МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ВОЛГОГРАДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

На правах рукописи

П.Д. Кравченя

Облачные вычислительные системы

Учебное пособие

Волгоград
2021

УДК 519.816:004.89:004.94

Кравченя П.Д.
Облачные вычислительные системы : учеб. пособие / П.Д. Кравченя; ВолгГТУ. – Волгоград, 2021. – 33 с.

В учебном пособии рассмотрены вопросы, связанные с принципами и технологиями создания веб-сервисов для поддержки моделей машинного обучения, а также их развертывание с использованием систем контейнеризации.

Учебное пособие предназначено для магистров, обучающихся по программам магистратуры по профилю «искусственный интеллект» по направлениям 09.04.01 «Информатика и вычислительная техника», 09.04.03 «Прикладная информатика», 09.04.02 «Информационные системы и технологии». Учебное пособие выполнено в рамках реализации гранта на разработку программ бакалавриата и программ магистратуры по профилю «Искусственный интеллект», а также на повышение квалификации педагогических работников образовательных организаций высшего образования в сфере искусственного интеллекта (конкурс 2021-ИИ-01 от 10.06.2021).

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Облачные вычислительные системы реализуют технологии распределённой обработки данных, в которой компьютерные ресурсы и мощности предоставляются пользователю как веб-сервис. В современных системах для разворачивания и управления запущенными сервисами широко используются технологии контейнеризации. В данном курсе рассматриваются вопросы проектирования и реализации веб-сервисов, инструменты для воспроизводимости экспериментов и управления жизненным циклом моделей анализа данных, представленных в виде сервисов, а также способы их размещения на вычислительных узлах с использованием современных систем для запуска и управления контейнерами.

1. МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ К ПРАКТИЧЕСКИМ ЗАНЯТИЯМ

1.1. Практика № 1. Основы протокола HTTP.

1.1.1. Цель практической работы

Цель практической работы № 1 состоит в том, чтобы на практических примерах дать студентам общее представление об устройстве и принципах работы протокола HTTP, положенного в основу функционирования веб-сервисов. Рассматриваются основные методы протокола HTTP, содержание и значение основных заголовков запросов и ответов, правила построения запросов и ответов.

1.1.2. Список задач к практическому занятию

1. Разберите представленный HTTP-контент. Какая операция выполняется с его помощью?

```
   POST /cgi-bin/qtest HTTP/1.1
   Host: aram
   Content-Type: multipart/form-data;
boundary=2a8ae6ad-f4ad-4d9a-a92c-6d217011fe0f
   Content-Length: 514
   --2a8ae6ad-f4ad-4d9a-a92c-6d217011fe0f
   Content-Disposition: form-data;
name="datafile1"; filename="r.gif"
   Content-Type: image/gif
   GIF87a.............,...........D..;
   --2a8ae6ad-f4ad-4d9a-a92c-6d217011fe0f
   Content-Disposition: form-data;
name="datafile2"; filename="g.gif"
   Content-Type: image/gif
```

```
  GIF87a.....,.....D..;
  --2a8ae6ad-f4ad-4d9a-a92c-6d217011fe0f
  Content-Disposition: form-data;
name="datafile3"; filename="b.gif"
  Content-Type: image/gif
  GIF87a.....,.....D..;
--2a8ae6ad-f4ad-4d9a-a92c-6d217011fe0f--
```

2. В ответ на запрос пользователя веб-сервер вернул ответ, представленный ниже. Составьте пример запроса, ответ на который мог выглядеть таким образом. Попросите сервер передать, по возможности, сжатую версию ресурса.

```
  HTTP/1.1 200 OK
  Cache-Control: private
  Content-Type: text/html
  Content-Encoding: gzip
  Vary: Accept-Encoding
  Server: Microsoft-IIS/7.5
  Date: Sun, 04 Aug 2013 13:33:59 GMT
  Content-Length: 8434
```

3. Пользователь Петя хочет отправить веб-серверу изображение foto.jpg для его размещения по адресу www.trustserver.org/users/fotos/17/myfoto.jpg. Помогите Пете составить соответствующий HTTP-запрос, принимая во внимание следующие требования.

а) Для передачи информации используется протокол HTTP версии 1.1.

б) Размер изображения составляет 2,5 килобайта.

в) Серверу нужно сообщить используемое клиентское ПО: браузер Mozilla Firefox 12.

г) После обработки запроса и выдачи ответа сервер, по возможности, не должен закрывать соединение, ожидая на нем последующего запроса.

4. Пользователь запросил у веб-сервера некоторый документ, который на сервере представлен экземплярами на различных языках. Запрос пользователя представлен ниже. На каком (-их) языке(-ах) пользователь предпочитает получить документ? Поясните ответ.

```
  GET /docs/presentations.html HTTP/1.1
  Host: example.com
  Date: Wed, 29 Jul 2020 15:01:17 GMT
  Accept-Language:  fr-CH,  fr;q=0.9,  en;q=0.8,
de;q=0.7, *;q=0.5
```

5. Иногда при получении ответов на предыдущий запрос возникает ситуация, когда клиент отображает версию страницы не того языка, который он указал в предпочтениях, хотя этот язык поддерживается сервером. Почему такое может случаться? Сформируйте пример ответа на запрос предыдущего задания, указав в нем заголовок, который позволит исправить ситуацию.

1.2. Практика № 2. Организация кэширования в протоколе HTTP

1.2.1. Цель практической работы

Цель практической работы № 2 состоит в том, чтобы рассмотреть на практике цель и необходимость кэширования, понять принципы управления кэшированием HTTP-содержимого с помощью заголовков протокола.

1.2.2. Список задач к практическому занятию

1. На некотором веб-сервере расположены ресурсы, запросы HEAD к которым привели к получению ответов следующего содержания:

```
HTTP/1.1 200 OK
Date: Fri, 25 Nov 2016 06:24:22 GMT
Content-Type: application/json
Content-Length: 1467
Last-Modified: Wed, 23 Nov 2016 01:00:00 GMT
ETag: 6hf83h3b96t
```

```
HTTP/1.1 200 OK
Date: Fri, 25 Nov 2016 06:24:24 GMT
Content-Type: image/jpeg
Content-Length: 3853
ETag: 0054gs66s
Cache-Control: max-age=3600,must-revalidate
```

Затем в течение двух часов от пользователей поступали запросы на получение ресурсов. Определите, в каких случаях промежуточные кэширующие сервера отправляли валидационные запросы на веб-сервер, а также их примерное содержание.

2. В некоторой сети функционирует HTTP-сервер, доступ к которому осуществляется через кэширующий веб-прокси. С кэширующим прокси-сервером взаимодействует пользователь user1, выполняя запросы и получая ответы, заголовки которых представлены ниже.

```
GET /info/image.png HTTP/1.1
HTTP/1.1 200 OK
Date: Wed, 11 Feb 2009 08:05:16 GMT
Cache-Control: max-age=120, must-revalidate
```

```
GET /info/script.js HTTP/1.1
```

```
HTTP/1.1 200 OK
Date: Wed, 11 Feb 2009 08:06:14 GMT
Cache-Control: max-age=60, no-cache, must-revalidate
```

```
GET /info/data.json HTTP/1.1
HTTP/1.1 200 OK
Date: Wed, 11 Feb 2009 08:06:20 GMT
Cache-Control: max-age=120, private
```

В некоторый момент времени с кэширующим сервером соединяется другой пользователь user2, и оба пользователя обращаются к прокси с запросами, представленными ниже. В 08 часов 06 минут ресурс /info/image.png изменяется на веб-сервере.

```
GET /info/script.js HTTP/1.1
Date: Wed, 11 Feb 2009 08:06:51 GMT
From: user2@df.com
```

```
GET /info/image.png HTTP/1.1
Date: Wed, 11 Feb 2009 08:07:02 GMT
From: user2@df.com
```

```
GET /info/data.json HTTP/1.1
Date: Wed, 11 Feb 2009 08:07:58 GMT
From: user1@df.com
```

```
GET /info/script.js HTTP/1.1
Date: Wed, 11 Feb 2009 08:08:02 GMT
From: user1@df.com
```

```
GET /info/data.json HTTP/1.1
Date: Wed, 11 Feb 2009 08:08:10 GMT
From: user2@df.com
```

а) Определите последовательность и типы запросов от кэширующего прокси к веб-серверу в различные моменты времени. Какие из них завершатся повторной передачей ресурса веб-сервером?

