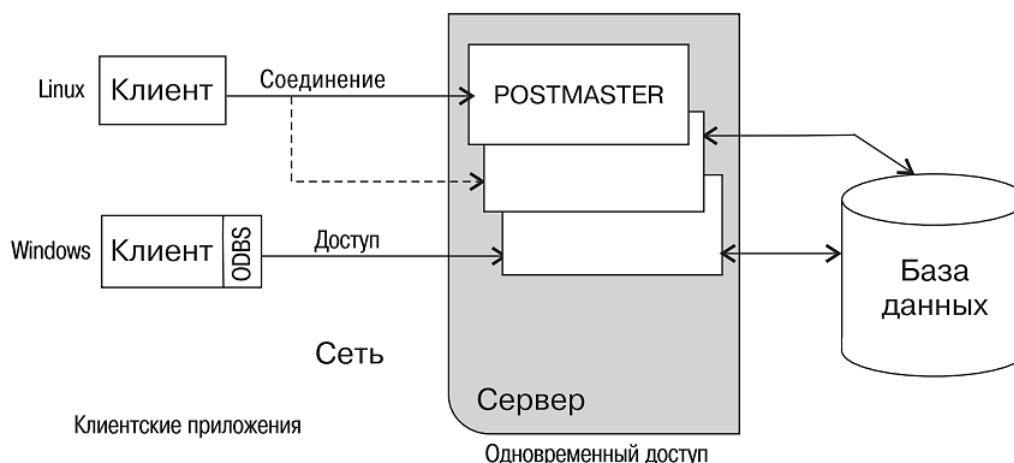


Рис. 3.4 Схема типичной модели приложения PostgreSQL

По данной схеме несколько клиентов соединяются с сервером по сети. PostgreSQL ориентирован на протокол TCP/IP, и каждый клиент соединяется с основным серверным процессом БД (**POSTMASTER**), который создает новый серверный процесс специально для обслуживания запросов на доступ к данным конкретного клиента.

Поскольку манипулирование данными сосредоточено на сервере, СУБД не приходится контролировать многочисленных клиентов, получающих доступ в совместно используемый каталог сервера. Это означает, что PostgreSQL поддерживает целостность данных даже при одновременном доступе большого количества пользователей.

Приложения-клиенты соединяются с БД по специальному протоколу PostgreSQL. Однако можно установить на стороне клиента ПО, предоставляющее стандартный интерфейс для работы с нужным приложением (например, по стандарту ODBC или JDBC). Доступность ODBC-драйвера позволяет применять PostgreSQL в качестве БД для многих приложений, включая Excel и Access.

Архитектура клиент-сервер делает возможным разделение труда. Машина-сервер хорошо подходит для хранения и управления доступом к большим объемам данных, она может использоваться как надежный репозиторий. Для клиентов могут быть разработаны сложные графические приложения.

3.4 Создание БД в Яндекс.Облаке

ЛР3. Цель: Создать на базе «Яндекс.Облако» БД под управлением PostgreSQL и рассмотреть основные операции по работе с ней.

Пример выполнения лабораторной работы №3 также описан с помощью пронумерованных шагов. При этом требования к каждому шагу аналогичны требованиям лабораторной работы №1 и №2.

Задание. Выполните самостоятельно все шаги аналогично приведенному примеру. В качестве варианта задания на рисунках должна присутствовать *персональная информация* о студенте, которая в данном примере выделена красной рамкой.

Шаг 1. Откройте консоль управления «Яндекс.Облака», выберите облако и справа вверху нажмите кнопку **Создать каталог** (см. рис. 3.5).

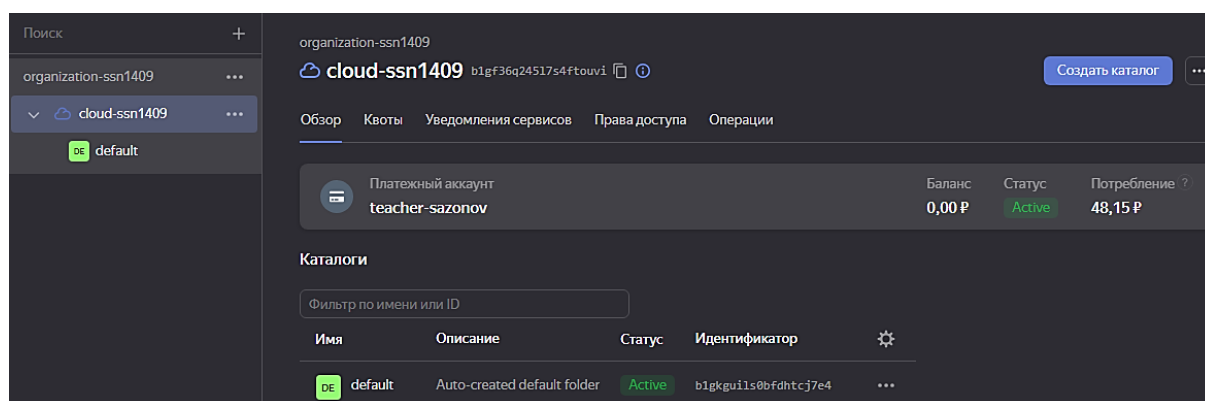


Рис. 3.5 Окно выбранного облака с его параметрами

Шаг 2. Введите имя (последние символы – ваши инициалы), описание и нажмите на кнопку **Создать**, как показано на рисунке 3.6.

