

Севастопольский государственный университет

Методические указания
к выполнению лабораторных работ
по дисциплине «Основы облачных
технологий»

А.Р. Умеров¹

Е.Н. Мащенко²

22 июня 2016 г.³

¹admin@amet13.name

²elmachenko@mail.ru

³Дата последней правки документа

Содержание

1	Лабораторная работа №1	3
1.1	Теоретические основы виртуализации	3
1.2	Порядок выполнения работы	7
1.3	Контрольные вопросы	7
2	Лабораторная работа №2	8
2.1	Теоретические основы облачных вычислений	8
2.2	Порядок выполнения работы	10
2.3	Контрольные вопросы	10
3	Лабораторная работа №3	11
3.1	Краткие сведения о методологиях разработки ПО	11
3.2	Порядок выполнения работы	12
3.3	Контрольные вопросы	12
A	Пример создания виртуальной машины в Oracle VM VirtualBox	13
B	Пример установки Debian GNU/Linux в VirtualBox	20
C	Пример установки ownCloud Server в Debian GNU/Linux	32
D	Настройка подключения ownCloud Client к серверу	36
E	Пример деплоя приложения на Heroku	39

1 Лабораторная работа №1.

Знакомство с виртуализацией. Система виртуализации Oracle VM VirtualBox

Цель работы: ознакомиться с основными понятиями виртуализации, системой виртуализации VirtualBox, научиться настраивать виртуальную машину (ВМ), совершать простейшие операции с ней, устанавливать операционную систему на ВМ.

1.1 Теоретические основы виртуализации

Виртуализация — абстракция вычислительных ресурсов и предоставление пользователю системы, которая инкапсулирует (скрывает в себе) собственную реализацию.

Виртуализацию можно использовать в:

- консолидации серверов (позволяет мигрировать с физических серверов на виртуальные, тем самым увеличивается коэффициент использования аппаратуры, что позволяет существенно сэкономить на аппаратуре, электроэнергии и обслуживании);
- разработке и тестировании приложений (возможность одновременно запускать несколько различных ОС, это удобно при разработке кроссплатформенного ПО, тем самым значительно повышается качество, скорость разработки и тестирования приложений);
- бизнесе (использование виртуализации в бизнесе растет с каждым днем и постоянно находятся новые способы применения этой технологии, например, возможность безболезненно сделать снапшот¹ и быстро восстановить систему в случае сбоя);
- организации виртуальных рабочих станций (так называемых «тонких клиентов»²).

Общая схема взаимодействия виртуализации с аппаратурой и программным обеспечением (ПО) представлена на рис. 1.

Взаимодействие приложений и операционной системы (ОС) с аппаратным обеспечением осуществляется через абстрагированный слой виртуализации.

Существует несколько подходов организации виртуализации:

- эмуляция оборудования (QEMU, Bochs, Dynamips);
- полная виртуализация (KVM, HyperV, VirtualBox);
- паравиртуализация (Xen, L4, Trango);
- виртуализация уровня ОС (LXC, OpenVZ, Jails, Solaris Zones).

¹Снапшот (англ. snapshot) — снимок состояния ВМ в определенный момент времени. Сюда входят настройки ВМ, содержимое памяти и дисков

²Тонкий клиент (англ. thin client) — бездисковый компьютер-клиент в сетях с клиент-серверной или терминальной архитектурой, который переносит все или большую часть задач по обработке информации на сервер

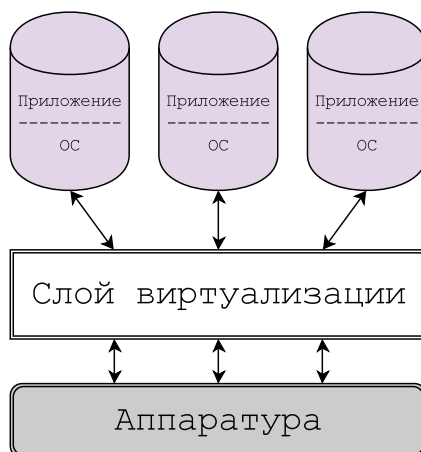


Рис. 1: Схема взаимодействия виртуализации с аппаратурой и ПО

Эмуляция аппаратных средств является одним из самых сложных методов виртуализации (рис. 2). В то же время главной проблемой при эмуляции аппаратных средств является низкая скорость работы, в связи с тем, что каждая команда моделируется на основных аппаратных средствах. В эмуляции оборудования используется механизм динамической трансляции, то есть каждая из инструкций эмулируемой платформы заменяется на заранее подготовленный фрагмент инструкций физического процессора.

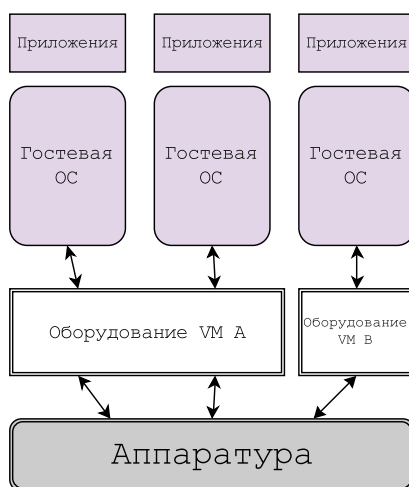


Рис. 2: Эмуляция оборудования моделирует аппаратные средства

В случае полной виртуализации поверх уже установленной ОС, устанавливается программа-гипервизор¹, которая осуществляет взаимосвязь между гостевыми ОС и хост-компьютером (рис. 3).

Преимуществом технологии полной виртуализации является установка различных ОС, а недостатком — меньшая производительность, за счет накладных расходов на гипервизор, а также понижение скорости работы с подсистемой ввода/вывода из-за необходимости изоляции.

¹Гипервизор (англ. hypervisor)— программа или аппаратная схема, позволяющая одновременное, параллельное выполнение нескольких ОС на одном и том же компьютере, обеспечивает изоляцию операционных систем друг от друга

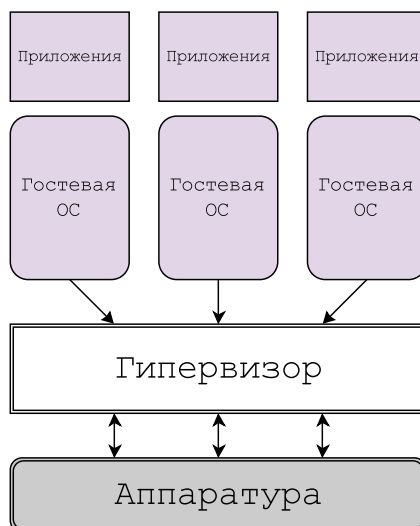


Рис. 3: Полная виртуализация использует гипервизор

Паравиртуализация имеет некоторые сходства с полной виртуализацией. В данном методе также используется гипервизор для разделения доступа к аппаратуре, но объединяется код, касающийся виртуализации, в ОС (рис. 4).

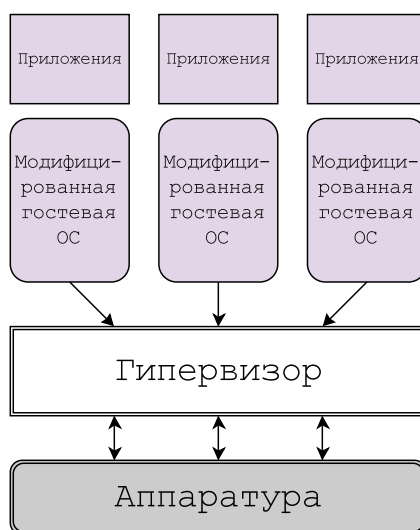


Рис. 4: Паравиртуализация разделяет процесс с гостевой ОС

Недостатком паравиртуализации является необходимость изменения гостевой ОС для гипервизора, однако таким образом гораздо увеличивается производительность.

Виртуализация уровня операционной системы не нуждается в гипервизоре. Для ее работы необходимо модифицированное ядро на хост-системе с набором патчей и утилит для управления контейнерами¹ (рис. 5).

За счет того, что контейнер напрямую взаимодействует с ядром, а не через гипервизор, обеспечивается максимальное быстродействие. Но, так как для всех контейнеров используется общее ядро, то нет возможности использовать разные ОС в контейнерах.

¹ Контейнер или VPS/VDS (англ. Virtual Private/Dedicated Server) — виртуальный выделенный сервер, эмулирует работу физического сервера

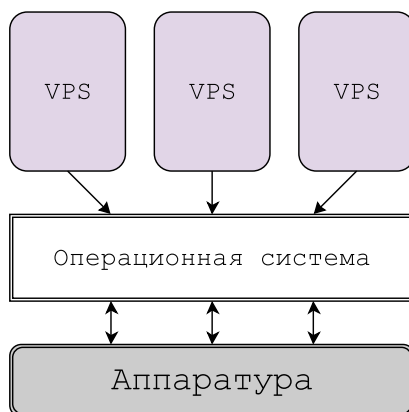


Рис. 5: Виртуализация уровня ОС изолирует серверы

Oracle VM VirtualBox — система виртуализации, разработанная компанией Innotek в 2007 году, позже приобретена компанией Sun Microsystems. Ключевыми возможностями системы является кроссплатформенность, наличие графического интерфейса, локализация, поддержка аппаратной виртуализации, экспериментальное 3D-ускорение, поддержка различных образов жестких дисков, возможность установки дополнений гостевой ОС, например для корректной работы проброшенных USB-устройств или возможности изменения разрешения рабочего стола гостевой ОС.

На рис. 6 изображен пример одновременной работы двух виртуальных машин (Windows 7 и CentOS 7).

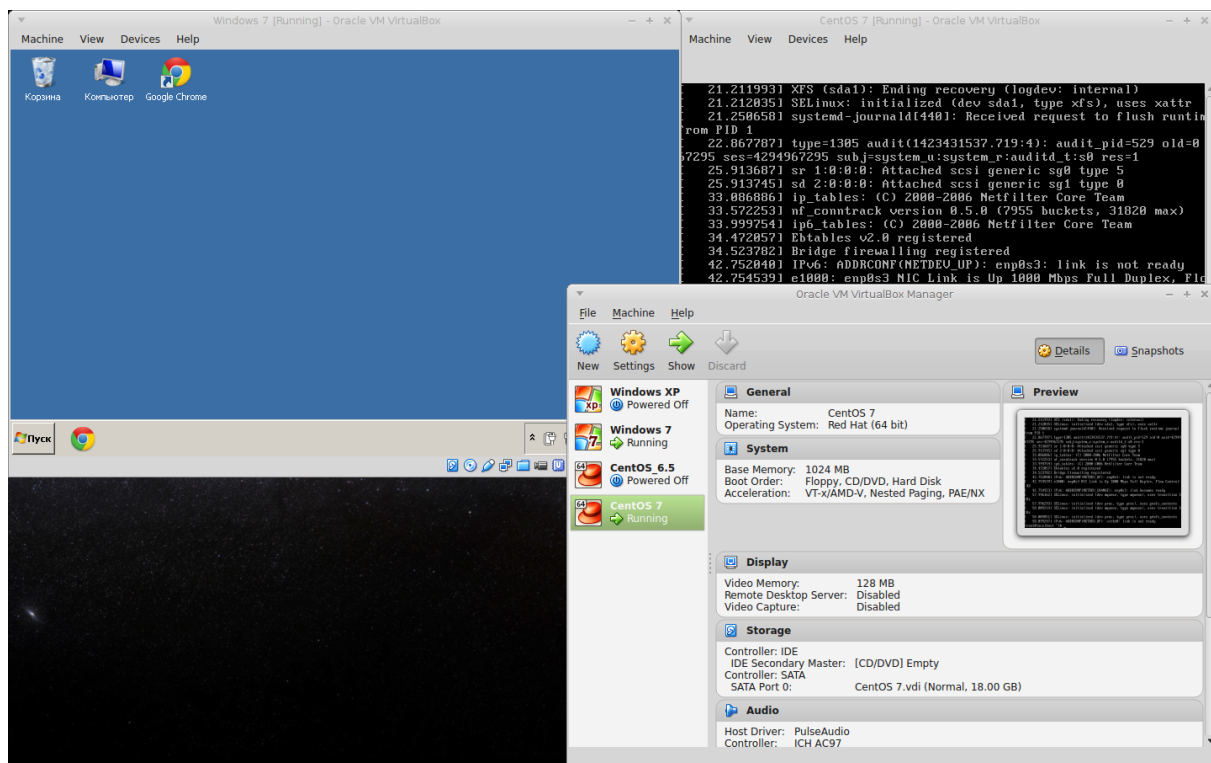


Рис. 6: Пример одновременной работы Windows 7 и CentOS 7

1.2 Порядок выполнения работы

1. Скачать¹ и установить Oracle VM VirtualBox;
2. Изучить интерфейс программы и создать² первую виртуальную машину на основе образа Debian GNU/Linux³;
3. Подключить к виртуальной машине ранее скачанный образ Debian GNU/Linux;
4. Загрузиться с подключенного образа и установить⁴ дистрибутив на виртуальный жесткий диск;
5. Во время установки дистрибутива необходимо будет задать пароль суперпользователя, пароль: `toor`, пароль для локального пользователя можно задать произвольный, но лучше его запомнить;
6. С помощью команд `free`, `df`⁵, `cat /proc/cpuinfo`, `ifconfig`, посмотреть параметры виртуальной машины;
7. С помощью команды `ping` проверить доступность виртуальной машины в сети как с хост-ноды, так и с других компьютеров сети;
8. С помощью SSH-клиента (например Putty⁶) подключиться по SSH к виртуальной машине.