б) В какой момент времени пользователи получат обновленный ресурс /info/image.png?

в) Как изменится ответ на предыдущий вопрос, если в момент времени 08 часов 06 минут на веб-сервере обновится ресурс /info/script.js?

3. Пользователь получает от веб-сервера три файла: /index.html, /logo.png и /script.js. Известно, что:

а) изображение меняется крайне редко, только в исключительных случаях;

б) исходный код страницы может изменяться часто, до нескольких раз в день;

в) исходный код скрипта изменяется периодически раз в два месяца;

г) устаревшая версия скрипта приведет к неработоспособности веб-приложения у пользователя.

Сформируйте примеры ответов веб-сервера на запросы пользователя на получение этих ресурсов, установив в них заголовки кэширования, позволяющие оптимизировать использование кэша в соответствии с требованиями к ресурсам. Поясните решение.

1.3. Практика № 3. Проектирование RESTful-сервисов

1.3.1. Цель практической работы

Цель практической работы № 3 состоит в том, чтобы на практических примерах дать студентам общее представление о принципах работы веб-сервисов, правилах проектирования RESTful-сервисов и технологиях, используемых при их построении.

1.3.2. Пример задачи к практическому занятию

Спроектируйте RESTful-сервис для организации школьного электронного журнала. Журнал должен отражать успеваемость учеников в школе и посещение ими занятий. Пользователями сервиса могут быть учителя и ученики, которые обладают своим набором прав в приложении. В процессе решения задачи:

а) определите CRUD-операции для каждой категории пользователей;

б) определите требуемые URI и постройте их иерархию в соответствии с REST;

в) приведите примеры HTTP-запросов и ответов сервиса, удовлетворяющих требованиям архитектуры REST.

1.4. Практика № 4. Проектирование микросервисной архитектуры

1.4.1. Цель практической работы

Цель практической работы № 4 состоит в том, чтобы на практических примерах дать студентам общее представление о микросервисной архитектуре приложений, определить ее преимущества и недостатки, понять разницу между монолитным приложением и микросервисами, научиться проектировать приложения как коллекцию взаимодействующих друг с другом микросервисов.

1.4.2. Пример задачи к практическому занятию

Спроектируйте микросервисную архитектуру системы, реализующую магазин электронных книг. Данная система должна позволять зарегистрированному пользователю получить список книг для покупки, оплатить книгу, скачать оплаченную книгу на компьютер пользователя. Система должна поддерживать повторное скачивание купленных книг. В процессе решения задачи:

а) выделите микросервисы, определите их функциональность и интерфейсы;

б) определите способ взаимодействия между микросервисами;

в) определите способ авторизации клиентов магазина.

1.5. Практика № 5. Сборка образов Docker

1.5.1. Цель практической работы

Цель практической работы № 5 состоит в том, чтобы на практических примерах дать студентам общее представление о способах изоляции приложений в пределах одного вычислительного узла, изучить преимущества контейнеризации, научиться писать содержимое Dockerfile для сборки контейнера Docker.

1.5.2. Пример задачи к практическому занятию

Составьте Dockerfile для формирования образа с TCP-сервером. Исходный код сервера представлен в файле server.cpp, который необходимо скомпилировать в процессе создания образа. Примите во внимание, что:

а) образ нужно составить на базе чистой системы Centos 7;

б) установка компилятора C++ производится командой: yum install gcc-c++;

в) настройки сервера хранятся в директории options, находящейся в папке с исполняемым файлом;

г) запуск сервера производится командой: server.out –d;

д) после запуска сервер ожидает соединения на порту 12553;

е) для работоспособности сервера переменная окружения TCPSERVER_MODE должна быть установлена в значение DAEMON.

1.6. Практика № 6. Запуск контейнеризованных приложений

1.6.1. Цель практической работы

Цель практической работы № 6 состоит в том, чтобы на практических примерах дать студентам общее представление о способах исполнения контейнеров Docker, изучить назначение и различие образов и контейнеров Docker, научиться собирать и запускать контейнеры Docker на вычислительной машине под управлением Linux.

1.6.2. Пример задач к практическому занятию

1. Запустите Docker-контейнер на базе образа, собранного в предыдущем задании. Обеспечьте доступность сервера на порту 8080 хоста. Также, обеспечьте сохранность настроек сервера после его перезапуска.

2. Вышеописанный сервер может обеспечивать и UDP-соединение. Для этого он должен быть запущен командой: server.out -d -m udp –data=..., где data — ключ, указывающий расположение пользовательских файлов с данными для UDP-соединения. Составьте Dockerfile для формирования образа UDP-сервера, используя в качестве базового ранее созданный образ. Приведите команду запуска контейнера на базе данного образа.

1.7. Практика № 7. Работа с системой DVC

1.7.1. Цель практической работы

Цель практической работы № 7 состоит в том, чтобы на практических примерах дать студентам дать общее представление об воспроизводимости

вычислительных экспериментов, назначении и принципе работы инструмента DVC.

1.7.2. Описание практической работы

На практическом занятии рассматривается применение инструмента DVC для обеспечения воспроизводимости экспериментов анализа данных конкретного проекта анализа данных. Изучаются возможности DVC, его связь с Git, их применение для обеспечения контроля версий кода, данных и моделей. Рассматриваются построение конвейеров обработки данных и их практическое применение.

1.8. Практика № 8. Работа с системой MLFlow

1.8.1. Цель практической работы

Цель практической работы № 8 состоит в том, чтобы на практических примерах дать студентам дать общее представление о системах управления жизненным циклом машинного обучения на примере MLFlow.

1.8.2. Описание практической работы

На практическом занятии рассматривается применение инструмента MLFlow для обеспечения сбора метрик и гиперпараметров в процессе обучения конкретной модели анализа данных, упаковки обученной модели в воспроизводимый формат для дальнейшего повторного использования и ее размещения в реестре моделей.

2. МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ

2.1 Лабораторная работа № 1. Создание RESTful-сервисов.

2.1.1 Цели и задачи

Целью работы является ознакомление с общими принципами построения и реализации RESTful веб-сервисов.

Задачи работы:

1. Изучить принцип работы протокола HTTP.
2. Познакомиться с механизмами взаимодействия клиента и веб-сервиса.
3. Получить навыки разработки RESTful-сервисов.

2.1.2 Постановка задачи

Реализовать многопользовательский RESTful веб-сервис, предоставляющий клиентам функциональность в соответствии с индивидуальным заданием. Индивидуальные задания определяются номером студента в списке группы в соответствии с приведенными в таблице 1 темами (либо с другими возможными темами на усмотрение преподавателя). Реализовать клиентское приложение для работы с сервисом. Клиент и сервер должны удовлетворять следующим требованиям:

1. Сервис должен быть спроектирован и реализован с использованием архитектуры REST.
2. Сервис должен реализовывать обработку CRUD-запросов над спроектированными сущностями.
3. Сетевое взаимодействие между клиентом и сервисом должно осуществляться посредством HTTP протокола. На усмотрение студента допускается использование HTTPS-протокола.

4. Языки программирования и фреймворки для реализации клиента и веб-сервиса выбираются студентом самостоятельно.

5. Сервис должен сохранять работоспособность при одновременной работе с ним нескольких пользователей. Каждый клиент должен быть обслужен независимо от остальных.

6. Сервис должен предоставлять возможность работы двум категориям пользователей (обычные пользователи и пользователи с повышенными правами).

Табл. 1. Темы индивидуальных заданий

Номер варианта	Функциональность веб-сервиса
1	Работа с электронным каталогом библиотеки
2	Работа с интернет-магазином
3	Работа с сервисной службой интернет-провайдера
4	Работа с электронной системой заказа блюд в ресторане
5	Работа с системой заказа такси
6	Работа с системой заказа билетов в театр
7	Работа с системой электронного расписания ВУЗа
8	Работа с системой записи пациентов к врачу
9	Работа со школьным электронным журналом
10	Работа с брокером фондовой биржи
11	Работа с системой почтовой доставки товаров
12	Работа с системой бронирования номера в гостинице
13	Подбор изображений заданной тематики на основе нейросетей
14	Выполнение классификации текстов по темам на базе искусственного интеллекта
15	Кластеризация изображений по заданным пользователем параметрам

2.1.3 Требования и состав отчёта

1. Отчёт должен быть выполнен на листах размера А4.