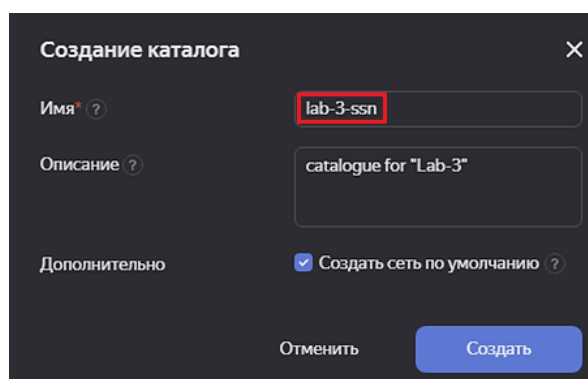


Рис. 3.6 Создание каталога для лабораторной работы №3

Шаг 3. На вкладке «Дашборд каталога» в списке сервисов выберите **Managed Service for PostgreSQL**. После этого откроется стартовое окно создания кластера PostgreSQL, как на рисунке 3.7.



Рис. 3.7 Стартовое окно создания кластера PostgreSQL

Шаг 4. Нажмите кнопку **Создать кластер** и введите первую часть параметров кластера аналогично показанным на рисунке 3.8.

Рис. 3.8 Ввод параметров кластера PostgreSQL (часть 1)

Шаг 5. Укажите атрибуты БД (имя БД, имя пользователя и пароль), создайте группы безопасности, поставьте галочки для параметров по рисунку 3.9 и для завершения нажмите **Создать кластер**.

Доступ из DataLens ? ☒

Доступ из консоли управления ? ☒

Доступ из Data Transfer в режиме Serverless ? ☐

Доступ из Serverless ? ☐

Сбор статистики ? ☒

Интервал сбора сессий ? 1 86400

Интервал сбора запросов ? 60 86400

Автоматическое переключение мастера ☒

Режим работы менеджера подключений

Защита от удаления ? ☒

Настройки СУБД

⚠ Расширенные настройки СУБД могут значительно повлиять на производительность кластера. Меняйте их только если вы точно знаете, что хотите сделать.

Рис. 3.9 Ввод параметров кластера PostgreSQL (часть 2)

Процесс создания кластера занимает некоторое время, поэтому необходимо подождать до тех пор, пока **Доступность** примет значение **Alive**, чтобы появилось окно, показанное на рисунке 3.10.

Имя	Идентификатор	Описание	Доступность	Дата создания	Версия	Окружение	
postgresql-ssn	c9qk2gnqocsm7n1nlhnm	Lab-3	Alive	16.08.2023, в 07:30	15	PRODUCTION	...

Рис. 3.10 Окно созданного и действующего кластера PostgreSQL

Шаг 6. Откройте в созданном кластере вкладку «Обзор» с его параметрами. Они должны быть аналогичны рисункам 3.11 и 3.12.

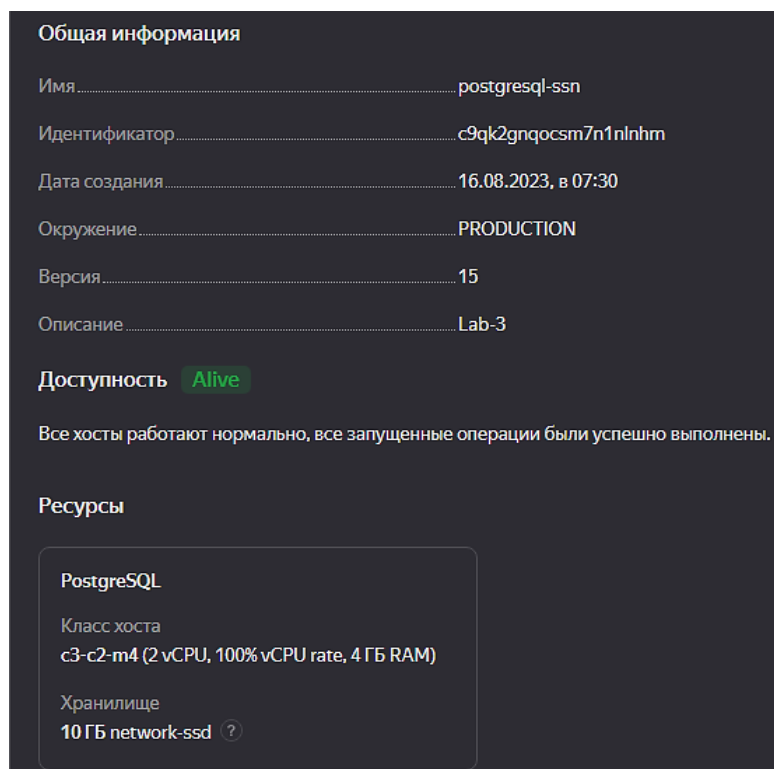


Рис. 3.11 Окно «Обзор» действующего кластера (часть 1)