1.3 Контрольные вопросы

1. Какие типы виртуализации Вы знаете? В чем между ними различия?
2. К какому типу виртуализации относится система Oracle VM VirtualBox?
3. Как может использовать виртуализацию в работе веб-программист, системный администратор, сетевой инженер?
4. В чем состоит основное преимущество и недостаток полной виртуализации от контейнерной?
5. В чем различие между режимами сети NAT и сетевой мост (Network Bridge)?

¹<https://www.virtualbox.org/wiki/Downloads>

²Пример создания виртуальной машины описан в прил. А

³<https://www.debian.org/distrib/>

⁴Процесс установки Debian GNU/Linux описан в прил. В

⁵Команды `free` и `df` используют ключ `-h` для показа информации в удобном для человека виде

⁶<http://www.chiark.greenend.org.uk/~sgtatham/putty/>

2 Лабораторная работа №2.

Модель обслуживания SaaS (Software-as-a-Service) на примере развертывания облачного хранилища ownCloud

Цель работы: ознакомиться с основными моделями представления облачных услуг (SaaS, PaaS, IaaS), развернуть собственное облачное хранилище, доступное пользователям в пределах локальной сети.

2.1 Теоретические основы облачных вычислений

Облачные вычисления («облака», Cloud computing) — модель предоставления вычислительных ресурсов, охватывающая все, начиная от приложений и до центров обработки данных (ЦОД), через Интернет при условии оплаты за фактическое использование.

Одной из первых, кто стал внедрять услугу облачных вычислений стала компания Amazon, в то время (2002 г.) она еще являлась книжным Интернет-магазином, который впоследствии перерос, благодаря этим внедрениям, в одну из мощнейших технологических компаний. Уже в 2006 г. был запущен проект под названием Computing Cloud (Amazon EC2), после этого в 2009 г. компания Google представила Google Apps. После этих событий были сформированы общие понятия об облачных вычислениях, в частности выделены наиболее важные модели обслуживания и модели развертывания.

Облачные вычисления являются следующим шагом в эволюции архитектуры построения информационных систем. Благодаря преимуществам данного подхода вполне очевидно, что многие информационные системы в ближайшее время переносятся или будут перенесены в облако. Процесс уже идет полным ходом и его игнорирование или недооценка может привести к поражению в конкурентной борьбе на рынке. В данном случае имеется в виду не только отставание ИТ или неоправданные затраты на него, но и отставание в развитии бизнеса компании, которая зависит от гибкости информационной инфраструктуры и скорости вывода новых сервисов и продуктов на рынок.

Различают три основные модели обслуживания:

1. программное обеспечение как услуга (Software-as-a-Service, SaaS);
2. платформа как услуга (Platform-as-a-Service, PaaS);
3. инфраструктура как услуга (Infrastructure-as-a-Service, IaaS).

SaaS — модель, при которой не требуется приобретать, устанавливать, обновлять и поддерживать ПО, эту задачу берет на себя поставщик услуги. Кроме того, осуществить регистрацию и использование облачных приложений можно немедленно, приложения и данные приложений доступны с любого устройства, подключенного к Интернету. При поломке устройства данные не теряются, они хранятся в облаке, пользователю, как правило, доступны локальные настройки конфигурации приложения. Примерами моделей SaaS могут служить: почтовая служба Gmail, Skype, CRM (система управления взаимоотношения с клиентами) и ERP (планирование ресурсов предприятия) системы и другие.

РaaS — модель, при которой пользователю предоставляется возможность использования облачной инфраструктуры для размещения базового ПО. В таком случае конфигурирование программного обеспечения целиком ложится на пользователя, предоставляется только платформа для развертывания ПО. Примером модели РaaS является предоставление услуги развертывания собственного ПО в рамках облачной инфраструктуры. Например Heroku, OpenShift.

IaaS — модель, при которой пользователю доступно полное управление облаком в рамках операционной системы. Потребитель обладает контролем над операционными системами, сетевыми сервисами. Данная модель подходит задачам, для которых характерно быстрое изменение нагрузки. Пользователю предоставляется виртуализированное окружение, как правило с «чистой» операционной системой, пригодной для развертывания любого приложения. Примеры: Digital Ocean, Microsoft Azure, Google App Engine и т.д;

Модели развертывания облака можно разделить на четыре вида:

1. частное облако (private cloud);
2. публичное облако (public cloud);
3. общественное облако (community cloud);
4. гибридное облако (hybrid cloud).

Частное облако предназначено для использования одной организацией, публичное — для широкой публики, общественное, как правило для сообщества или организации, гибридное облако состоит из двух или более различных облачных инфраструктур.

Основными преимуществами использования облачных технологий являются:

- снижение расходов на закупку оборудования и построения центров обработки данных (ЦОД);
- удобство использования приложений с большого количества устройств, в том числе и мобильных;
- обеспечение надежности хранения данных, производительности приложений, за счет простоты использования ПО, мониторинга, балансировки нагрузки, миграции данных и т.д.

ownCloud — свободное веб-приложение, предназначенное для синхронизации данных между сервером и клиентами, как правило данными являются документы и медиаконтент. ownCloud является альтернативой таким облачным сервисам как Dropbox, Google Drive, Яндекс Диск, MEGA и др. Отличие состоит в том, что приложение можно развернуть на собственном сервере как для домашнего использования, так и для использования в организациях. Так как приложение распространяется бесплатно, имеет открытый исходный код¹ и периодически обрастает новыми функциями (планировщик задач, календарь, фотогалерея, просмотрщик документов, гибкая аутентификация пользователей и другие) проект обрел популярность как у пользователей, так и у Open-source разработчиков.

Установка ownCloud происходит в несколько простых шагов. ownCloud Server доступен для платформ Windows и Linux, ownCloud Client также доступен для Windows, Linux, а также Mac.

¹<https://github.com/owncloud/>

2.2 Порядок выполнения работы

Выполнять работу рекомендуется группами студентов по 3-5 человек. В качестве сервера может использоваться виртуальная машина с установленным дистрибутивом Debian, ранее установленная в лабораторной работе №1. Виртуальная машина должна иметь доступ в сеть Интернет для скачивания нужных пакетов.

1. Установить¹ ownCloud Server в виртуальную машину;
2. На хост-машине или на другом компьютере сети скачать² и установить ownCloud Client;
3. Подключить ownCloud Client к серверу во время первого запуска клиента, а также синхронизировать все данные с сервера³;
4. Проверить работу синхронизации данных между клиентом и сервером (создать, удалить, переместить файл или каталог);
5. Предоставить публичную ссылку на файл или каталог и проверить его доступность в пределах локальной сети;
6. Ознакомиться с дополнительными возможностями ownCloud.

2.3 Контрольные вопросы

1. Каковы основные преимущества использования облачных технологий?
2. В чем состоит отличие SaaS от PaaS и IaaS?
3. В чем состоят преимущества ownCloud по сравнению с Dropbox или другими облачными хранилищами? Недостатки?
4. Почему Skype можно отнести к модели SaaS?

¹Пример установки ownCloud Server представлен в прил. С

²<https://owncloud.org/install/>

³Скриншоты настройки ownCloud Client представлены в прил. D

3 Лабораторная работа №3.

Модель обслуживания PaaS (Platform-as-a-Service) на примере деплоя приложения в Heroku

Цель работы: ознакомиться с моделью гибкой (agile) разработки программного обеспечения, развернуть приложение на Heroku, ознакомиться с системой контроля версий Git.

3.1 Краткие сведения о методологиях разработки ПО

За время существования информационных технологий создавались и изменялись подходы к построению информационных систем. Первой моделью информационной системы была монолитная архитектура. В данной модели на одном компьютере работали и приложения и база данных (БД), а пользователи сидели у «тонких» терминалов которые отображали информацию с компьютера.

У данной архитектуры было большое количество недостатков, поэтому впоследствии ее сменила более перспективная клиент-серверная архитектура. В этом случае на компьютере располагался выделенный сервер баз данных, а пользователи с «толстых клиентов» разгружали сервер БД.

Затем появилась более современная многоуровневая архитектура, у которой логика приложений вынесена на отдельный компьютер, который называется сервер приложений, а пользователи работали на «тонких» клиентах через веб-браузеры. В современном информационном мире большинство приложений выполнено именно в многоуровневой архитектуре. Она подразумевает развертывание всей ИТ-инфраструктуры на территории заказчика.

Организация процесса разработки программного обеспечения также претерпела много изменений с течением времени. Одна из самых старых методологий — каскадная (водопадная), подразумевает последовательное прохождение стадий, каждая из которых должна завершиться полностью до начала следующей. В этой модели легко управлять проектом. Благодаря ее жесткости, разработка проходит быстро, стоимость и срок заранее определены.

В «гибкой» (Agile) методологии разработки после каждой итерации заказчик может наблюдать результат и понимать, удовлетворяет он его или нет. Это одно из преимуществ гибкой модели. К ее недостаткам относят то, что из-за отсутствия конкретных формулировок результатов сложно оценить трудозатраты и стоимость, требуемые на разработку. Экстремальное программирование (XP) является одним из наиболее известных применений гибкой модели на практике.

В последнее время все большую популярность приобретает именно Agile-методология, в которой необходимо на каждой итерации разработки проводить развертку новой версии ПО, гораздо удобнее это делать с помощью PaaS-решений, таких как Heroku.

Heroku является облачной PaaS-платформой, поддерживающей ряд языков программирования, таких как Java, Node.js, Scala, Clojure, Python и PHP. В Heroku можно легко развертывать (деплоить) проекты, не беспокоясь о развертывании собственной инфраструктуры для одного приложения.

Приложения, работающие на Heroku, используют также DNS-сервер Heroku. Для каждого приложения выделяется несколько независимых виртуальных процессов,

которые называются «dynos». Они распределены по специальной виртуальной сетке «dynos grid», которая состоит из нескольких серверов. Heroku также поддерживает систему контроля версий Git, а также подключение к аккаунту GitHub.

3.2 Порядок выполнения работы

В качестве сервера может использоваться виртуальная машина с установленным дистрибутивом Debian, ранее установленная в лабораторной работе №1. Виртуальная машина должна иметь доступ в сеть Интернет для скачивания нужных пакетов и работы с Heroku.

1. Зарегистрироваться в Heroku¹;
2. Создать тестовое приложение на языке Python;
3. Задеплоить приложение на Heroku;
4. Внести изменения в исходный код приложения и снова задеплоить приложение;
5. Ознакомиться с дополнительными возможностями Heroku и Git.