2. Отчёт должен начинаться с титульного листа с названием вуза и факультета, номером и названием лабораторной работы, вариантом, ФИО студента, № группы, ФИО преподавателя, городом и годом.

3. В отчёте нужно кратко описать задание, показать основные этапы решения задачи, сформулировать выводы.

4. Отчёт предоставить в бумажном или электронном виде (записать на флэш-накопитель и продублировать на электронную почту).

2.1.4 Вопросы к отчету лабораторной работы

1. Протокол HTTP. Передача данных посредством протокола. Заголовки и тела запросов и ответов. Примеры.

2. Методы протокола HTTP. Безопасные, идемпотентные и кэшируемые методы. Взаимодействие между клиентом и сервером по протоколу HTTP. Коды ответов сервера.

3. Управление кэшированием в протоколе HTTP. Инструкция Cache-Control.

4. Понятие веб-сервисов. Стандарты, используемые при построении веб-сервисов. Компоненты веб-сервисов.

5. Понятие архитектуры REST, ее преимущества и недостатки. RESTful-сервисы. Принципы проектирования RESTful-сервисов.

6. Микросервисная архитектура: понятие, преимущества и недостатки. Примеры.

2.2 Лабораторная работа № 2. Изучение технологий контейнеризации на базе Docker.

2.2.1 Цели и задачи

Целями работы являются ознакомление с общими принципами контейнеризации приложений и реализация запуска веб-сервисов в контейнерах Docker.

Задачи работы:

1. Изучить принцип работы технологии контейнеризации.
2. Познакомиться с архитектурой и принципами работы системы Docker.
3. Изучить инструменты Docker для работы с образами и контейнерами.
4. Получить навыки контейнеризации пользовательских приложений.

2.2.2 Постановка задачи

1. Выполните соединение с кластером и откройте удаленный рабочий стол X2Go на узле **`node39`** (см. инструкцию в Приложении).
2. Создайте рабочую директорию, названную с использованием фамилии пользователя, и скопируйте в нее файды Dockerfile, hello.conf и index.html, полученные от преподавателя (см. инструкцию в Приложении). Войдите в созданную директорию.

```
$ mkdir ivanov-directory
$ cd ivanov-directory
```

3. Запустите контейнер с Nginx с помощью команды:

```
$ docker run --name ivanov-nginx -d -p XXXX:80
nginxdemos/hello
```

где `ivanov` – имя пользователя кластера, `XXXX` – свободный порт на узле кластера. Подождите, пока контейнер запустится.

4. Проверьте статус контейнера, вызвав команду:

```
$ docker ps
```

5. Проверьте работоспособность запущенного контейнера, перейдя в браузере по адресу:

http://node39.cluster:XXXX

Сделайте снимок экрана и поместите его в протокол.

6. Остановите контейнер с помощью команды:

```
$ docker stop ivanov-nginx
```

7. Удалите контейнер с помощью команды:

```
$ docker rm ivanov-nginx
```

8. Внесите изменения в файл стартовой страницы `index.html`.

9. В директории с файлами `Dockerfile`, `hello.conf` и измененным файлом `index.html` выполните следующую команду:

```
$ docker build -t ivanov-new-nginx .
```

10. После окончания сборки образа запустите контейнер на его основе:

```
$ docker run --name ivanov-nginx -d -p YYYY:80
ivanov-new-nginx
```

11. Проверьте работоспособность запущенного контейнера, перейдя в браузере по адресу:

http://node39.cluster:YYYY

Сделайте снимок экрана и поместите его в протокол. Сравните его со снимком экрана, сделанного ранее. Сделайте выводы.

12. Остановите контейнер с помощью команды:

```
$ docker stop ivanov-nginx
```

13. Удалите контейнер с помощью команды:

```
$ docker rm ivanov-nginx
```

14. Поэкспериментируйте с содержимым `Dockerfile` и соберите несколько образов с различными данными. Запустите контейнеры на основе данных образов. Проанализируйте результаты.

15. Индивидуальное задание – выполните контейнеризацию сервиса, разработанного в рамках лабораторной работы № 1. Для этого подготовьте Dockerfile и необходимые файлы для сборки, соберите образ и запустите контейнер на его основе. Продемонстрируйте его работоспособность. Сделайте выводы.

2.2.3 Требования и состав отчёта

1. Отчёт должен быть выполнен на листах размера А4.

2. Отчёт должен начинаться с титульного листа с названием вуза и факультета, номером и названием лабораторной работы, вариантом, ФИО студента, № группы, ФИО преподавателя, городом и годом.

3. В отчёте нужно кратко описать задание, показать основные этапы решения задачи, сформулировать выводы.

4. Отчёт предоставить в бумажном или электронном виде (записать на флэш-накопитель и продублировать на электронную почту).

2.2.4 Вопросы к отчету лабораторной работы

1. Понятие виртуализации и контейнеризации, их возможности. Преимущества и недостатки каждой из технологий.

2. Принцип работы технологии контейнеризации. Пространство пользователя и пространство ядра. Kernel Namespaces, CGroups, UnionFS.

3. Система Docker. Архитектура системы. Функции каждого компонента архитектуры и взаимодействие между ними.

4. Процесс создания образов Docker. Dockerfile и контекст создания образа. Уровни образа. UnionFS. Влияние кэширования на процесс создания образов.

5. Основные инструкции Dockerfile, их функции и правила использования. Способы записи инструкций.

6. Обеспечение коммуникации между контейнерами: проброс портов и подключение контейнеров.

7. Сохранение данных контейнера, понятие томов (volumes). Управление томами Docker.

8. Основные команды Docker для работы с образами и контейнерами.

2.3 Лабораторная работа № 3. Управление образами и контейнерами Docker с помощью инструмента Docker-compose

2.3.1 Цели и задачи

Целью работы является ознакомление с инструментом управления образами и контейнерами Docker-compose на практических примерах.

Задачи работы:

1. Изучить способы и инструменты оркестрации контейнеров Docker.

2. Изучить принцип работы и основные команды Docker-compose.

3. Получить навыки создания, настойки и администрирования образов и контейнеров с помощью Docker-compose.

2.3.2 Постановка задачи

1. Выполните соединение с кластером и откройте удаленный рабочий стол X2Go на узле **`node39`** (см. инструкцию в Приложении).

2. Создайте рабочую директорию, названную с использованием фамилии пользователя, и скопируйте в нее содержимое архива, полученного от преподавателя (см. инструкцию в Приложении). Войдите в созданную директорию.

```
$ mkdir ivanov-directory
$ cd ivanov-directory
```

3. Архив содержит два проекта, реализованных в виде Docker-образов, и конфигурационный файл `Docker-compose`. Первый проект

«nginx» содержит образ веб-сервера и идентичен рассматриваемому во второй лабораторной работе. Второй проект «squid» представляет собой прокси-сервер для данных, отправляемых пользователю.

4. Откройте в текстовом редакторе файл docker-compose.yml и ознакомьтесь с его содержимым. Определите, что означает каждая инструкция в нем.

5. Выполните построение образов обоих проектов, выполнив в директории с конфигурационным файлом команду:

```
$ docker-compose build
```

Проследите процесс сборки образов.

6. Запустите контейнер с Nginx с помощью команды:

```
$ docker-compose up nginx
```

Проследите процесс запуска контейнера. Если система сообщает об ошибке из-за того, что требуемый для запуска сервера порт занят, измените порт хостовой системы для сервиса «nginx» в конфигурационном файле docker-compose.yml на любой из свободных и повторите попытку.

7. Проверьте работоспособность сервера, перейдя в браузере по адресу:

http://192.168.1.39:XXXX

где XXXX – номер порта хостовой системы, связанного с контейнером «nginx» в соответствии с содержимым конфигурационного файла. Если сервис неработоспособен, выясните причину ошибки и устраните ее.

8. Остановите работу контейнера нажатием комбинации клавиш «Ctrl-C» в терминале с работающим экземпляром docker-compose.