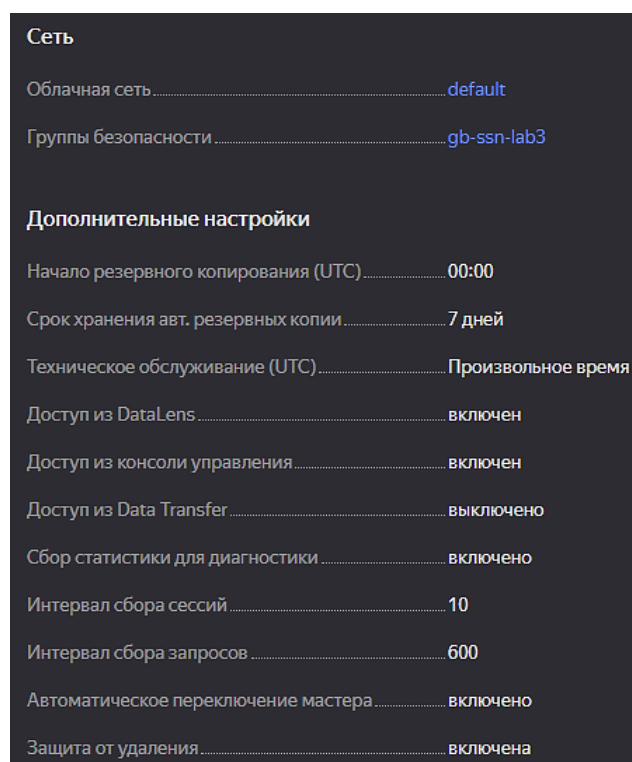


Рис. 3.12 Окно «Обзор» действующего кластера (часть 2)

Шаг 7. Откройте вкладку «Базы данных» созданного кластера и убедитесь, что в этом кластере имеется созданная БД (на рисунке 3.13 созданная БД имеет название **db1**).

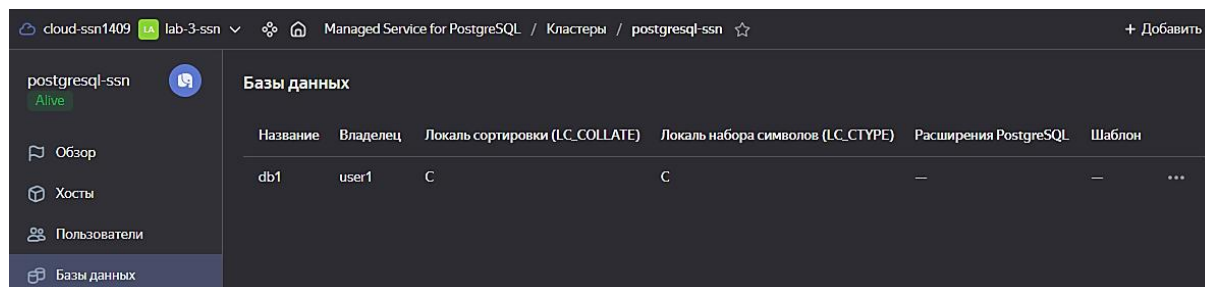


Рис. 3.13 Окно вкладки «Базы данных» действующего кластера

К созданным хостам кластера Managed Service for PostgreSQL можно подключиться двумя способами:

- Через интернет, если был настроен публичный доступ для требуемого хоста. К таким хостам можно подключиться только с использованием SSL-соединения.
- С помощью VM Yandex Cloud, которые расположены в той же облачной сети. Если к нужному хосту нет публичного доступа, то в этом случае для подключения с помощью VM обязательно использовать SSL-соединение.

В данной лабораторной работе рассматривается подключение к хостам кластера БД Managed Service for PostgreSQL и выполнение простых базовых операций с помощью предварительно установленной VM, на которой установлена ОС Ubuntu версии 22.04.

Для подключения к кластеру БД необходимо, чтобы группы безопасности содержали правила, с помощью которых разрешается трафик данных с определенных портов, IP-адресов или из других групп безопасности. Создание и редактирование правил для подключения к кластеру БД, которое можно осуществить с помощью специ-

ального сервиса Virtual Private Cloud, будет рассмотрено далее на шаге 11.

Шаг 8. В каталоге, который был создан на первом шаге, создайте ВМ с Ubuntu 22.04, параметры которой аналогичны рисунку 3.14.