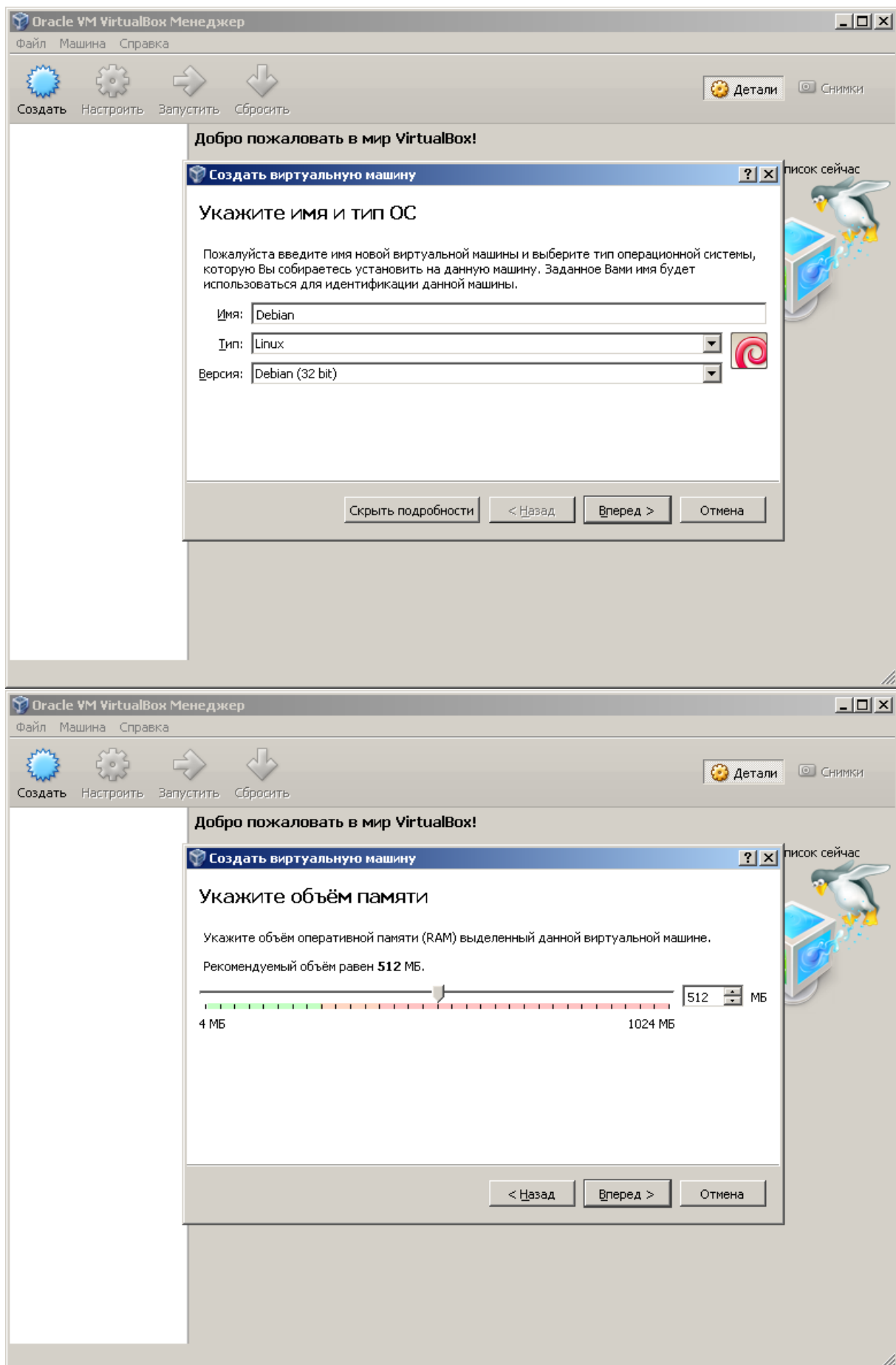
3.3 Контрольные вопросы

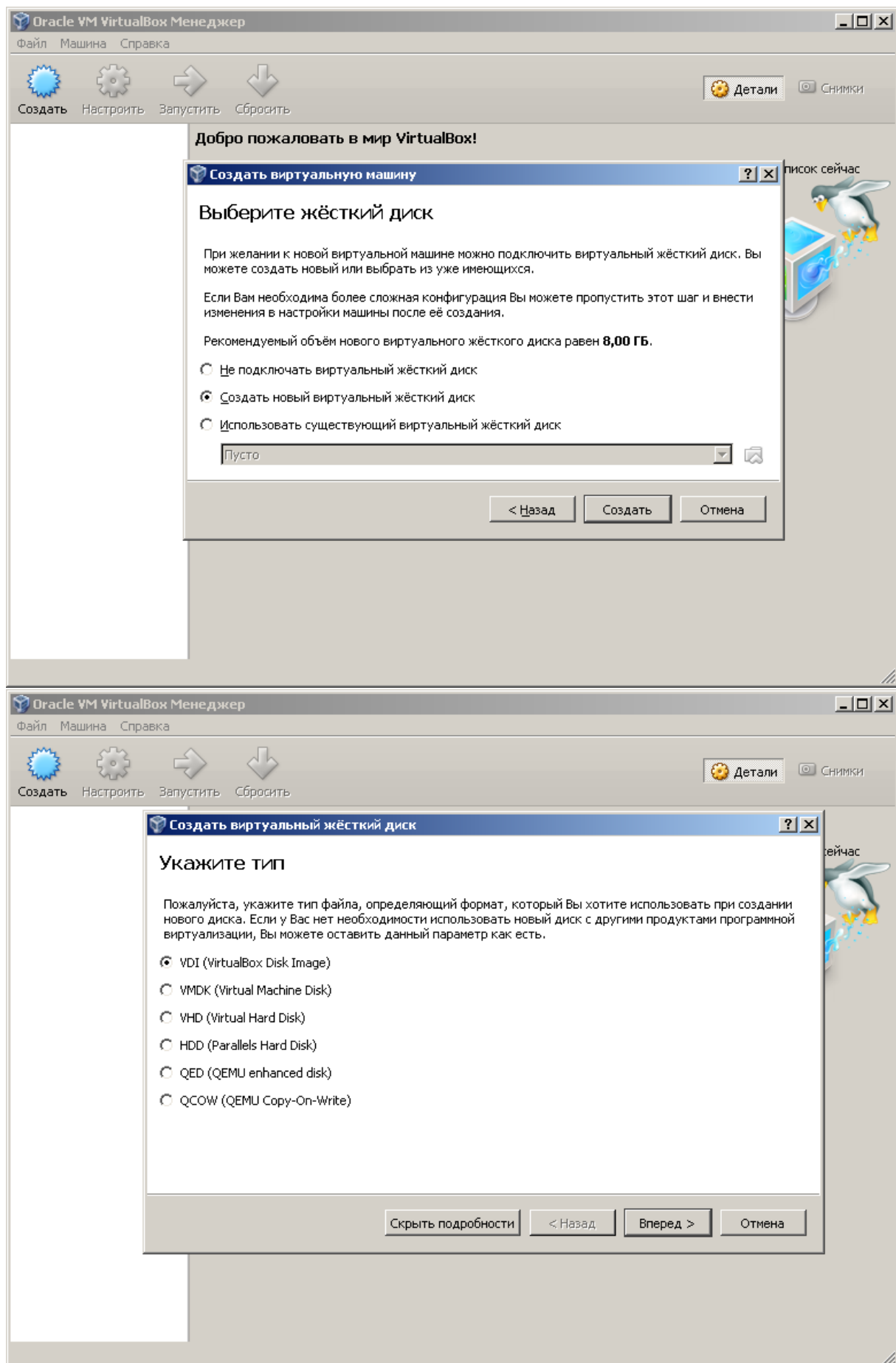
1. Почему методология гибкой разработки ПО становится все более популярной?
2. В чем преимущество использования PaaS-решений по сравнению с IaaS для развертывания своего ПО?
3. Для чего необходима команда commit в Git?
4. В чем преимущество использования Heroku по сравнению с деплоем в локальном окружении?

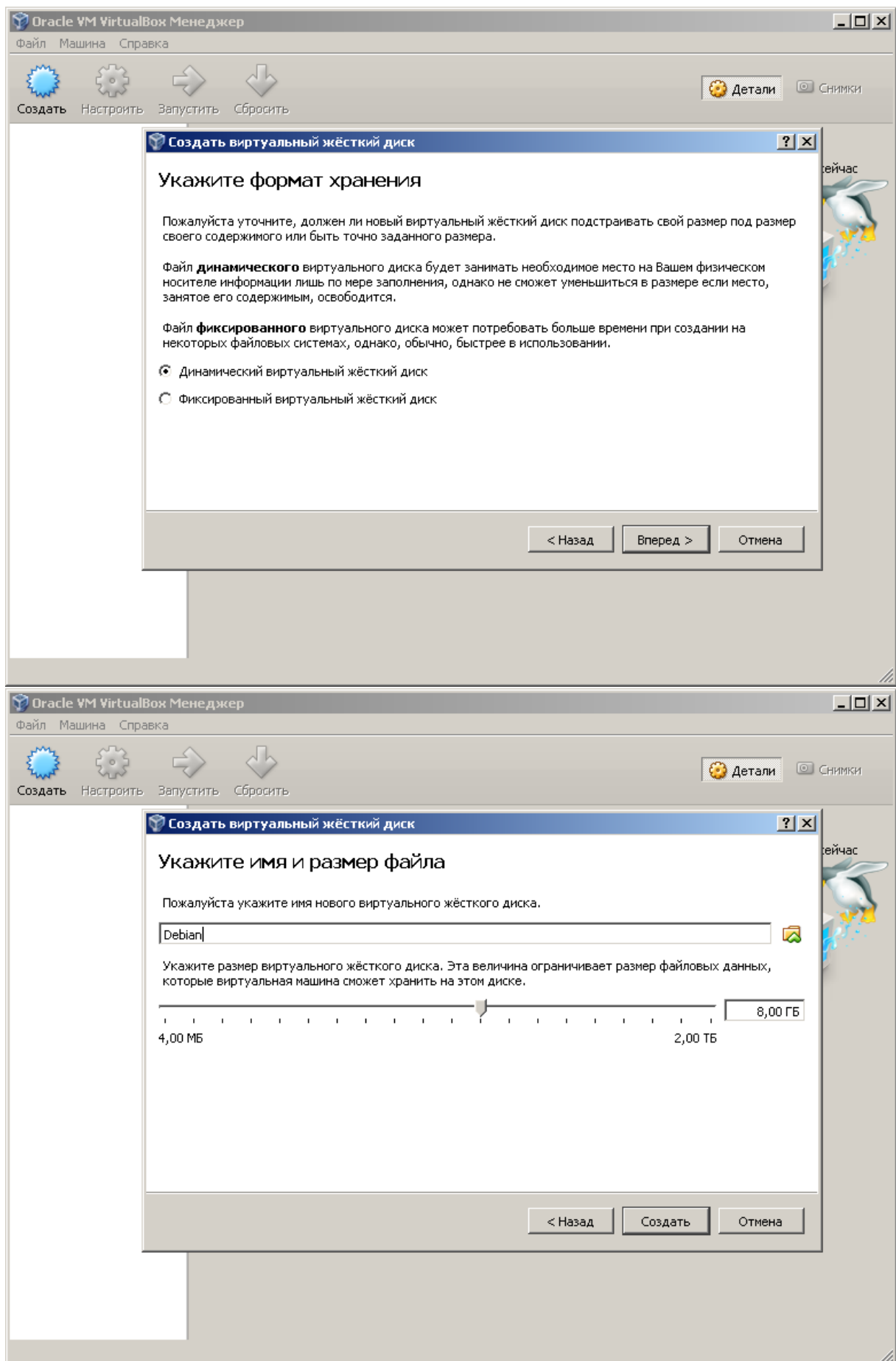
¹Пример работы с Heroku представлен в прил. Е

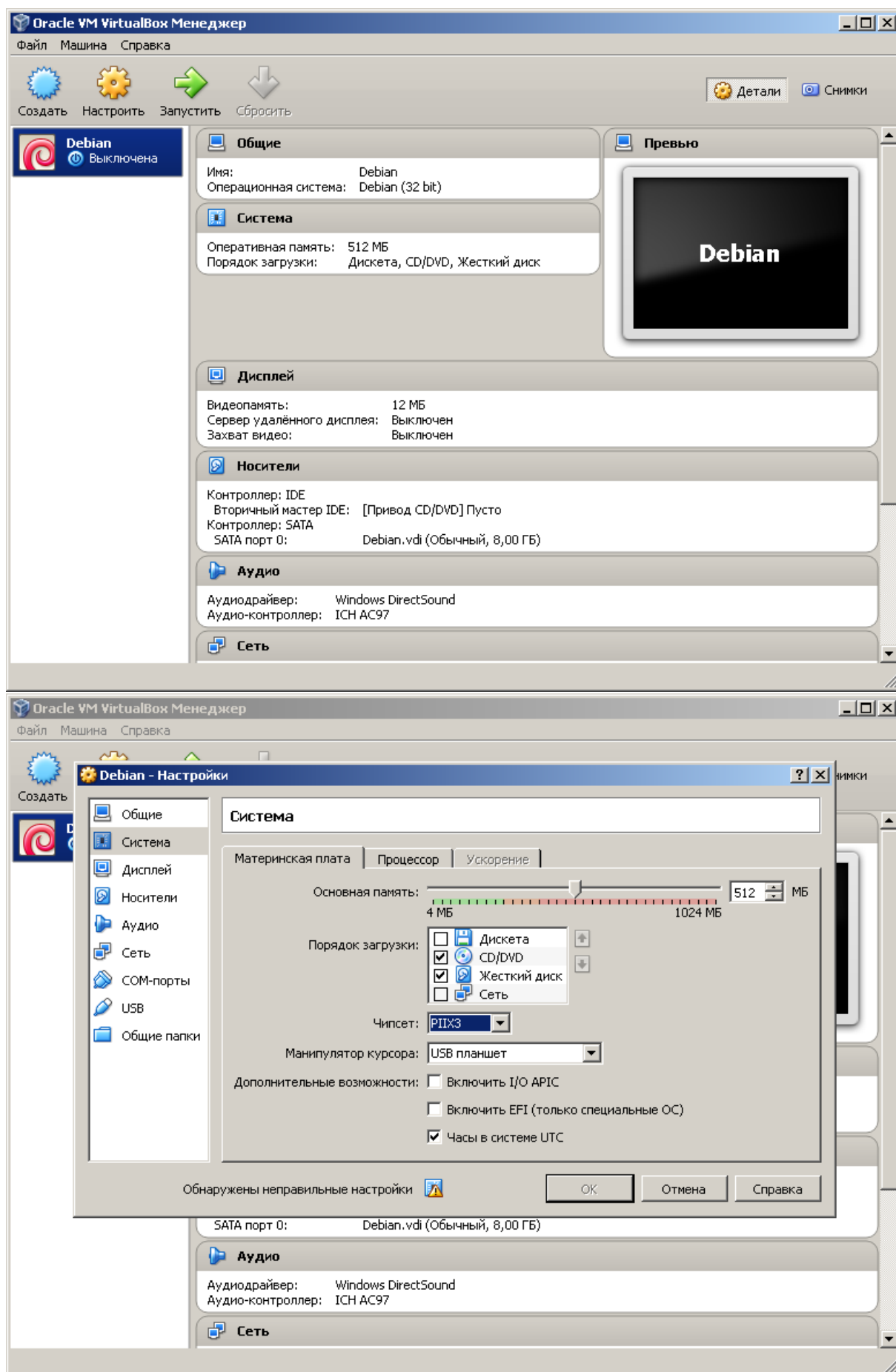
А Пример создания виртуальной машины в Oracle VM VirtualBox