9. Запустите оба контейнера с помощью команды:

```
$ docker-compose up
```

Проследите процесс запуска контейнеров. Если система сообщает об ошибке из-за того, что требуемые для запуска порты заняты, измените

порты хостовой системы для нужных сервисов в конфигурационном файле `docker-compose.yml` на любые из свободных и повторите попытку.

10. Укажите браузеру Firefox соединяться с сайтами через прокси-сервер (Настройки Прокси-сервер Настроить...) с параметрами, соответствующими значениям полей конфигурационного файла `docker-compose`. Очистите поле «Не использовать прокси для...» (рисунок 1).

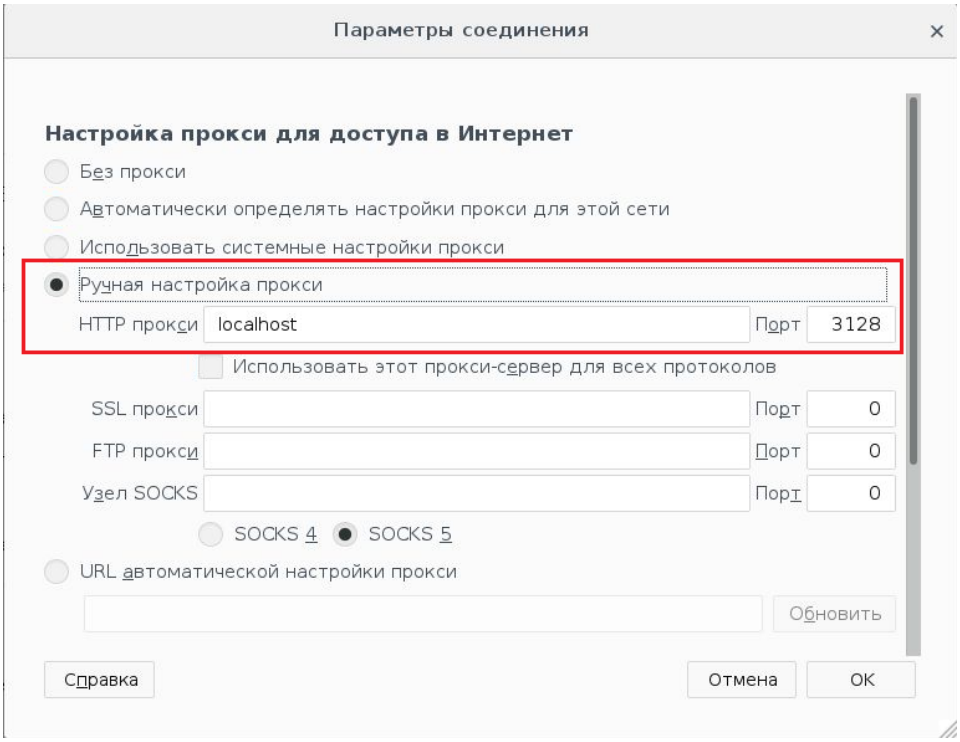


Рис. 1. Параметры соединения Mozilla Firefox с прокси-сервером

11. Проверьте работоспособность запущенных сервисов, перейдя в браузере по адресу:

`http:// 192.168.1.39:XXXX`

где XXXX – номер порта хостовой системы, связанного с контейнером «`nginx`» в соответствии с содержимым конфигурационного файла. Если сервисы неработоспособны, выясните причину ошибки и устраните ее.

12. Удалите созданные образы с помощью команды:

```
$ docker-compose rm
```

13. Измените содержимое файлов `Dockerfile` и `docker-compose.yml` таким образом, чтобы конфигурационный файл сервиса `squid` (`squid.conf`) можно было указывать при старте контейнера, а не при сборке образа (воспользуйтесь концепцией томов (`volumes`)). Опишите Ваше решение, приведите доказательства его работоспособности.

14. Поместите в протокол результаты, полученные при выполнении работы. Объясните их, сделайте выводы.

15. Индивидуальное задание – добавьте к проекту, разработанному Вами в рамках лабораторной работы № 2, еще один или несколько сервисов (прокси-сервер, сервер базы данных, сервер логирования и т.д.). Составьте соответствующие файлы `Dockerfile` и `docker-compose.yml`. Продемонстрируйте работоспособность проекта. Сделайте выводы.

2.3.3 Требования и состав отчёта

1. Отчёт должен быть выполнен на листах размера А4.

2. Отчёт должен начинаться с титульного листа с названием вуза и факультета, номером и названием лабораторной работы, вариантом, ФИО студента, № группы, ФИО преподавателя, городом и годом.

3. В отчёте нужно кратко описать задание, показать основные этапы решения задачи, сформулировать выводы.

4. Отчёт предоставить в бумажном или электронном виде (записать на флэш-накопитель и продублировать на электронную почту).

2.3.4 Вопросы к отчету лабораторной работы

1. Понятие оркестрации Docker-контейнеров. Инструменты оркестрации Docker.

2. Инструмент Docker-compose: назначение и принцип работы. Конфигурационный YAML-файл и его формат. Определение переменных окружающей среды (environment).

3. Определение и использование сервисов (services), сетей (networks) и томов (volumes) в docker-compose.yml. Типы драйверов для сетей.

4. Сборка Docker-образов проекта с помощью Docker-compose с использованием Dockerfile. Инструкция build в docker-compose.yml.

5. Использование существующих образов проекта в Docker-compose для запуска контейнера. Инструкция image в docker-compose.yml.

6. Политики перезапуска сервисов в Docker-compose. Инструкция restart в docker-compose.yml.

7. Определение зависимости между сервисами в Docker-compose. Инструкция depends_on в docker-compose.yml.

8. Основные команды Docker-compose. Построение и удаление образов с помощью Docker-compose. Запуск, просмотр статуса и удаление контейнеров с помощью Docker-compose.

3. МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ КУРСОВОЙ РАБОТЫ

3.1. Задание на курсовую работу и требования к ее выполнению

Курсовая работа выполняется индивидуально каждым студентом в рамках следующей общей темы:

«Создание сетевого клиент-серверного приложения для ... с применением технологий контейнеризации»

Точную формулировку темы студент формулирует самостоятельно в соответствии с индивидуальным вариантом к первой лабораторной работе по данной дисциплине. Работа должна представлять собой решения индивидуальных заданий трех лабораторных работ по дисциплине, оформленная в виде пояснительной записки к курсовой работе с добавлением теоретической главы.

Основные моменты, которые должны быть отражены в пояснительной записке к курсовой работе:

а) Теоретический материал, объемом не более трети работы, посвященный описанию и анализу информационных технологий, используемых при написании пояснительной записки.

б) Подробная постановка задачи (что изначально дано, и что требуется получить в процессе выполнения работы).

в) Выбор и описание технологии, выбранной для реализации веб-сервиса.

г) Подробное описание протокола взаимодействия клиента и сервиса.

д) Описание алгоритма работы сервера с использованием диаграмм прецедентов, последовательностей, классов, блок-схем алгоритмов, и т.д.

е) Описание процедуры создания Docker-образов сервера и вспомогательных сервисов (клиент контейнеризовать не нужно), содержимое файлов Dockerfile и docker-compose.yml.

ж) Описания процесса запуска контейнеров и приведения проекта в работоспособное состояние.

з) Тестирование работоспособности контейнеризованных сервисов и клиента.

и) Приведение исходных кодов программ клиента и веб-сервиса, самостоятельно разработанных студентом, в Приложении в курсовой работе.

Выполнение пунктов а) – е) является необходимым (обязательным) условием для защиты курсовой работы и достаточным условием для получения оценки «удовлетворительно».

3.2 Методические указания по выполнению курсовой работы

Работа выполняется параллельно и в контексте индивидуальных заданий к лабораторному практикуму по дисциплине. Оформляется в письменной форме в течение 10 недель с момента выдачи задания. Контрольный срок сдачи – последний месяц семестра.

Содержание пояснительной записки курсовой работы представляет собой авторский текст, поясняющий все пункты и этапы работы. Все основные рисунки, таблицы, схемы должны быть прокомментированы в тексте. Возможно использование ссылок на авторские тексты третьих лиц с обязательным указанием источников. Не допускается использование чужого авторского текста в оформлении работы без ссылок на первоисточник (плагиат). Работы, уличенные в плагиате, не допускаются до защиты.