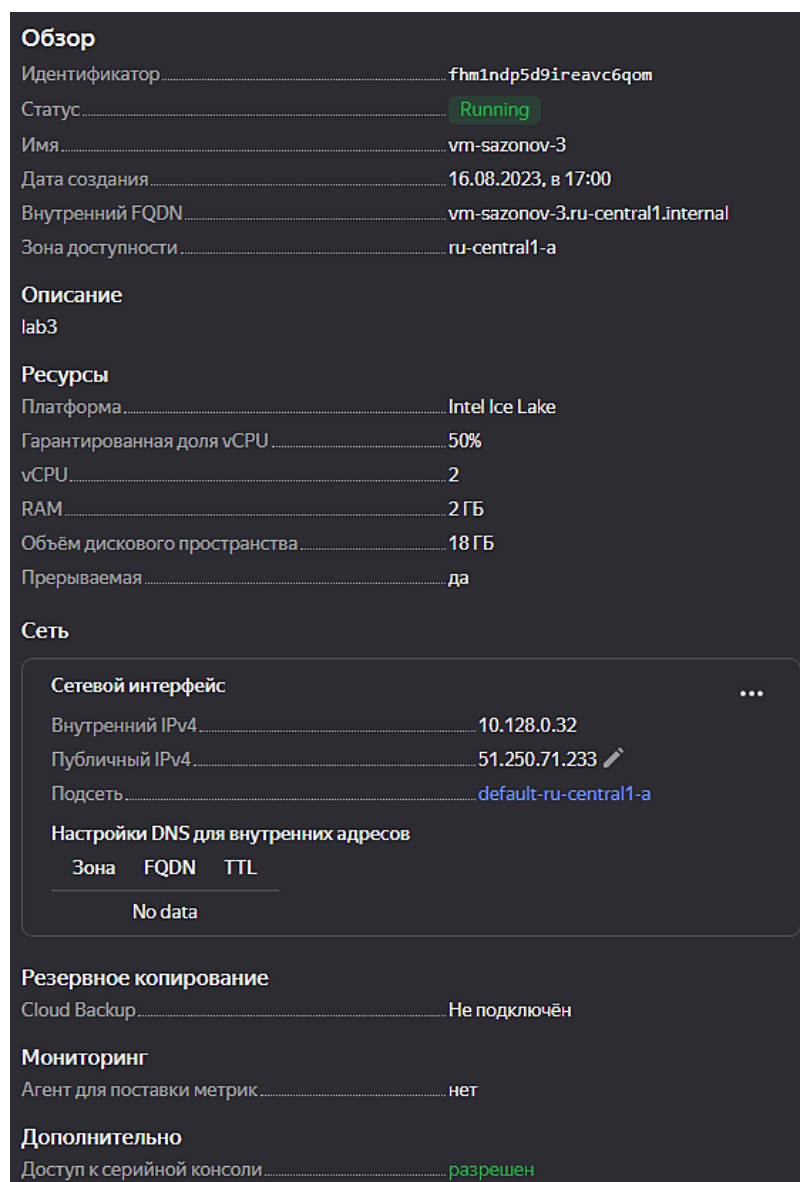


Рис. 3.14 Окно параметров созданной ВМ

Шаг 9. Выполните подключение к ВМ по протоколу **SSH**, используя программу PuTTY, как это делалось в лабораторной работе №2. Для подключения необходимо указать публичный IP-адрес ВМ. Этот адрес изменяется при остановке ВМ и повторном запуске.

Шаг 10. После настройки VM и установки соединения с сервером выполните обновление Ubuntu 22.04, а также инсталляцию клиента БД PostgreSQL с помощью следующих команд:

```
sudo apt update
```

```
sudo apt upgrade
```

```
sudo apt install postgresql-client
```

Шаг 11. Для подключения и работы с БД выполните настройку правил групп безопасности кластера. С этой целью в созданном сервисе Virtual Private Cloud нужно открыть Группы безопасности и выбрать созданную группу, как показано на рисунке 3.15.

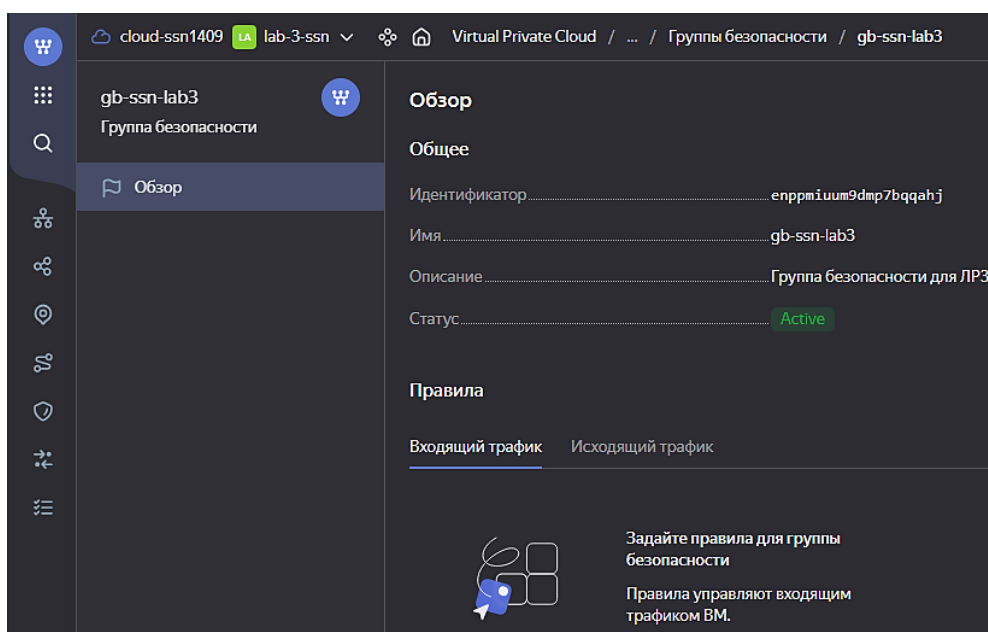


Рис. 3.15 Окно обзора выбранной группы безопасности

Шаг 12. Нажмите на кнопку «Редактирование», а затем «Добавить правило». Первое правило должно соответствовать рис. 3.16.

Добавление правила для входящего трафика [X]

Описание: lab3

Диапазон портов: 6432 [Выбрать весь диапазон]

Протокол: Любой [v]

Источник: CIDR | Группа безопасности | Проверки состояния балансировщика

Группа безопасности: Текущая

[Отменить] [Сохранить]

Рис. 3.16 Окно первого правила для всех групп безопасности

Второе правило предназначено для входящего трафика и его параметры должны соответствовать рисунку 3.17.

Добавление правила для входящего трафика [X]

Описание: lab3

Диапазон портов: 22 [Выбрать весь диапазон]

Протокол: TCP [v]

Источник: CIDR | Группа безопасности | Проверки состояния балансировщика

CIDR блоки: 0.0.0.0 / 0 [v]
[Добавить CIDR]

[Отменить] [Сохранить]

Рис. 3.17 Окно правил для входящего трафика

Третье правило предназначено для исходящего трафика и его параметры должны соответствовать рисунку 3.18.