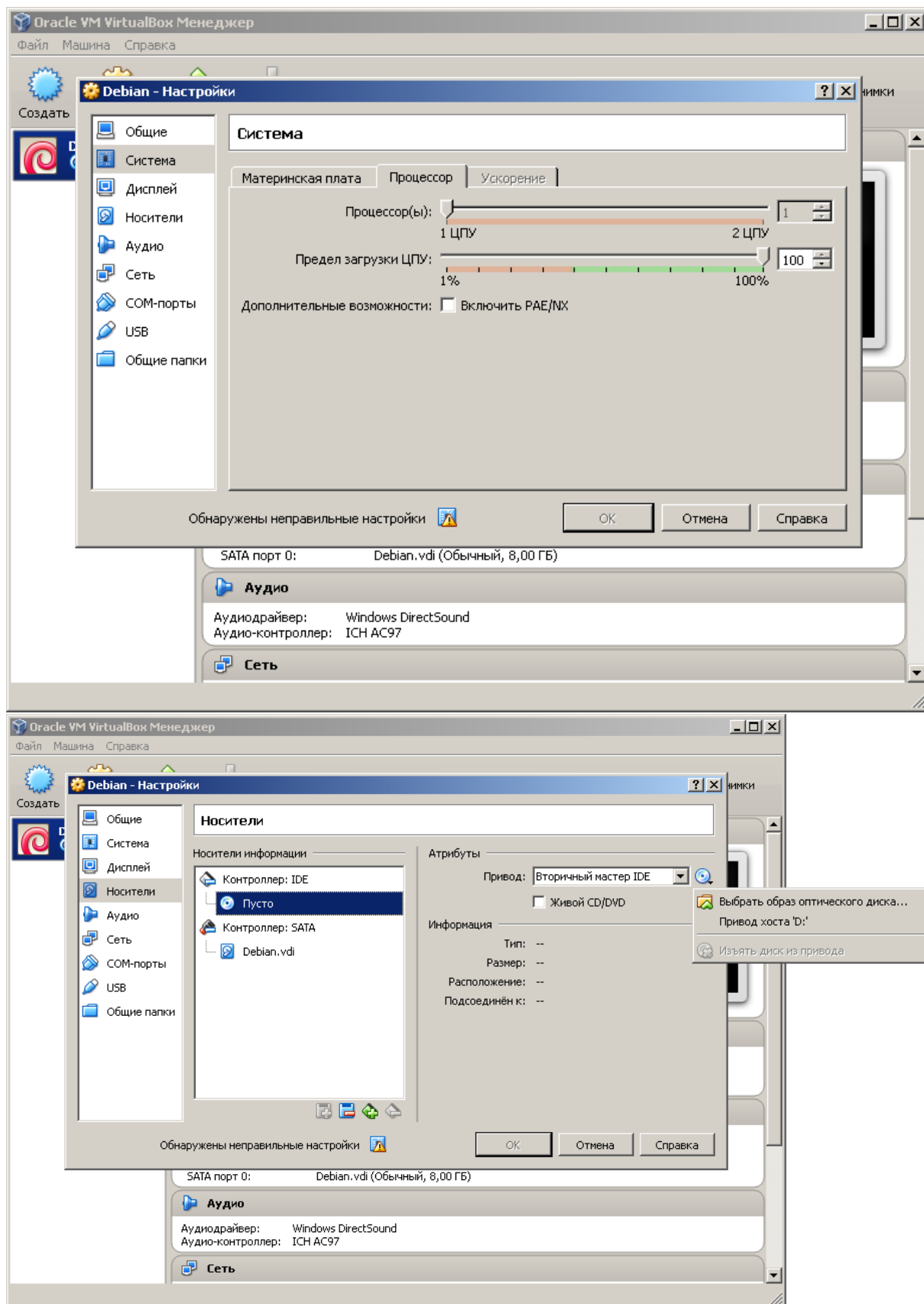


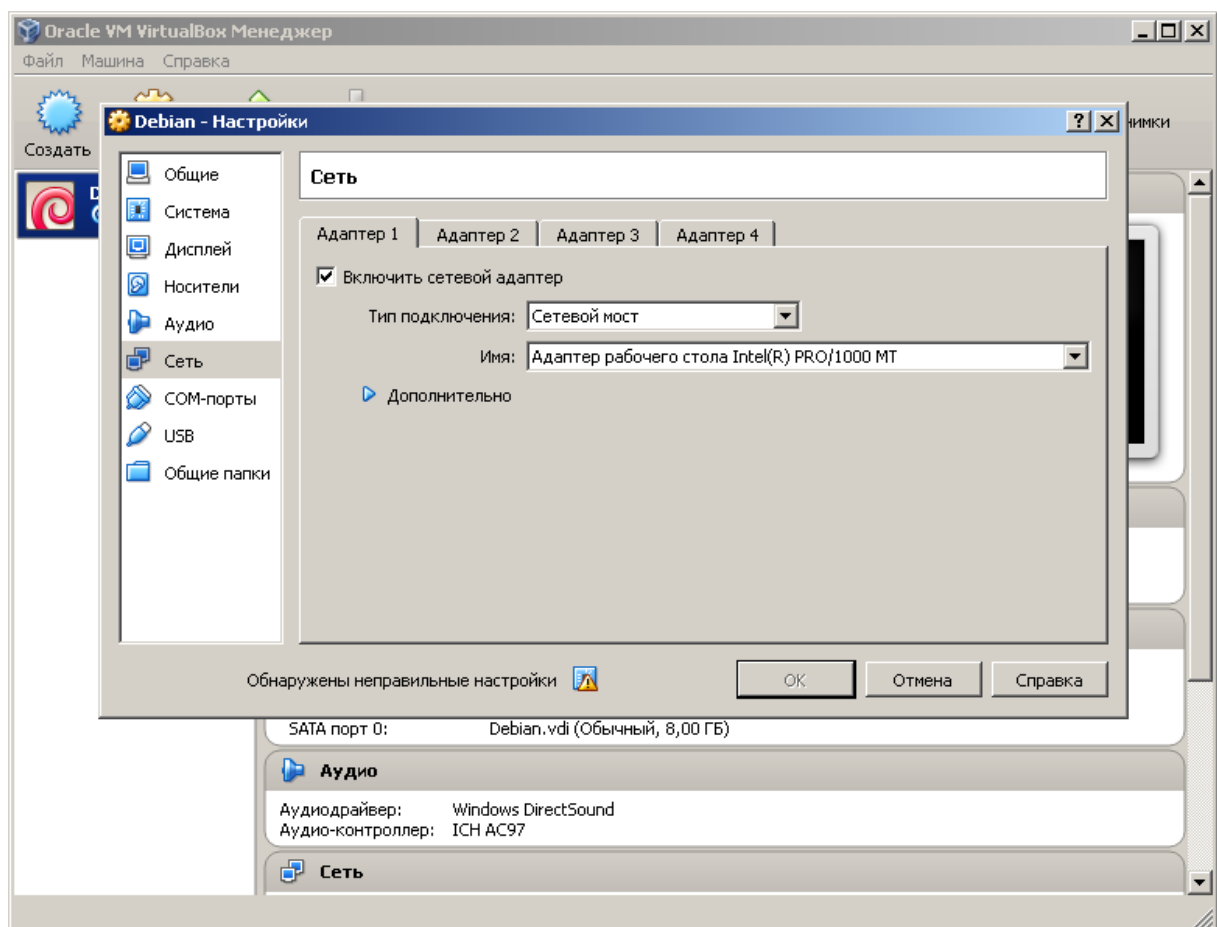




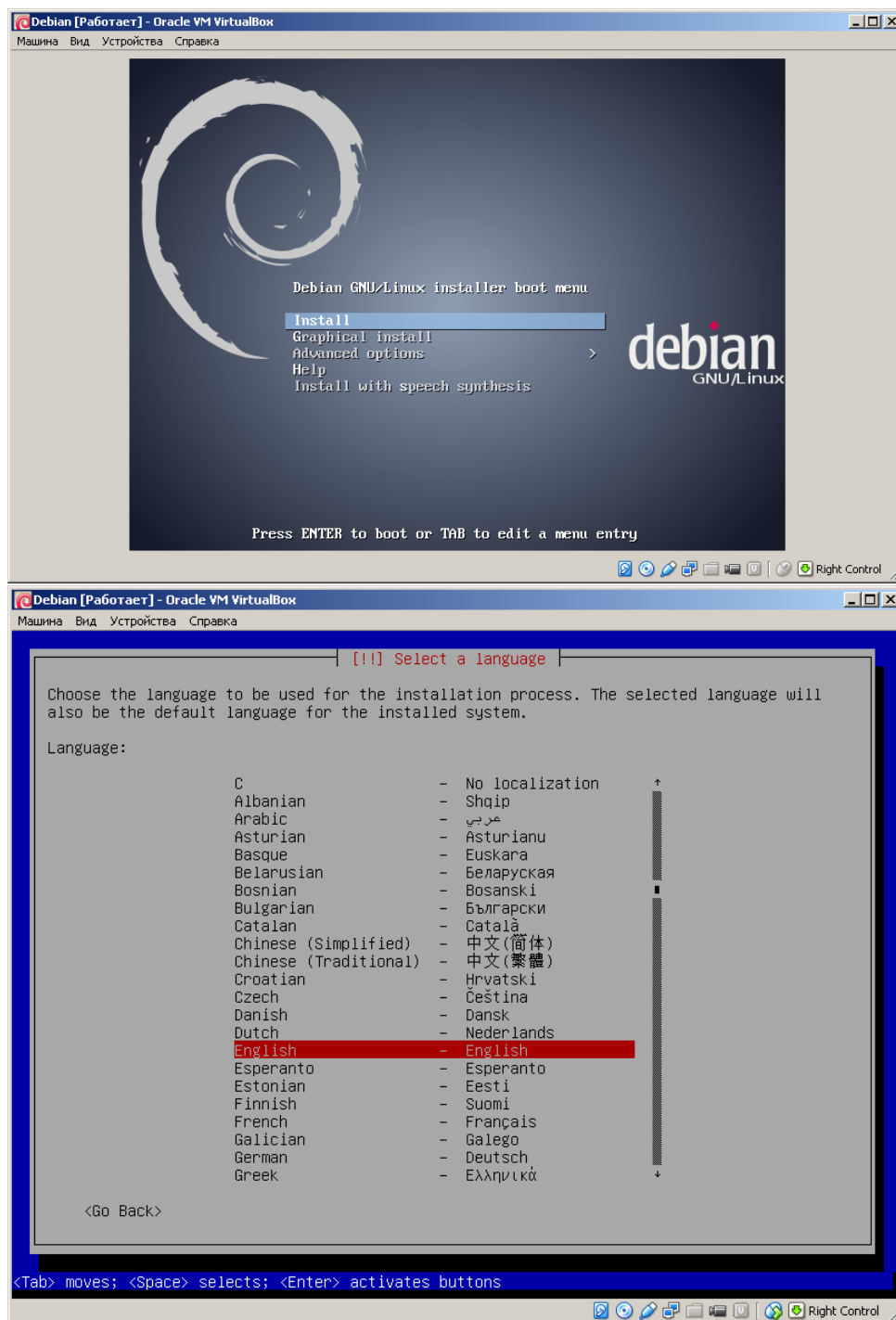


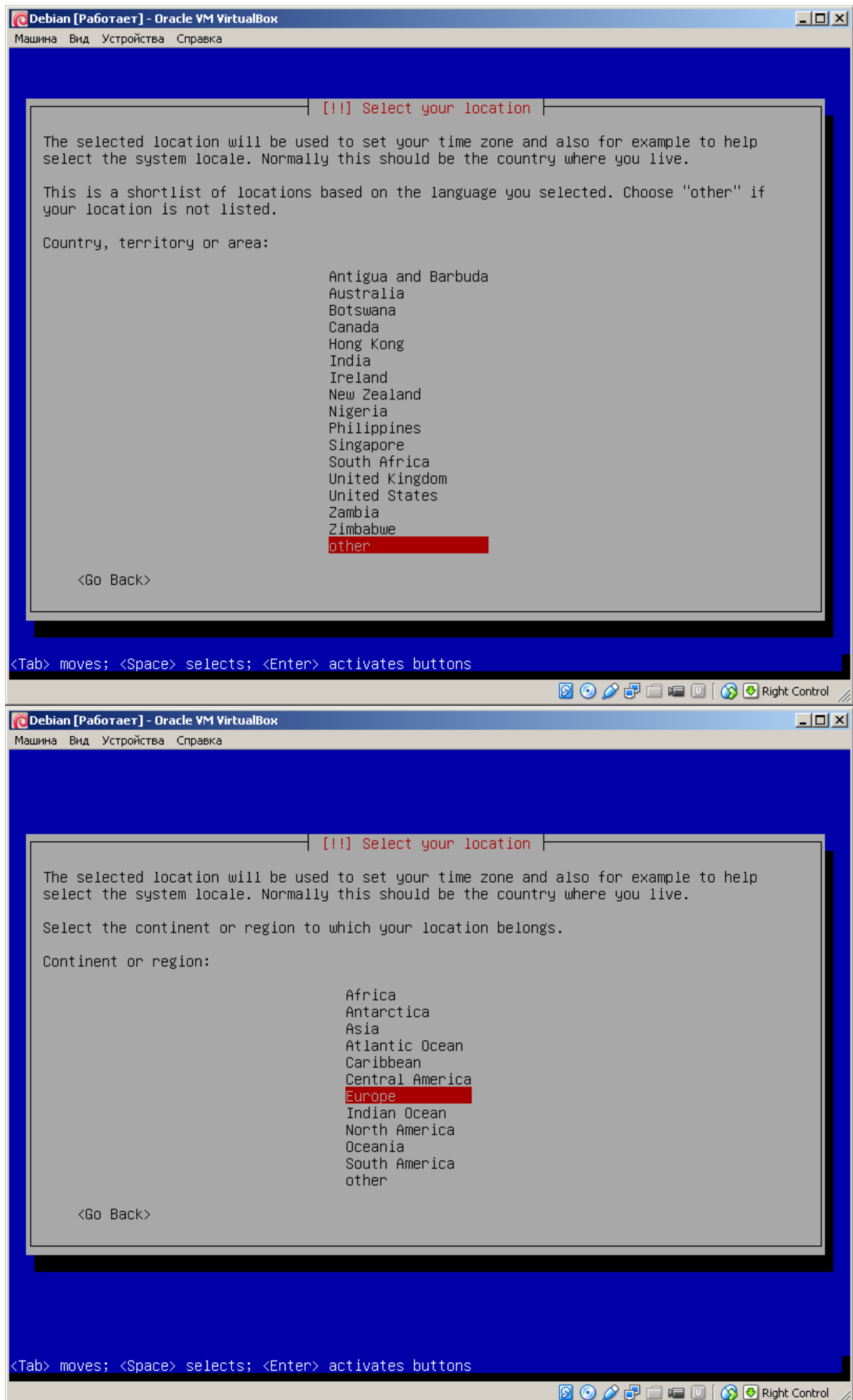