Рекомендуемый объем пояснительной записки – 20-25 страниц.

Правила оформления курсовой работы:

- курсовая работа оформляется в редакторе MS Word / OpenOffice (*.doc, *.docx, *.odt);

- листы формата А4, ориентация книжная;
- поля: левое – 30 мм, правое – 10 мм, верхнее – 15 мм, нижнее – 20 мм. Абзацы в тексте начинают с отступом, равным 15 (или 12,5) мм.
- шрифт – Times New Roman;
- размер шрифта 14 pt;
- междустрочный интервал – 1,5;
- абзацный отступ – 1,25 см;
- нумерация страниц сквозная, номер на первой странице не ставится;
- в конце работы необходим список использованной литературы в соответствии с ГОСТ Р 7.0.5 – 2008.

3.3. Примерное содержание курсовой работы

В пояснительной записке должны быть представлены следующие разделы:

- содержание;
- введение;
- не менее двух (лучше – трех) разделов, описывающих работу;
- заключение;
- список использованных источников;
- возможно, одно или несколько приложений.

3.4 Примерные варианты заданий контрольной работы

1. Создание веб-сервиса для поддержки интеллектуального анализа логов вычислительного кластера с применением технологий контейнеризации.
2. Создание веб-сервиса для анализа данных фондового рынка с помощью машинного обучения.

3. Создание веб-сервиса для генерации изображений заданного стиля посредством нейронных сетей.

4. Создание веб-сервиса распознавания голосовых команд для управления роботом.

5. Создание веб-сервиса для подбора научных статей по определенной тематике с помощью искусственного интеллекта.

6. Создание веб-сервиса для идентификации объектов на изображениях методами искусственного интеллекта.

7. Создание веб-сервиса для генерации текстового описания изображений методами машинного обучения.

8. Создание веб-сервиса распознавания сорняков на фотографиях посредством нейронных сетей.

9. Создание веб-сервиса для определения количества людей на фотографии с помощью нейросетей.

10. Создание веб-сервиса для суммаризации текста с помощью современных порождающих моделей.

ЗАКЛЮЧЕНИЕ

В рамках курса на практических примерах и в лабораторном практикуме рассматриваются общие вопросы проектирования и создания RESTful веб-сервисов, их контейнеризация и запуск в изолированном окружении с требуемыми параметрами, инструменты для управления жизненным циклом проектов анализа данных и воспроизводимости моделей. Уделяется особое внимание развитию у студента понимания принципов работы используемых инструментов и формированию навыков разработки веб-сервисов и их контейнеризации с использованием Docker.

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА ПО КУРСУ

1. Маркелов, А. А. Введение в технологию контейнеров и Kubernetes / А. А. Маркелов. — Москва : ДМК Пресс, 2019. — 194 с. — ISBN 978-5-97060-775-6. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/131702 (дата обращения: 12.10.2021).

2. Дэвис, К. Шаблоны проектирования для облачной среды : руководство / К. Дэвис ; перевод с английского Д. А. Беликова. — Москва : ДМК Пресс, 2020. — 388 с. — ISBN 978-5-97060-807-4. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/140593 (дата обращения: 12.10.2021).

3. Кочер, П. С. Микросервисы и контейнеры Docker : руководство / П. С. Кочер ; перевод с английского А. Н. Киселева. — Москва : ДМК Пресс, 2019. — 240 с. — ISBN 978-5-97060-739-8. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/123710 (дата обращения: 12.10.2021).

4. Антонио, Д. Библиотека Keras – инструмент глубокого обучения. Реализация нейронных сетей с помощью библиотек Theano и TensorFlow / Д. Антонио, П. Суджит ; перевод с английского А. А. Слинкин. — Москва : ДМК Пресс, 2018. — 294 с. — ISBN 978-5-97060-573-8. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/111438 (дата обращения: 12.10.2021).

5. Тоуманнен, Б. Программирование GPU при помощи Python и CUDA : руководство / Б. Тоуманнен ; перевод с английского А. В. Борескова. — Москва : ДМК Пресс, 2020. — 252 с. — ISBN 978-5-97060-821-0. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/179469 (дата обращения: 12.10.2021).

6. Сейерс, Э. Х. Docker на практике / Э. Х. Сейерс, А. Милл ; перевод с английского Д. А. Беликов. — Москва : ДМК Пресс, 2020. — 516 с. —

ISBN 978-5-97060-772-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/131719 (дата обращения: 12.10.2021).

7. Моуэт, Э. Использование Docker / Э. Моуэт ; научный редактор А. А. Маркелов ; перевод с английского А. В. Снастина. — Москва : ДМК Пресс, 2017. — 354 с. — ISBN 978-5-97060-426-7. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/93576 (дата обращения: 12.10.2021).

8. Воронина, В. В. Разработка веб-сервисов для анализа слабоструктурированных информационных ресурсов : учебное пособие / В. В. Воронина. — Ульяновск : УлГТУ, 2016. — 165 с. — ISBN 978-5-9795-1564-9. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/165020 (дата обращения: 12.10.2021).

9. Клашанов, Ф. К. Вычислительные системы и сети, облачные технологии : учебно-методическое пособие / Ф. К. Клашанов. — Москва : МИСИ – МГСУ, 2020. — 40 с. — ISBN 978-5-7264-2187-2. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/145093 (дата обращения: 12.10.2021).

ПРИЛОЖЕНИЕ

Процесс соединения с кластером и запуска удаленного рабочего стола

1. Проверьте наличие соединения компьютера с Интернетом.

2. Для организации удаленного рабочего стола на клиентском компьютере используется программа «X2Go Client». Проверьте наличие этой программы на компьютере и установите ее, если она отсутствует. Последнюю версию установщика программы «X2Go Client» можно скачать с сайта:

http://wiki.x2go.org/doku.php/doc:installation:x2goclient

3. Запустите программу «X2Go Client», и создайте новую сессию, нажав на соответствующую кнопку, показанную на рисунке. Если существующих сессий еще нет, окно запустится автоматически.

4. Задайте настройки новой сессии. На вкладке «Сессия» заполните следующие поля:

— «Имя сессии»: Любое имя сессии, желательно осмысленное

— «Хост»: node39

— «Пользователь»: имя пользователя кластера

— «SSH порт»: 22

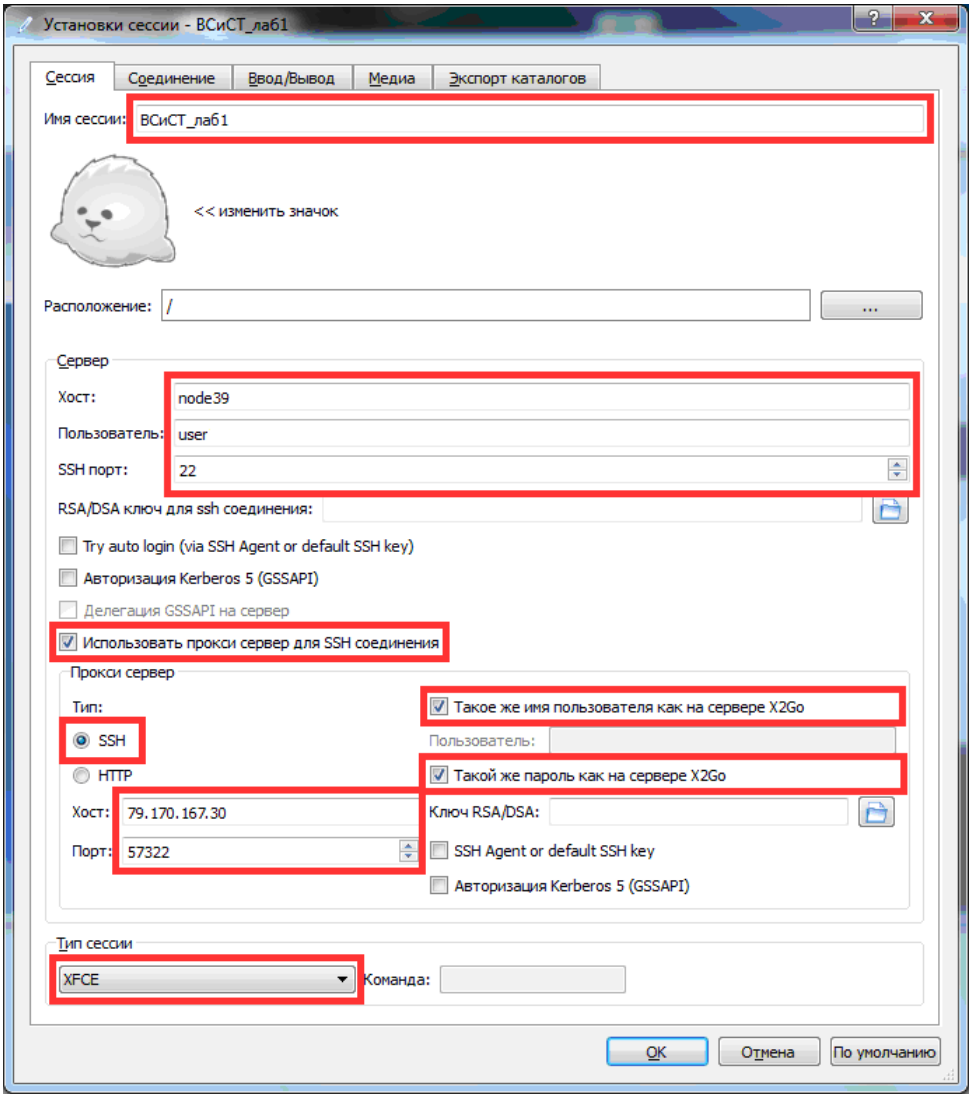
— «Тип сессии»: XFCE

Включите использование **SSH-прокси** с параметрами:

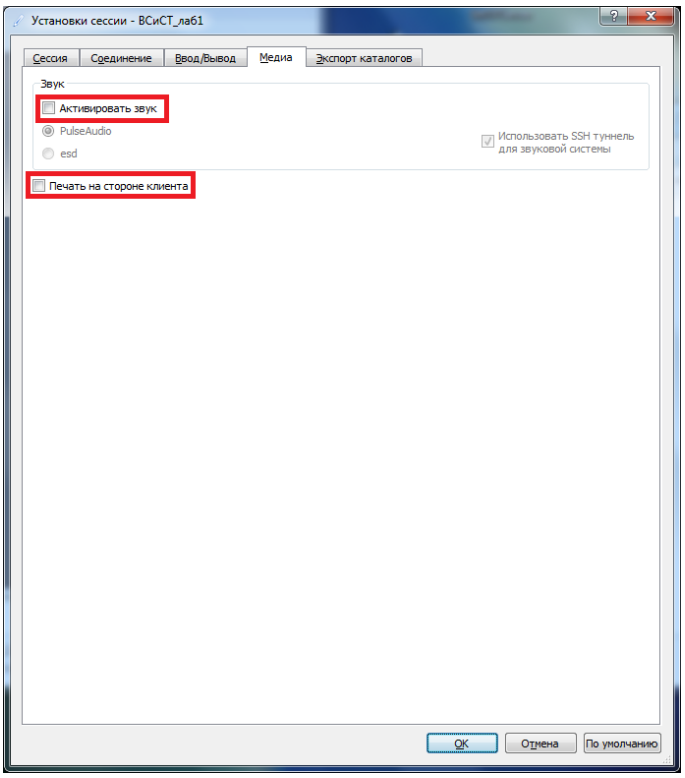
— «Хост»: `79.170.167.30`

— «Порт»: `57322`

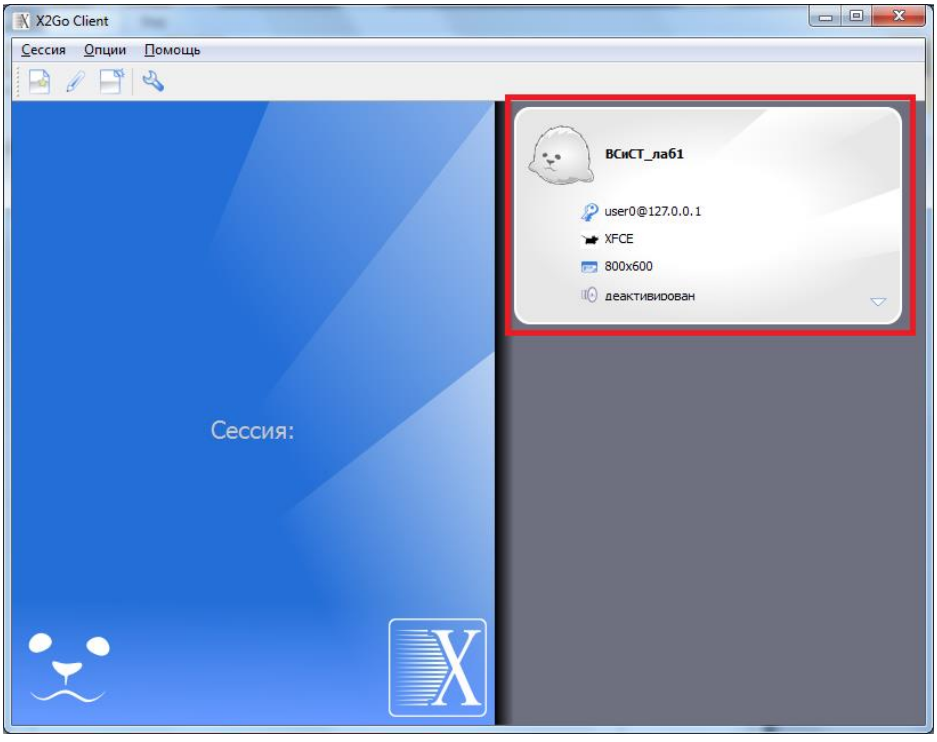
Укажите системе использовать на прокси-сервере имя пользователя и пароль **такие же**, как и на сервере X2Go.



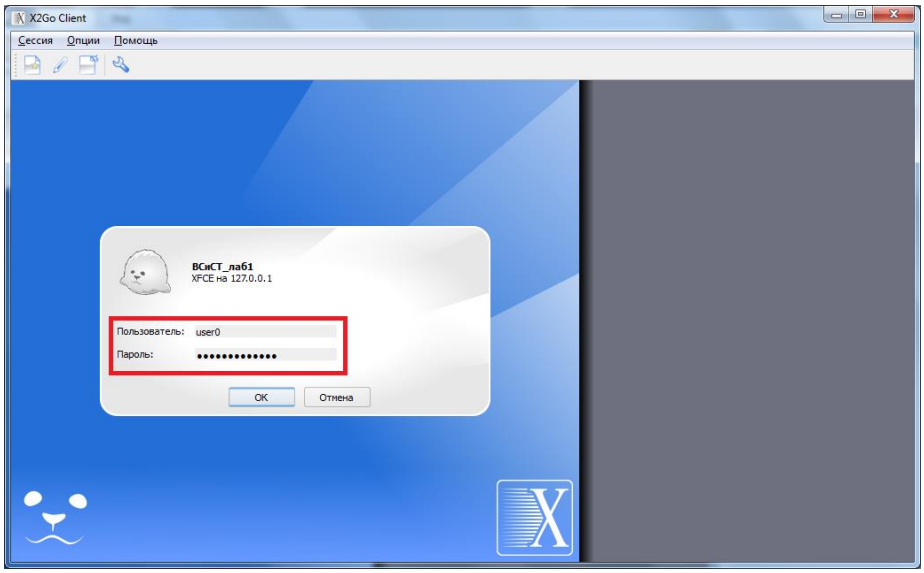
5. Переключитесь на вкладку «Медиа», на которой отключите звук и печать, которые не используются в лабораторной работе.