Добавление правила для исходящего трафика

Описание: lab3

Диапазон портов: 0-65535 Выбрать весь диапазон

Протокол: Любой

Назначение: CIDR | Группа безопасности | Проверки состояния балансировщика

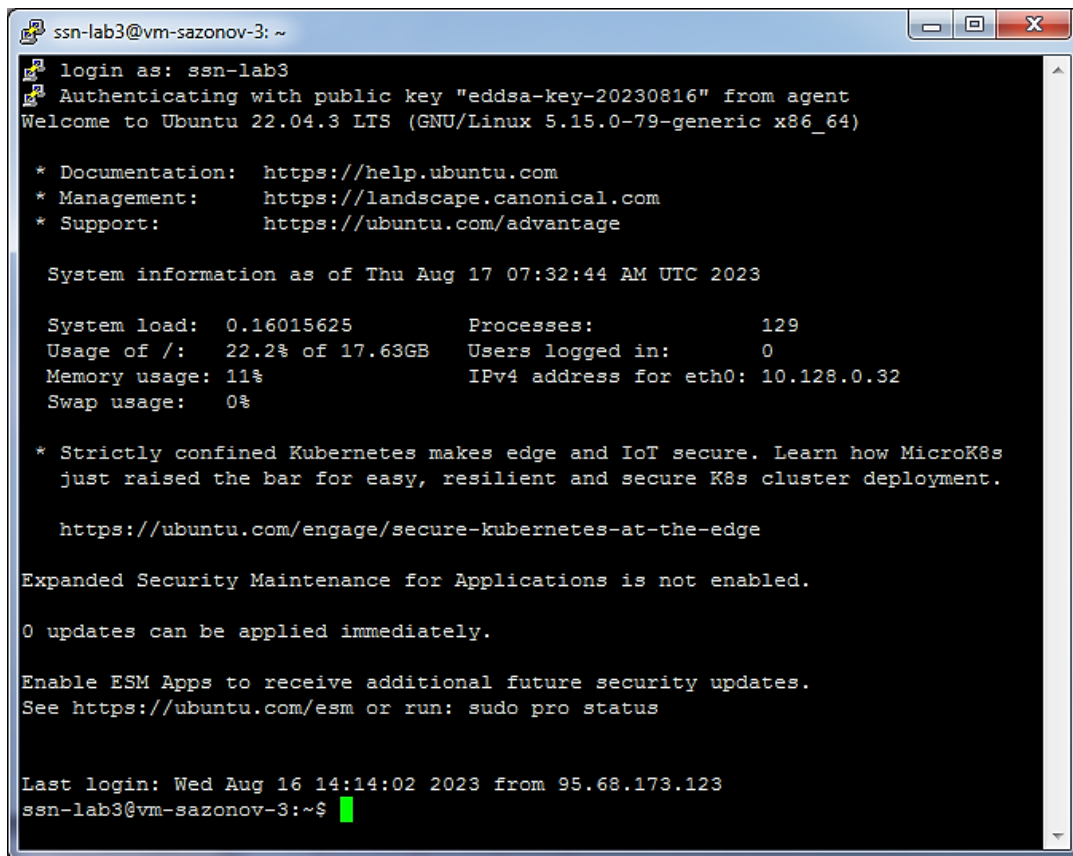
CIDR блоки: 0.0.0.0 / 0 Добавить CIDR

Отменить Сохранить

Рис. 3.18 Окно правил для исходящего трафика

Кнопки «Сохранить» необходимо нажимать во всех трех представленных окнах: для всех групп безопасности, входящего трафика, исходящего трафика, а также общего окна правил, которое не представлено на рисунках.

Шаг 13. Запустите приложение PuTTY через агент, введите публичный IP-адрес ВМ, порт 22, соединение SSH, а также путь к файлу приватного ключа (в пункте Connection → SSH → Auth → Credentials).

A screenshot of a PuTTY terminal window. The title bar shows 'ssn-lab3@vm-sazonov-3: ~'. The terminal output includes a login message for 'ssn-lab3', authentication details, and system information for Ubuntu 22.04.3 LTS. It lists documentation, management, and support links. System info includes load, processes, memory usage, and IP address. It also mentions Kubernetes security and ESM status. The prompt is 'ssn-lab3@vm-sazonov-3:~\$' with a green cursor.

```
ssn-lab3@vm-sazonov-3: ~
login as: ssn-lab3
Authenticating with public key "eddsa-key-20230816" from agent
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-79-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage

System information as of Thu Aug 17 07:32:44 AM UTC 2023

System load:  0.16015625      Processes:            129
Usage of /:   22.2% of 17.63GB Users logged in:       0
Memory usage: 11%           IPv4 address for eth0: 10.128.0.32
Swap usage:   0%

* Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
  just raised the bar for easy, resilient and secure K8s cluster deployment.

https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Wed Aug 16 14:14:02 2023 from 95.68.173.123
ssn-lab3@vm-sazonov-3:~$
```

Рис. 3.18 Окно запущенного приложения PuTTY

Шаг 14. Выполните еще раз команды Шага 10 и проверьте корректность установок правил для всех групп безопасности, которые были выполнены на Шаге 12. Пример реализации подключения для кластера с идентификатором c9qash3nb1v9ulc8j9nm к хосту-мастеру:

```
psql "host=c-c9qash3nb1v9ulc8j9nm.rw.mdb.yandexcloud.net \
port=6432 \
sslmode=verify-full \
dbname=<имя БД> \
user=<имя пользователя> \
target_session_attrs=read-write"
```

Шаг 14. Откройте вкладку SQL в каталоге созданного кластера БД PostgreSQL, введите имя пользователя БД, пароль, имя БД и нажмите кнопку «Подключиться», как показано на рисунке 3.19.