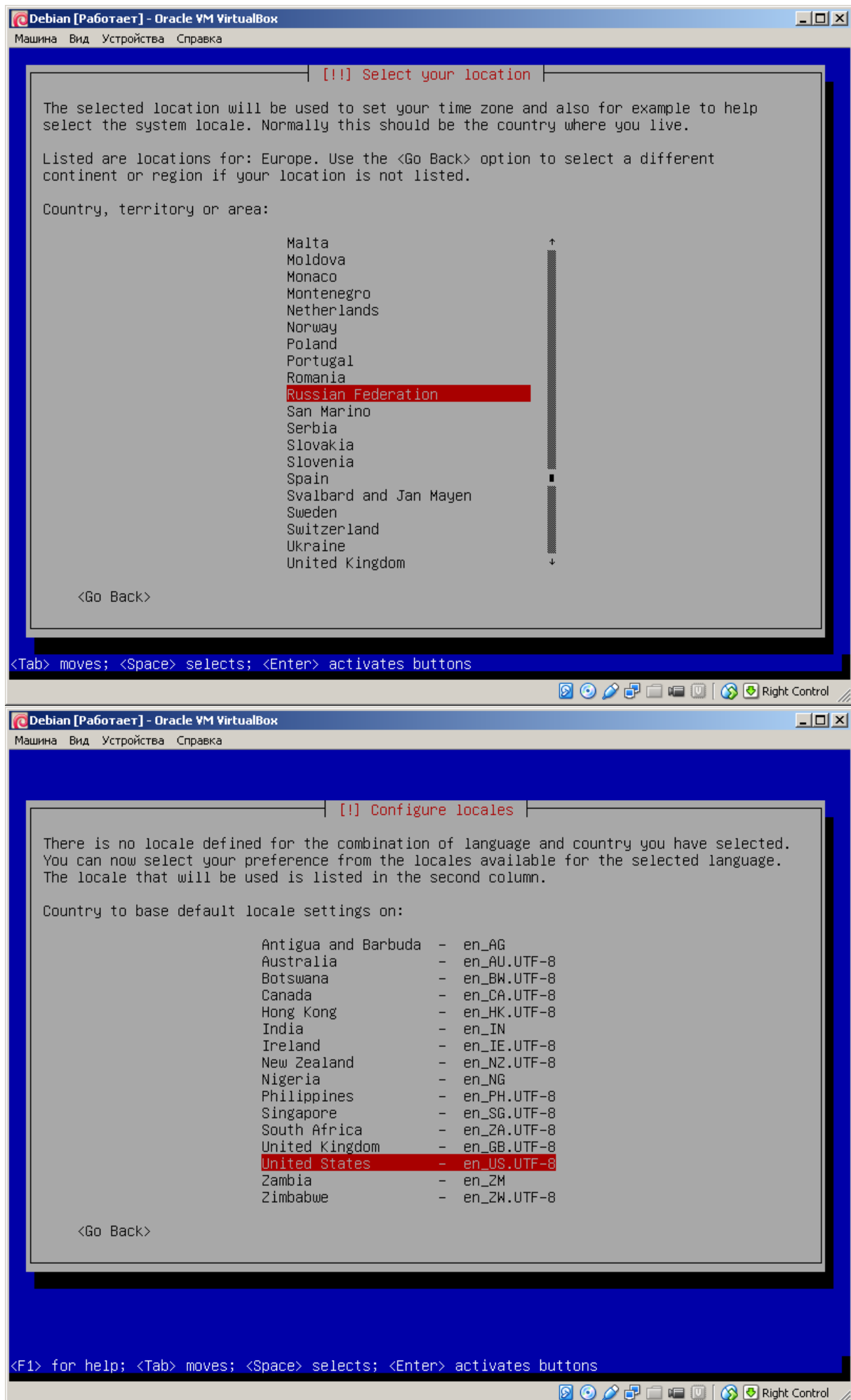


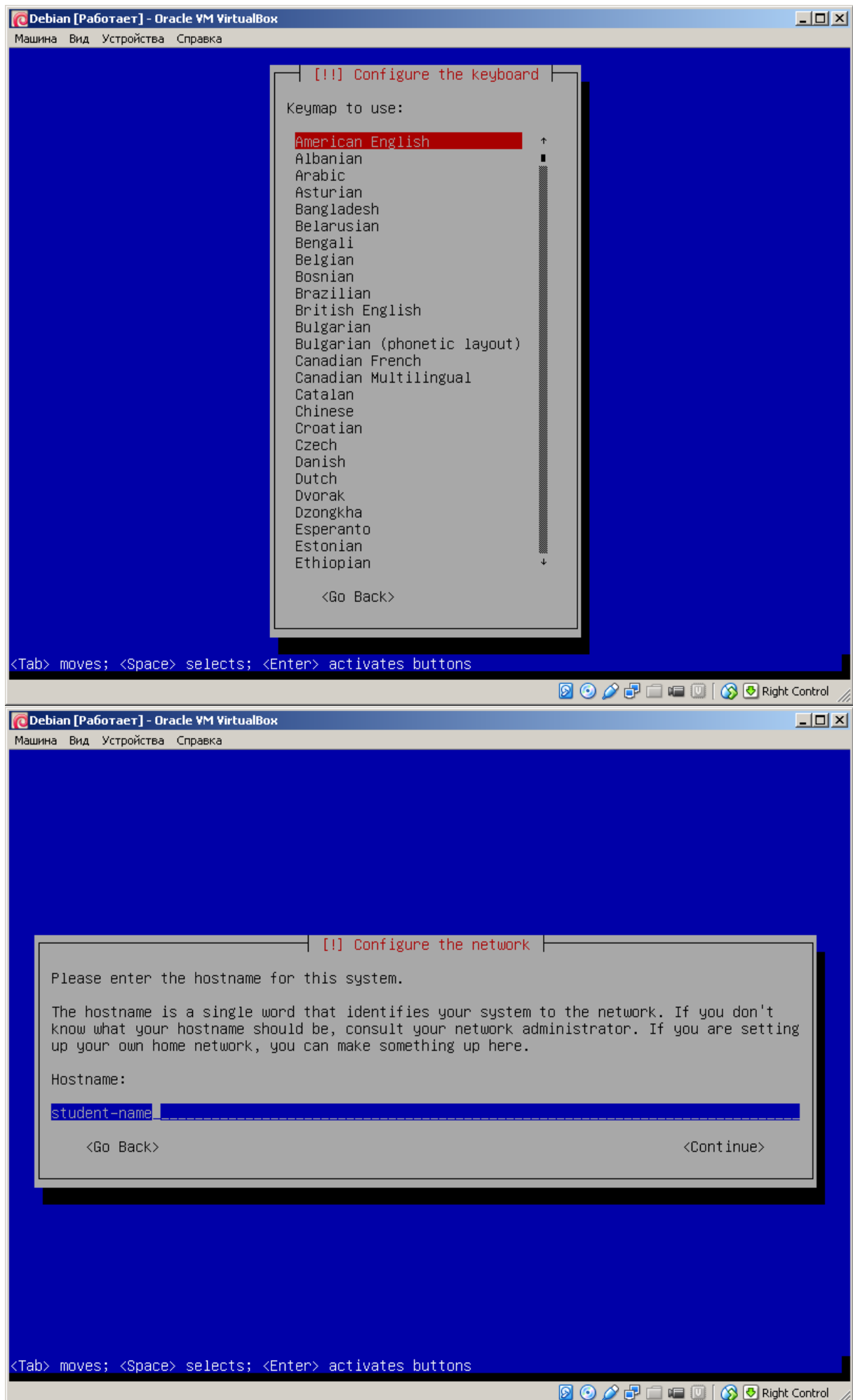


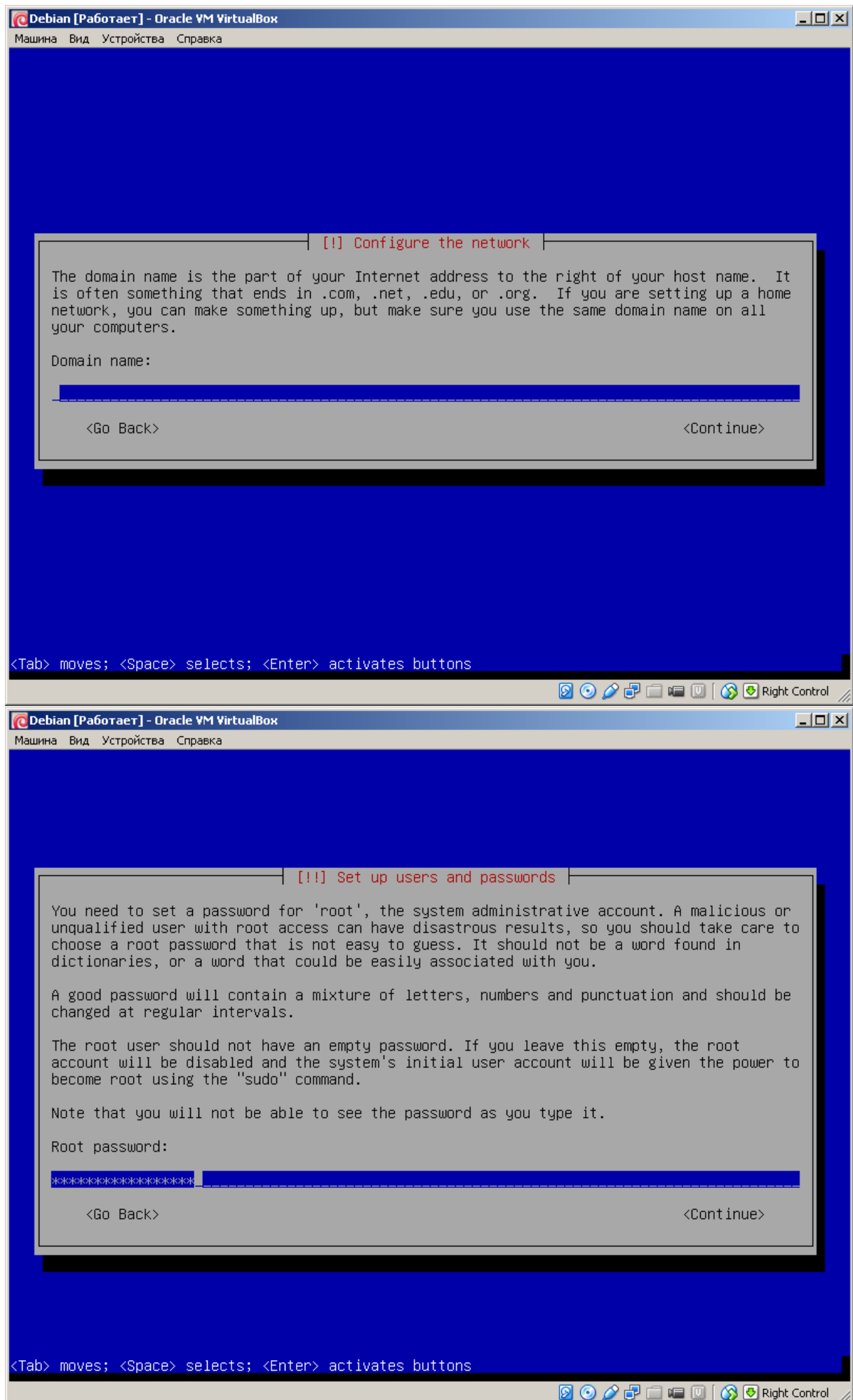
В Пример установки Debian GNU/Linux в VirtualBox