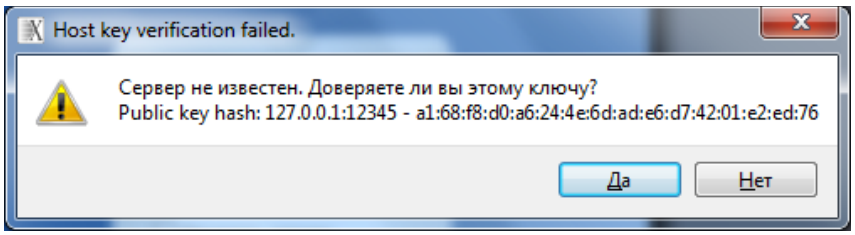
6. Сохраните все выполненные настройки, нажав на кнопку «OK». Новая созданная сессия будет доступна в правой части окна клиента X2Go. Показанное расширение окна не является принципиальным, его можно легко изменить уже после установки соединения. Чтобы запустить сессию, щелкните по ней мышкой.



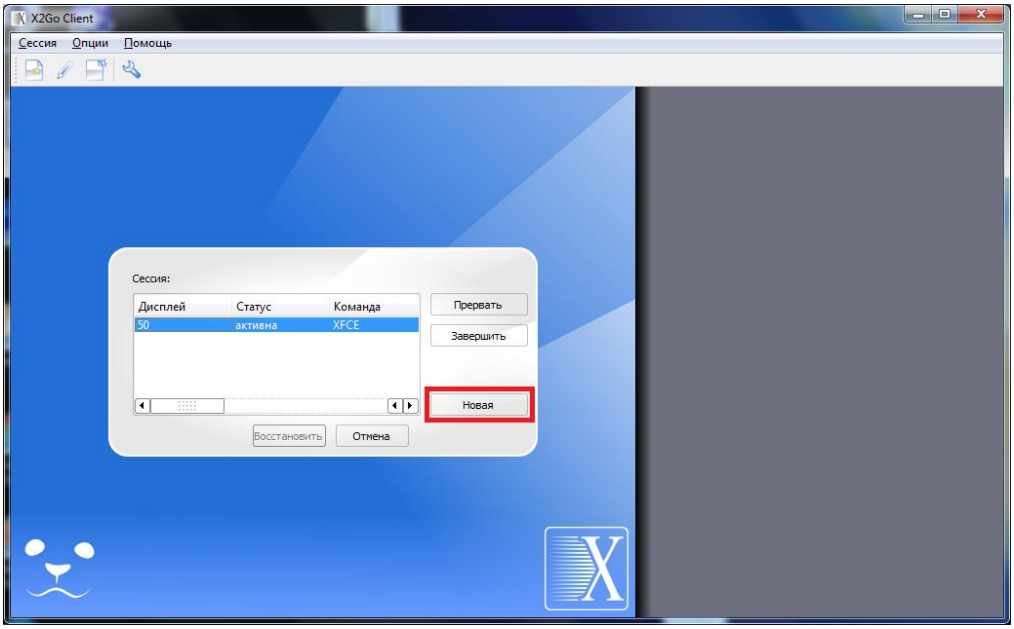
7. В появившемся окне введите пароль пользователя, выданный системным администратором, и нажмите «OK». Запустится процесс установки соединения.



8. Если соединение с кластером с данного клиентского компьютера производится *впервые*, система безопасности ключей выдаст запрос о доверии ключу. Для продолжения соединения на него нужно ответить утвердительно. **(Запросы могут быть неоднократными!)**



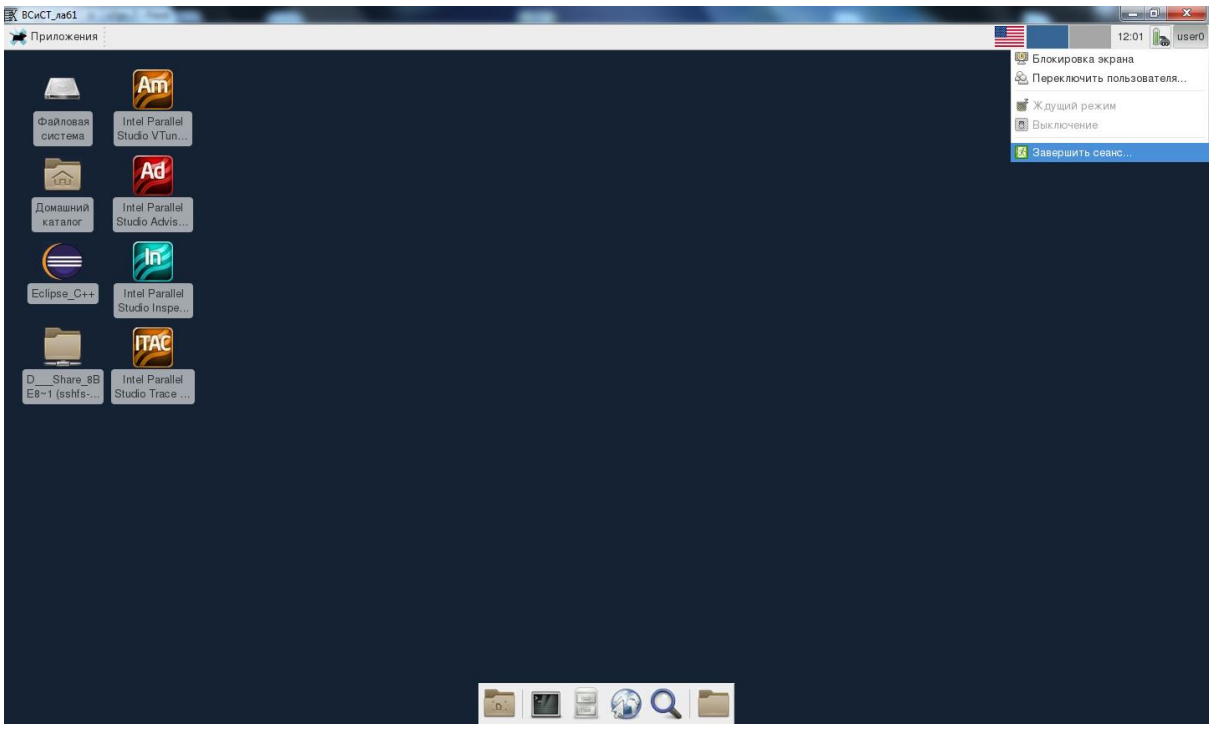
9. Система X2Go позволяет соединяться с сервером, каждый раз запуская новый сеанс графической оболочки. Это позволяет студентам с одинаковой учетной записью работать относительно независимо друг от друга в своей графической среде. **Если на момент соединения у сервера уже есть запущенные пользователем с данной учетной записью сессии**, то система запросит дальнейшие действия у пользователя. В ответ нужно создать **новую сессию**, не мешая работать другим пользователям.



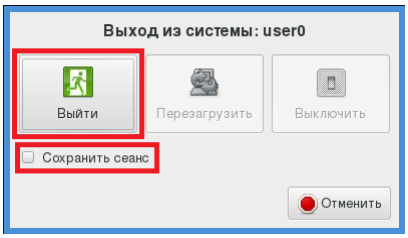
10. Дождитесь установки соединения. Если соединение прошло успешно, появится удаленный рабочий стол вычислительного кластера, с которым можно работать так же, как и на локальной системе. Директория, указанная в настройках клиента X2Go при конфигурировании сессии, будет смонтирована на рабочем столе кластера.

Процесс завершения соединения

1. После окончания работы и сохранения всех результатов необходимо в правом верхнем углу удаленного рабочего стола нажать на кнопку с надписью учетной записи и выбрать пункт «Завершить сеанс…». **Если просто закрыть окно клиента, сессия станет неактивной, но не будет остановлена**, и потом к ней можно будет подключиться вновь и продолжить работу. **Однако старайтесь не допускать неактивных сессий**, так как при большом количестве пользователей на них впустую тратится оперативная память удаленного сервера. Если окно было закрыто, а сессия больше не нужна, соединитесь с кластером вновь и корректно завершите сессию.



2. В появившемся окне снимите отметку пункта «Сохранить сеанс» и нажмите кнопку «Выйти». Дождитесь завершения сессии и закройте окно клиента X2Go.