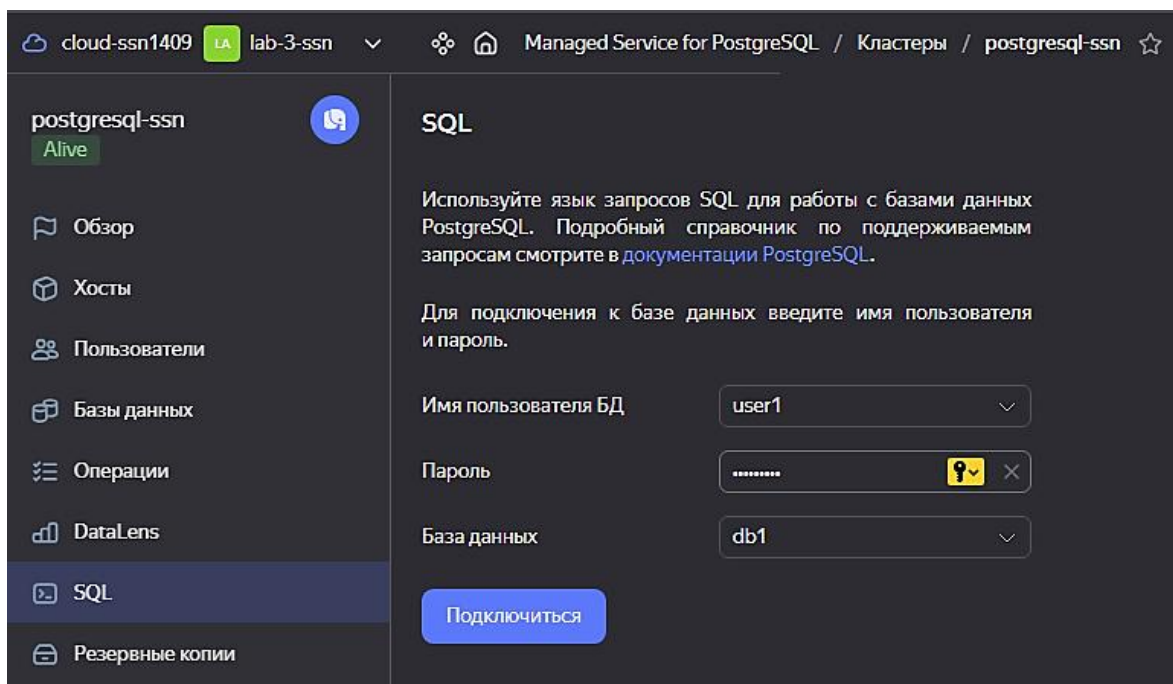


Рис. 3.19 Подключение к БД с помощью SQL

Шаг 15. Создайте в БД первую таблицу `students` с пятью колонками, как показано на рис. 3.20, и нажмите кнопку «Выполнить».

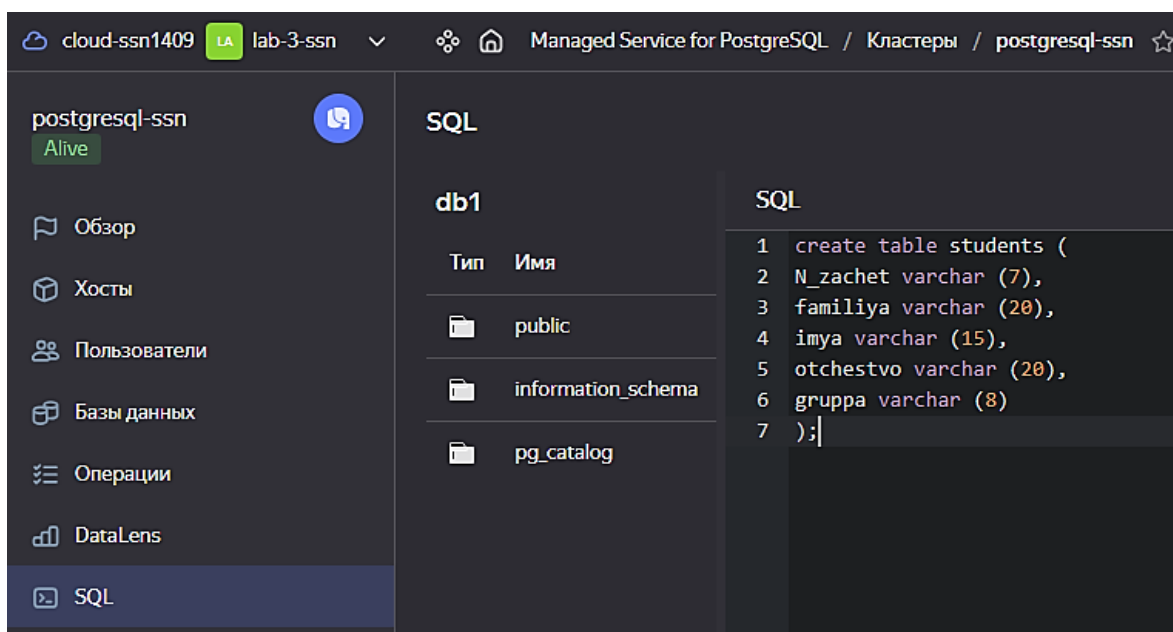


Рис. 3.20 Пример создание первой таблицы БД

Шаг 16. Аналогичным образом создайте в БД вторую таблицу `осенкі` также с пятью колонками. На рисунке 3.21 показан результат создания второй таблицы после нажатия кнопки «Выполнить».