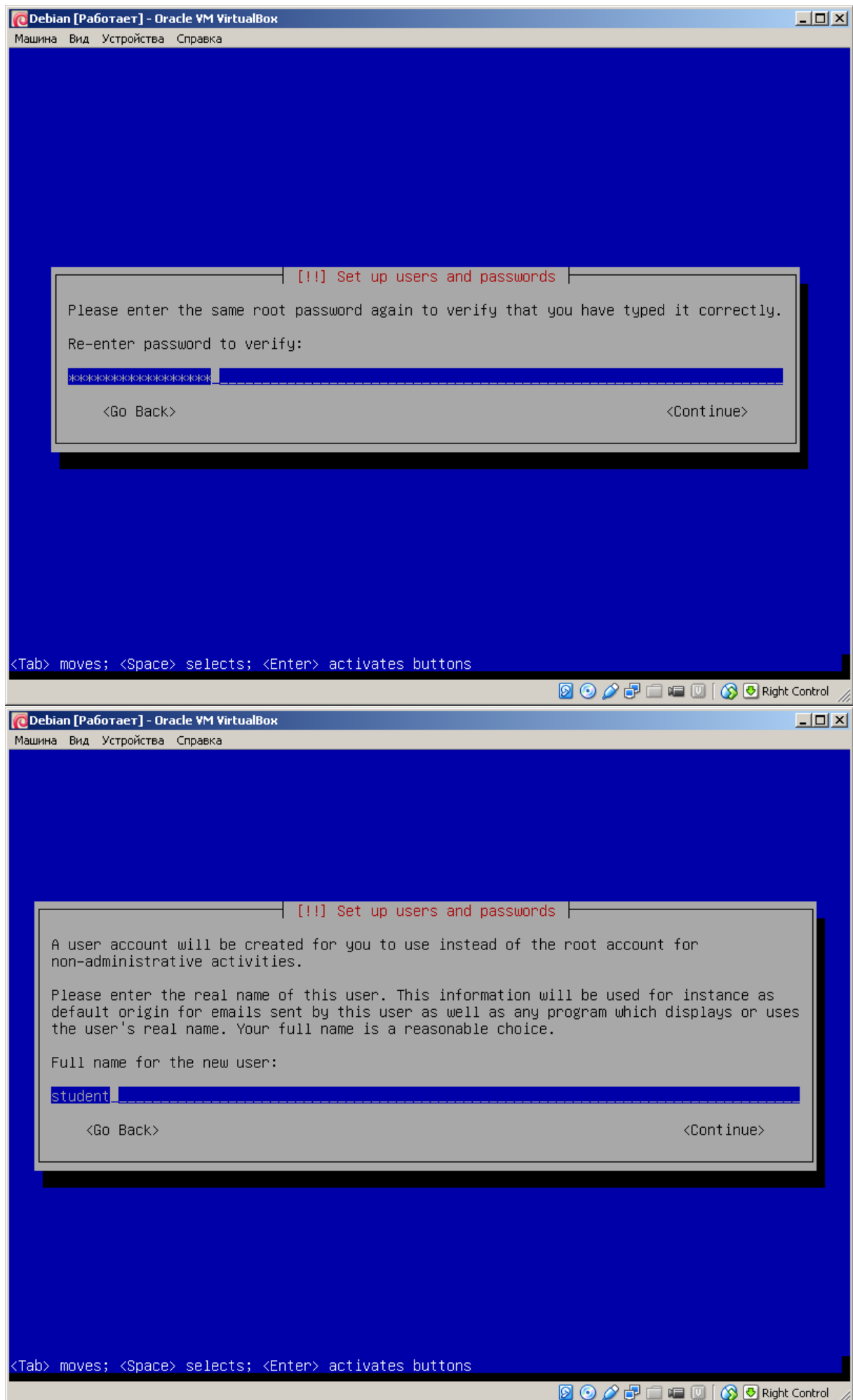


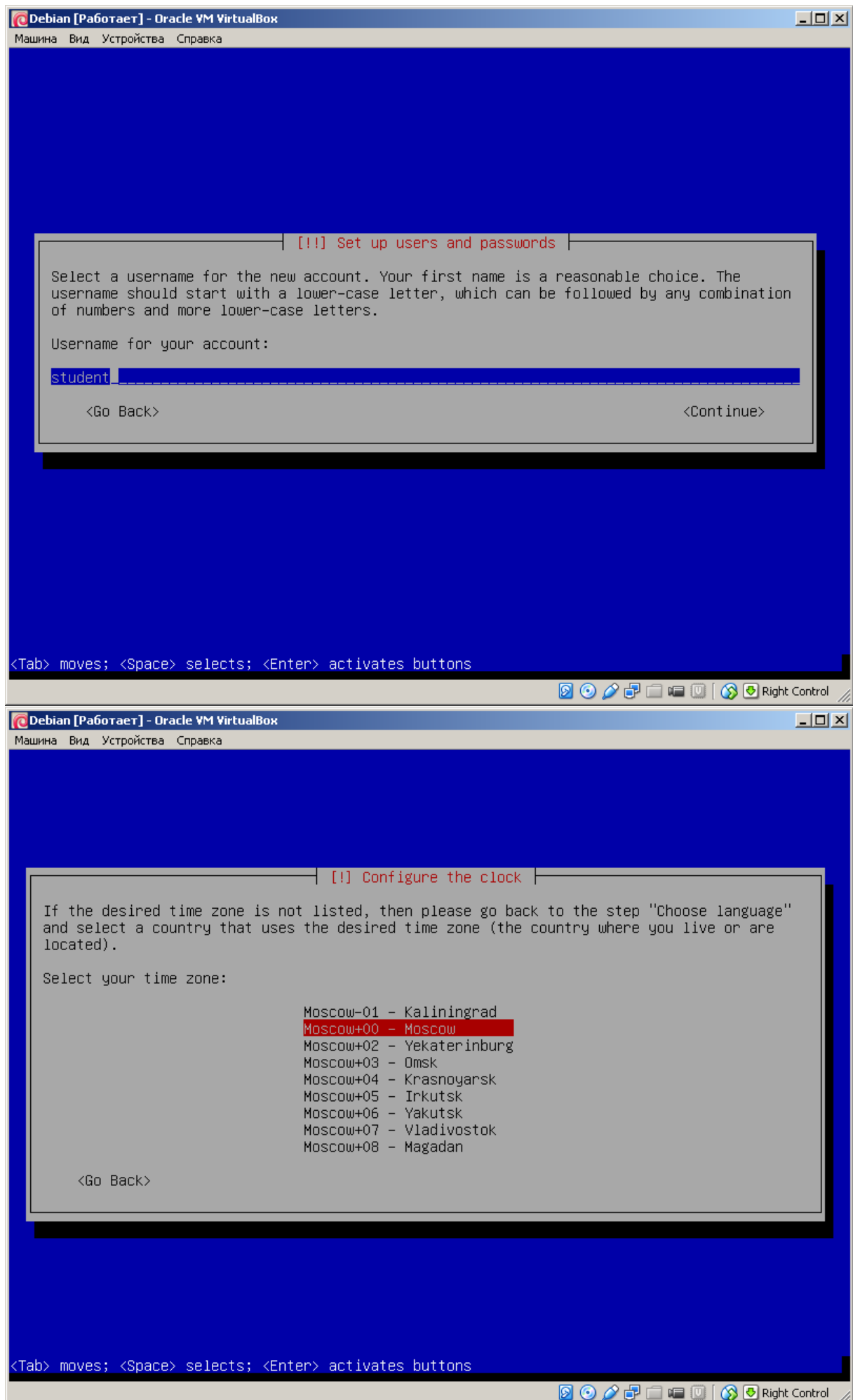


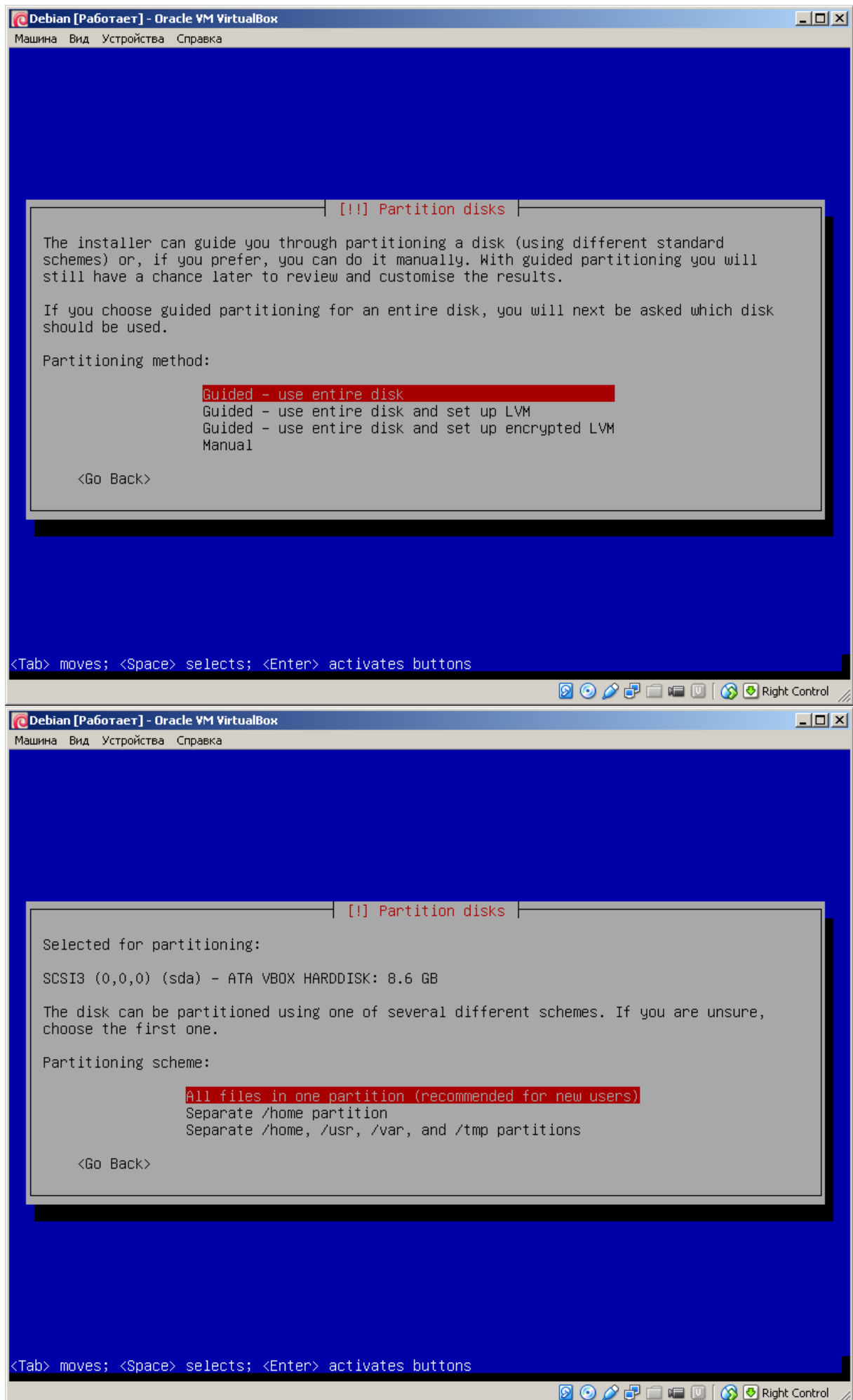


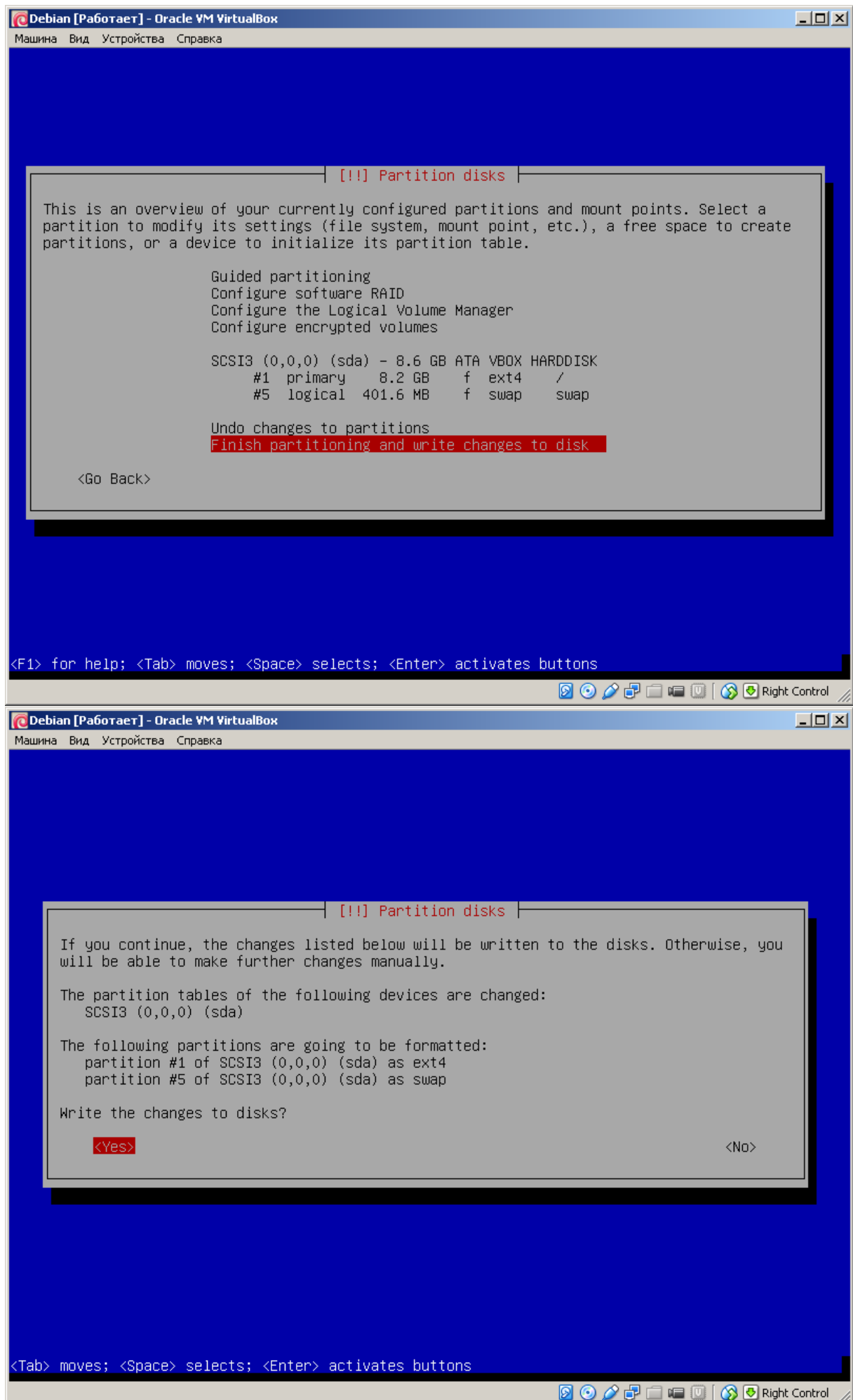


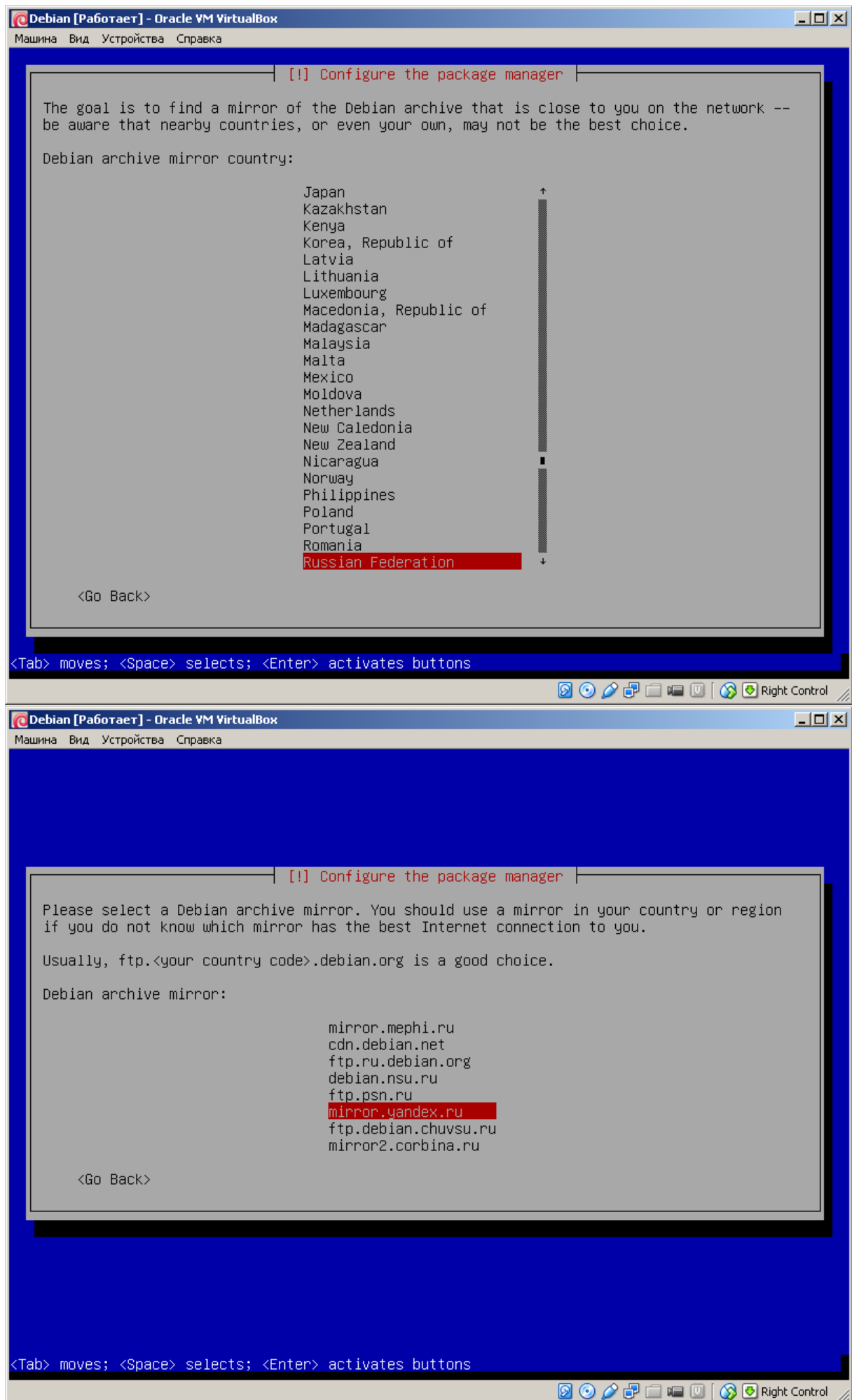


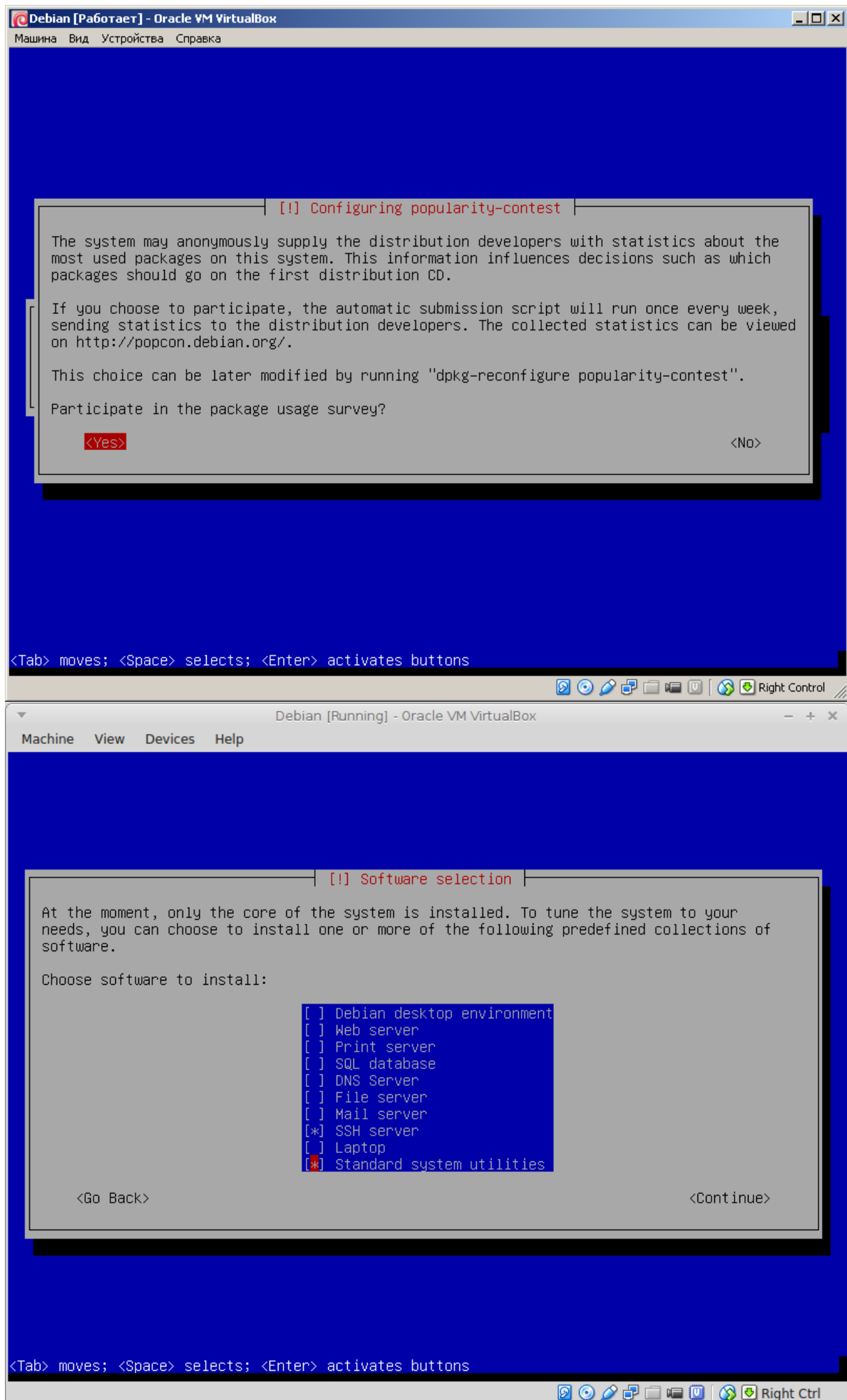














С Пример установки ownCloud Server в Debian GNU/Linux

После успешного входа в систему, в первую очередь необходимо получить права суперпользователя¹:

```
$ su -  
Password: toor  
# apt-get install sudo  
# usermod -a -G sudo student
```

Добавление репозитория и ключа для ownCloud Server:

```
# wget -nv https://goo.gl/mmV2ga -O Release.key  
# apt-key add - < Release.key  
OK  
# echo deb http://download.owncloud.org\  
> /download/repositories/stable/Debian_8.0/ / > \  
> /etc/apt/sources.list.d/owncloud.list
```

После добавления репозитория необходимо обновить список доступного в репозиториях ПО и запустить установку ownCloud Server (во время установки MySQL, установщик запросит пароль суперпользователя MySQL, он не обязательно должен совпадать с паролем пользователя root):

```
# apt-get update  
# apt-get install owncloud owncloud-files  
...  
New password for the MySQL "root" user: toor-mysql  
Repeat password for the MySQL "root" user: toor-mysql
```

После того, как установщик скачал и установил все необходимые пакеты, можно проверить корректность установки (рис. 7), зайдя по адресу <http://192.168.0.102/owncloud/>, где 192.168.0.102 — IP-адрес сервера (виртуальной машины).