Процесс организации удаленной передачи файлов между кластером и локальным компьютером

1. «Слабым местом» системы x2go является организация монтирования локальных ресурсов. Часто пользовательская директория оказывается несмонтированной на сервере, что приводит к проблемам с передачей файлов на кластер. В этом случае, для организации обмена файлами с кластером можно использовать удаленный файловый менеджер. Для этого запустите программу WinSCP.exe. Заполните поля следующими данными:

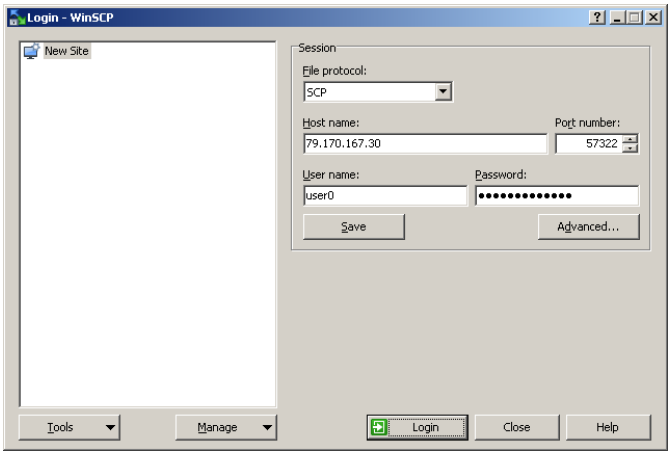
— «File protocol»: SCP

— «Host name»: 79.170.167.30

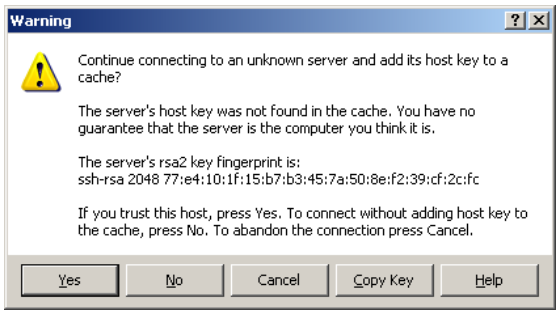
— «Port»: 57322

— «User name»: имя пользователя кластера

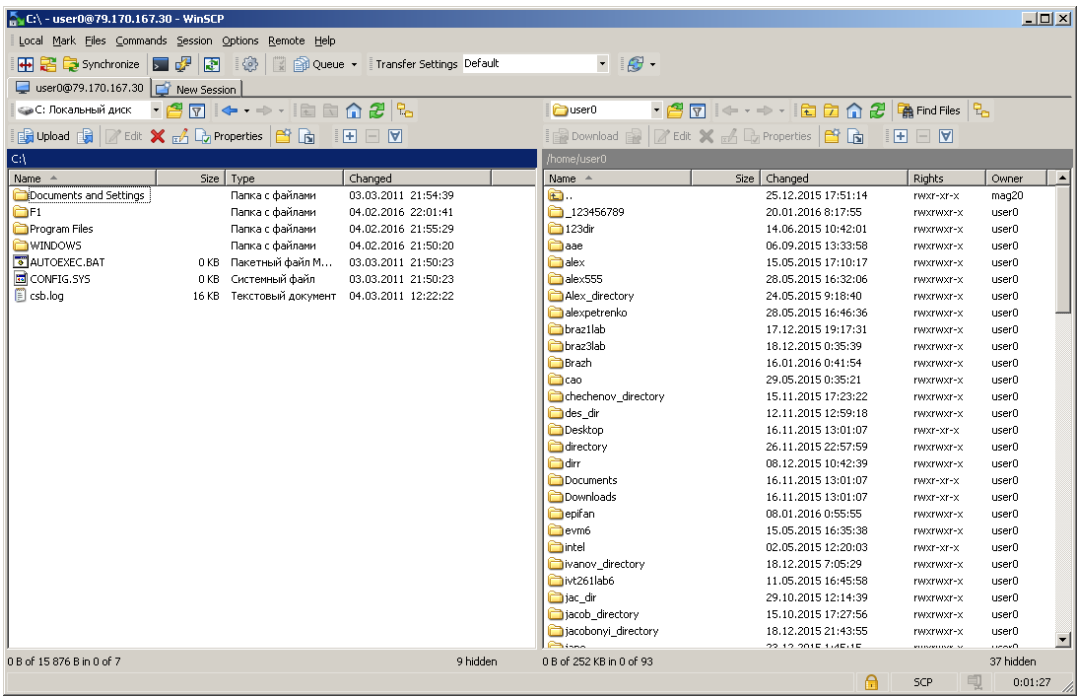
— «Password»: пароль пользователя кластера



2. Подтвердите ввод нажатием на кнопку «Login». Аналогично вышерассмотренной ситуации, на предупреждение системы безопасности следует ответить «Да»:



3. Если соединение прошло успешно, Вы увидите окно удаленного менеджера файлов. В левой половине окна представлена локальная файловая система (файлы и папки Вашего компьютера), а в правой половине — файловая система кластера. В Вашей домашней директории на кластере (она имеет вид /home/имя_Вашей_учетной_записи) Вы можете создавать свои папки, заходить в них, копировать файлы с Вашего компьютера на кластер и обратно, и т.д. Удобно для каждой задачи создать свою папку, перейти в нее и скопировать туда нужные файлы с локального компьютера.


C:\ - user0@79.170.167.30 - WinSCP
Local Mark Files Commands Session Options Remote Help
Synchronize Queue Transfer Settings Default
user0@79.170.167.30 New Session
C: Локальный диск
Upload Edit Properties
C:\
Name Size Type Changed
Documents and Settings Папка с файлами 03.03.2011 21:54:39
F1 Папка с файлами 04.02.2016 22:01:41
Program Files Папка с файлами 04.02.2016 21:55:29
WINDOWS Папка с файлами 04.02.2016 21:50:20
AUTOEXEC.BAT 0 KB Пакетный файл M... 03.03.2011 21:50:23
CONFIG.SYS 0 KB Системный файл 03.03.2011 21:50:23
csb.log 16 KB Текстовый документ 04.03.2011 12:22:22
0 B of 15 876 B in 0 of 7
9 hidden
user0
Find Files
Download Edit Properties
/home/user0
Name Size Changed Rights Owner
.. 25.12.2015 17:51:14 rwxr-xr-x mag20
_123456789 20.01.2016 8:17:55 rwxrwxr-x user0
123dir 14.06.2015 10:42:01 rwxrwxr-x user0
aae 06.09.2015 13:33:58 rwxrwxr-x user0
alex 15.05.2015 17:10:17 rwxrwxr-x user0
alex555 28.05.2015 16:32:06 rwxrwxr-x user0
Alex_directory 24.05.2015 9:18:40 rwxrwxr-x user0
alexpetrenko 28.05.2015 16:46:36 rwxrwxr-x user0
braz1lab 17.12.2015 19:17:31 rwxrwxr-x user0
braz3lab 18.12.2015 0:35:39 rwxrwxr-x user0
Brazh 16.01.2016 0:41:54 rwxrwxr-x user0
cao 29.05.2015 0:35:21 rwxrwxr-x user0
chechenov_directory 15.11.2015 17:23:22 rwxrwxr-x user0
des_dir 12.11.2015 12:59:18 rwxrwxr-x user0
Desktop 16.11.2015 13:01:07 rwxr-xr-x user0
directory 26.11.2015 22:57:59 rwxrwxr-x user0
dirr 08.12.2015 10:42:39 rwxrwxr-x user0
Documents 16.11.2015 13:01:07 rwxr-xr-x user0
Downloads 16.11.2015 13:01:07 rwxr-xr-x user0
epifan 08.01.2016 0:55:55 rwxrwxr-x user0
evm6 15.05.2015 16:35:38 rwxrwxr-x user0
intel 02.05.2015 12:20:03 rwxr-xr-x user0
ivanov_directory 18.12.2015 7:05:29 rwxrwxr-x user0
ivt261lab6 11.05.2015 16:45:58 rwxrwxr-x user0
jac_dir 29.10.2015 12:14:39 rwxrwxr-x user0
jacob_directory 15.10.2015 17:27:56 rwxrwxr-x user0
jacobonyi_directory 18.12.2015 21:43:55 rwxrwxr-x user0
0 B of 252 KB in 0 of 93
37 hidden
SCP
0:01:27

Учебное издание

Павел Дмитриевич Кравченя

ОБЛАЧНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ

Учебное пособие

Волгоградский государственный технический университет.
400005, г. Волгоград, просп. В. И. Ленина, 28, корп. 1.