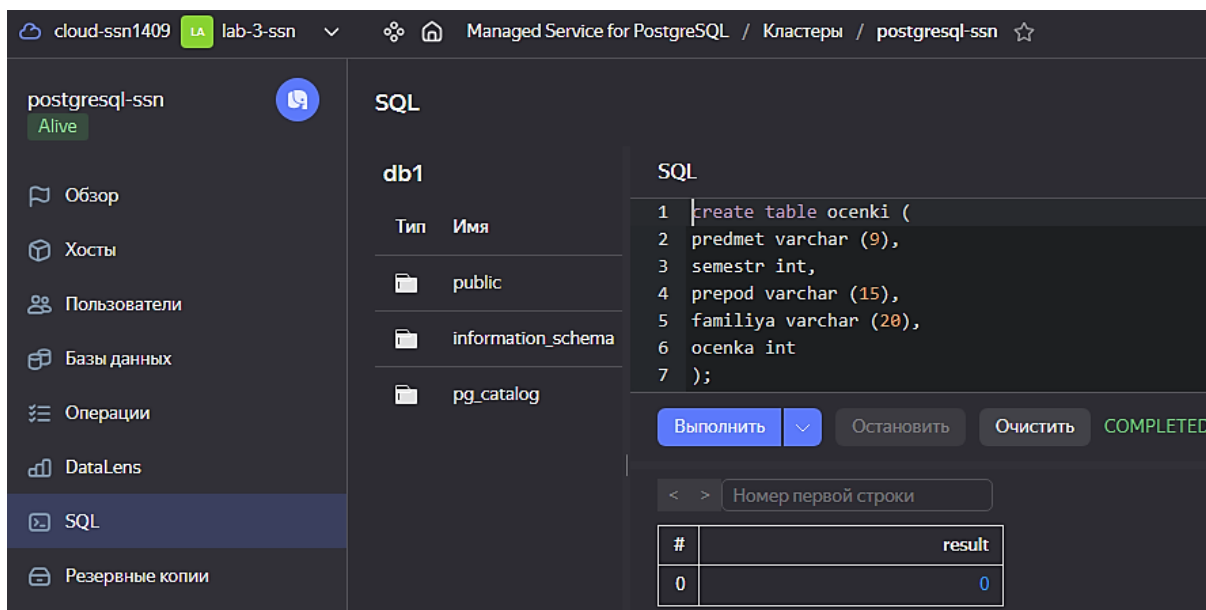


Рис. 3.21 Результат создания второй таблицы БД

Шаг 17. После создания двух таблиц БД необходимо наполнить их содержимым. Эта операция осуществляется с помощью команды insert into. На рисунке 3.22 показан пример наполнения данными первой таблицы students. В качестве варианта задания в этой таблице выполните замену данных студента «Ivanov» своими данными.

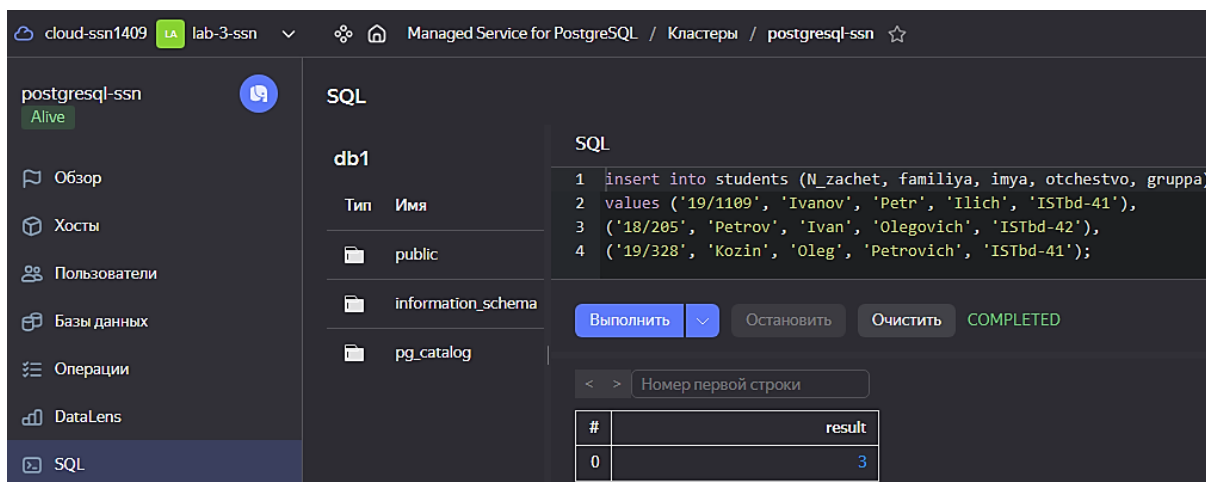


Рис. 3.22 Результат наполнения данными первой таблицы БД

Шаг 18. Аналогичным образом осуществляется наполнение второй таблицы ocenki. На рисунке 3.23 показан пример наполнения дан-

ными второй таблицы. Здесь также в качестве варианта задания нужно выполнить замену данных студента «Ivanov» своими данными.

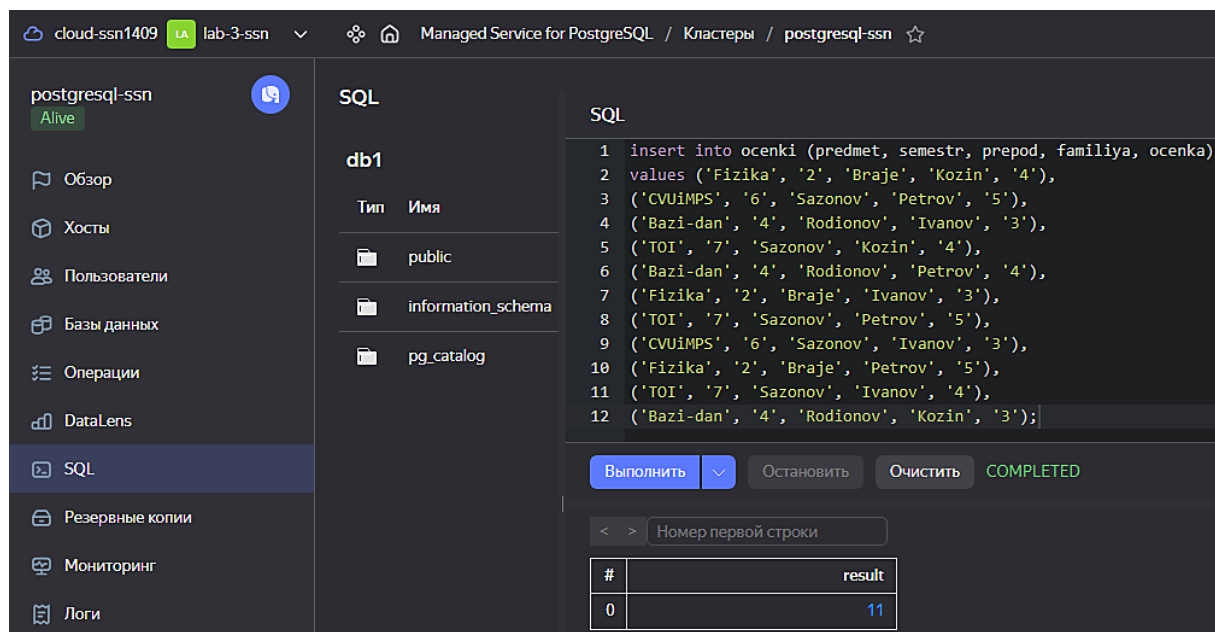


Рис. 3.23 Результат наполнения данными второй таблицы БД

Шаг 19. На последнем шаге этой работы выполните запрос своих оценок (вместо «Ivanov»). Пример запроса показан на рисунке 3.24.

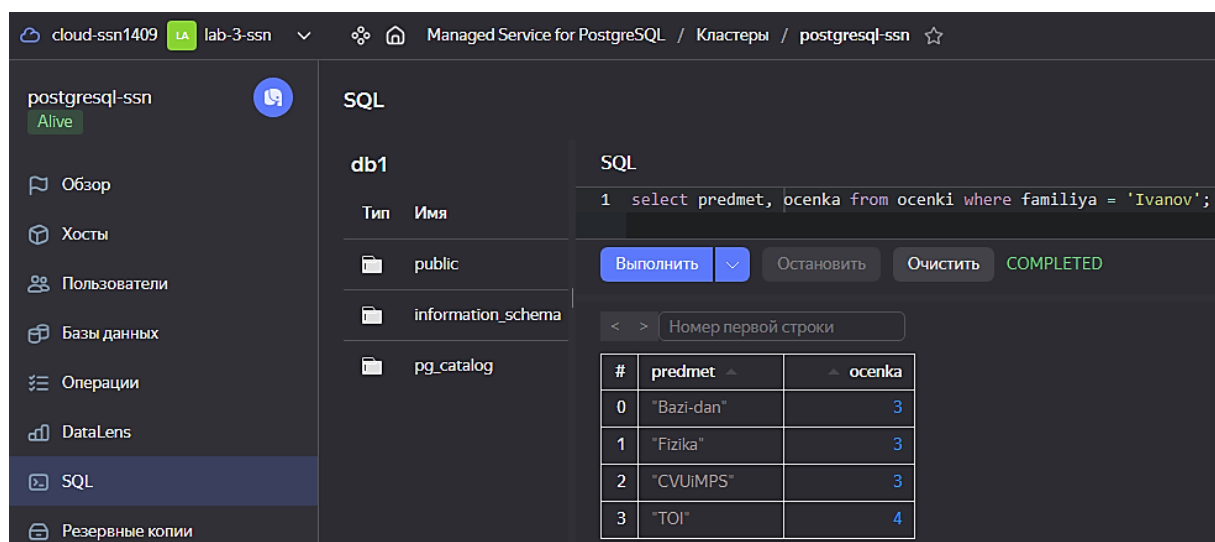


Рис. 3.24 Пример запроса оценок студента «Ivanov»

Отчет должен содержать пункты аналогичные другим работам.

Раздел 4. Создание сайтов

—