Приложение предлагает использовать базу данных SQLite по умолчанию, мы же будем использовать MySQL:

```
# mysql -uroot -ptoor-mysql  
mysql> CREATE DATABASE owncloud_DB;  
mysql> CREATE USER "owncloud-web"@"localhost" \  
-> IDENTIFIED BY "owncloud-passwd";  
mysql> GRANT ALL PRIVILEGES ON owncloud_DB.* \  
-> TO "owncloud-web"@"localhost";  
mysql> FLUSH PRIVILEGES;  
mysql> quit
```

После создания базы данных, необходимо обновить страницу приложения, настроить параметры базы данных и установить данные аккаунта администратора ownCloud (рис. 8).

После нажатия клавиши «Finish Setup» база данных и пользовательские настройки успешно подключаются к ownCloud (рис. 9).

¹ Пароль пользователя не отображается на экране во время набора

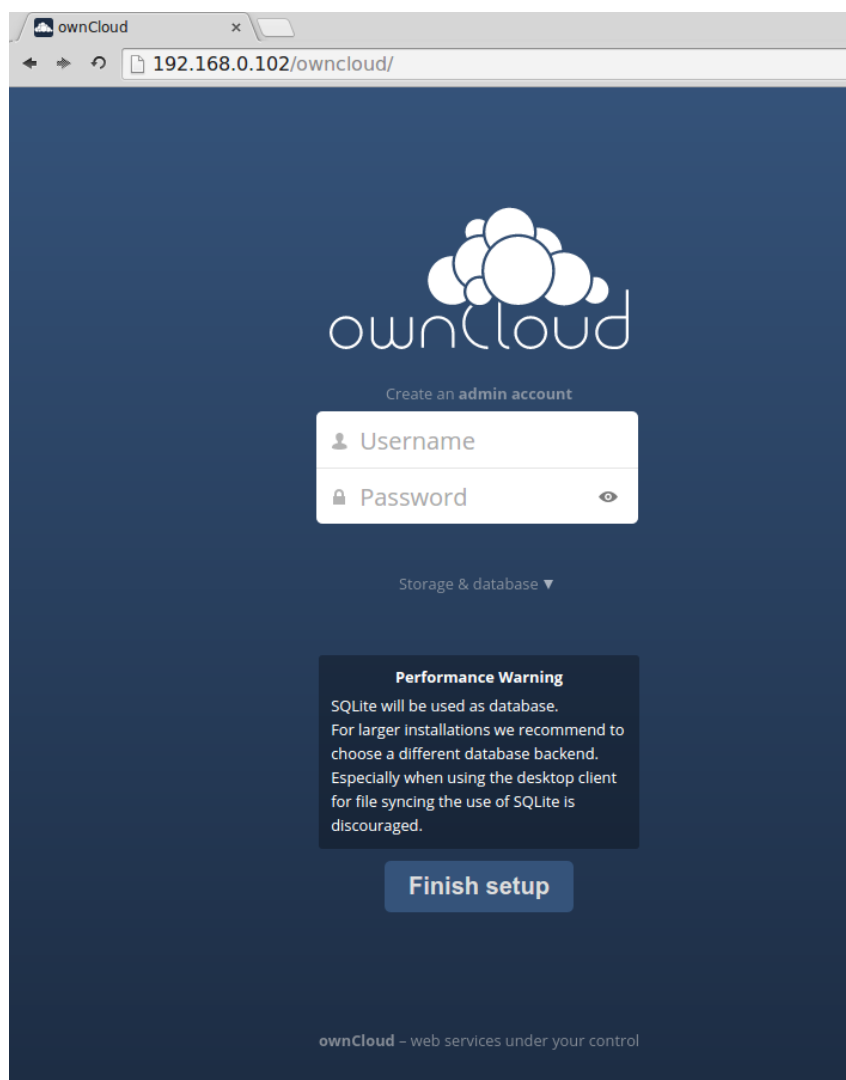



Рис. 7: Первый запуск приложения



Create an admin account

student

student-kvt-

Storage & database ▼

Data folder

/var/www/owncloud/data

Configure the database

SQLite MySQL/MariaDB PostgreSQL

owncloud-web

owncloud-passwd

owncloud-DB

localhost

Finish setup

ownCloud – web services under your control

Рис. 8: Параметры БД и аккаунта администратора

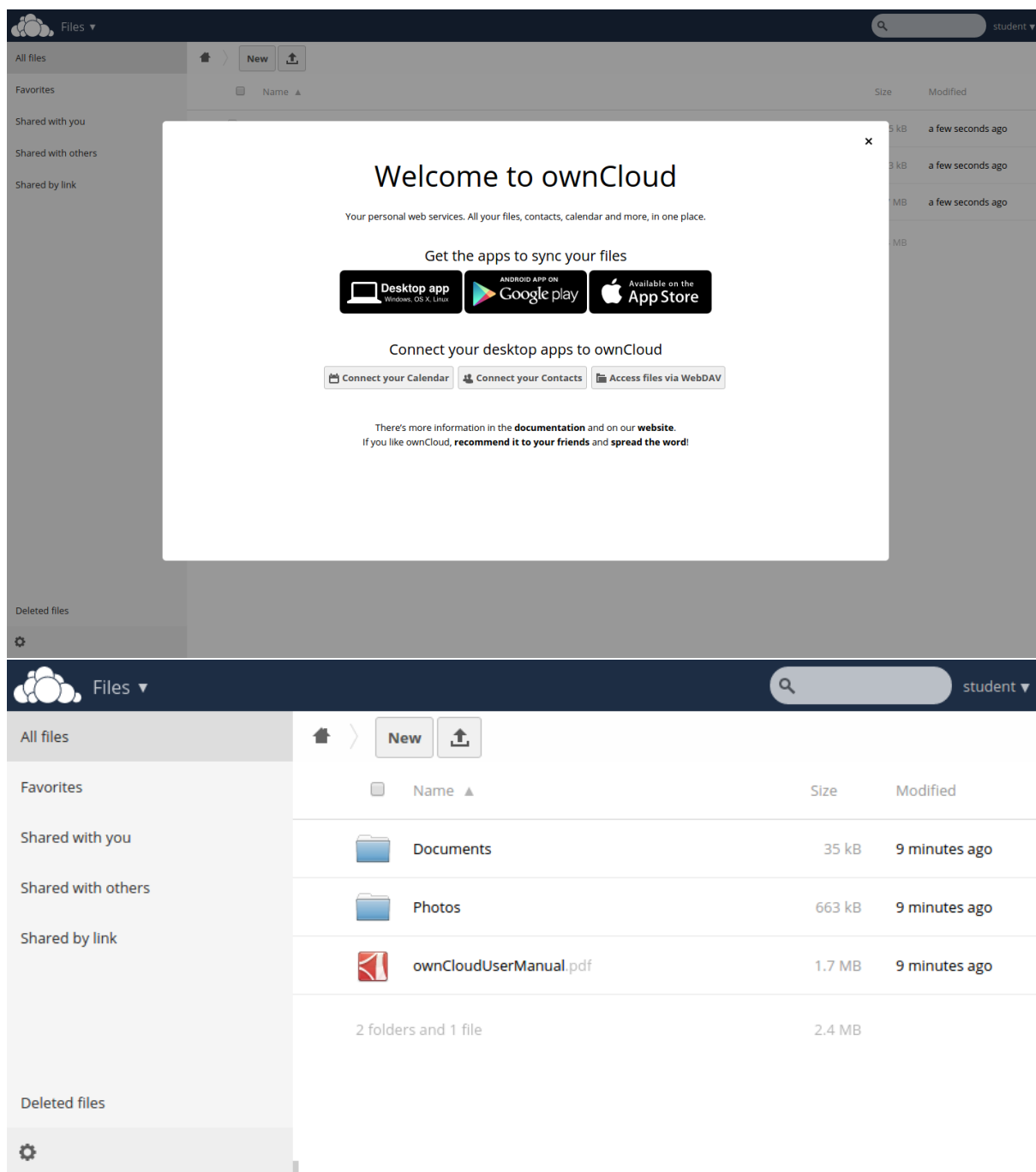
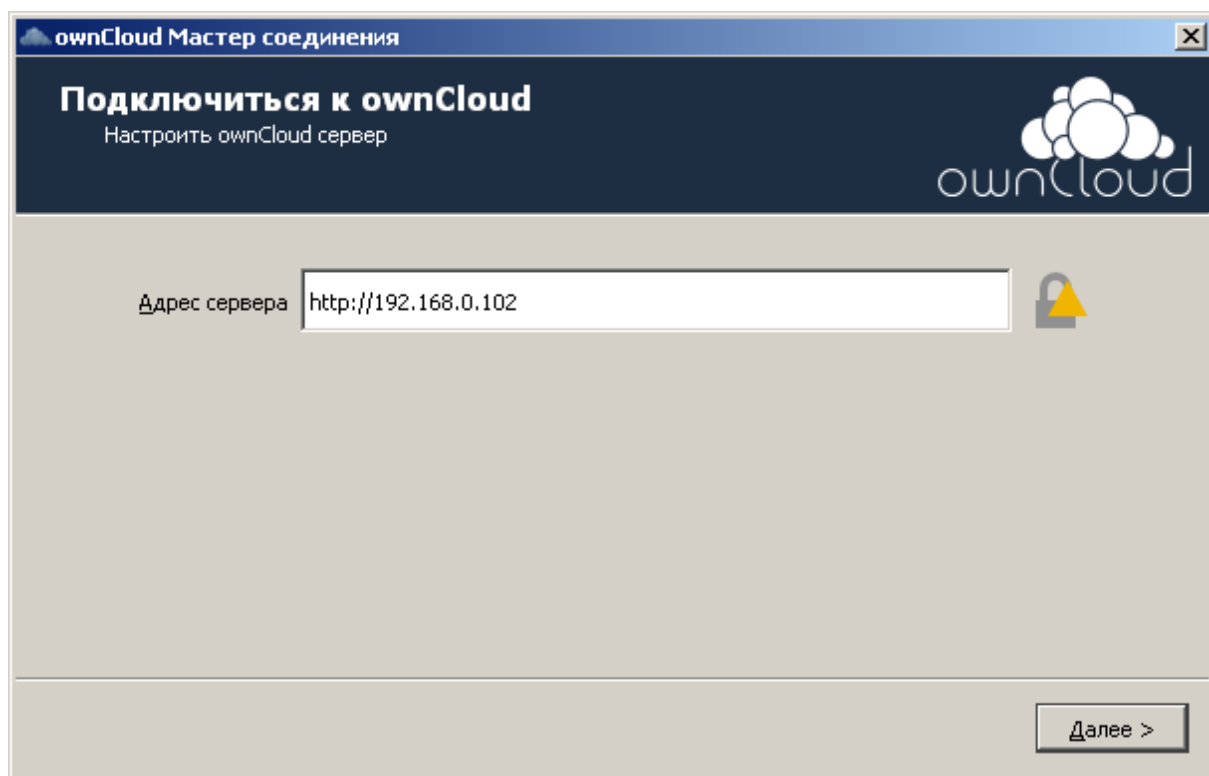


Рис. 9: Первый вход в ownCloud и интерфейс приложения

D Настройка подключения ownCloud Client к серверу



ownCloud Мастер соединения

Подключиться к ownCloud

Ввести учётные данные

Имя пользователя


Пароль

< Назад Далее >


ownCloud Мастер соединения

Подключиться к ownCloud

Настроить локальные папки

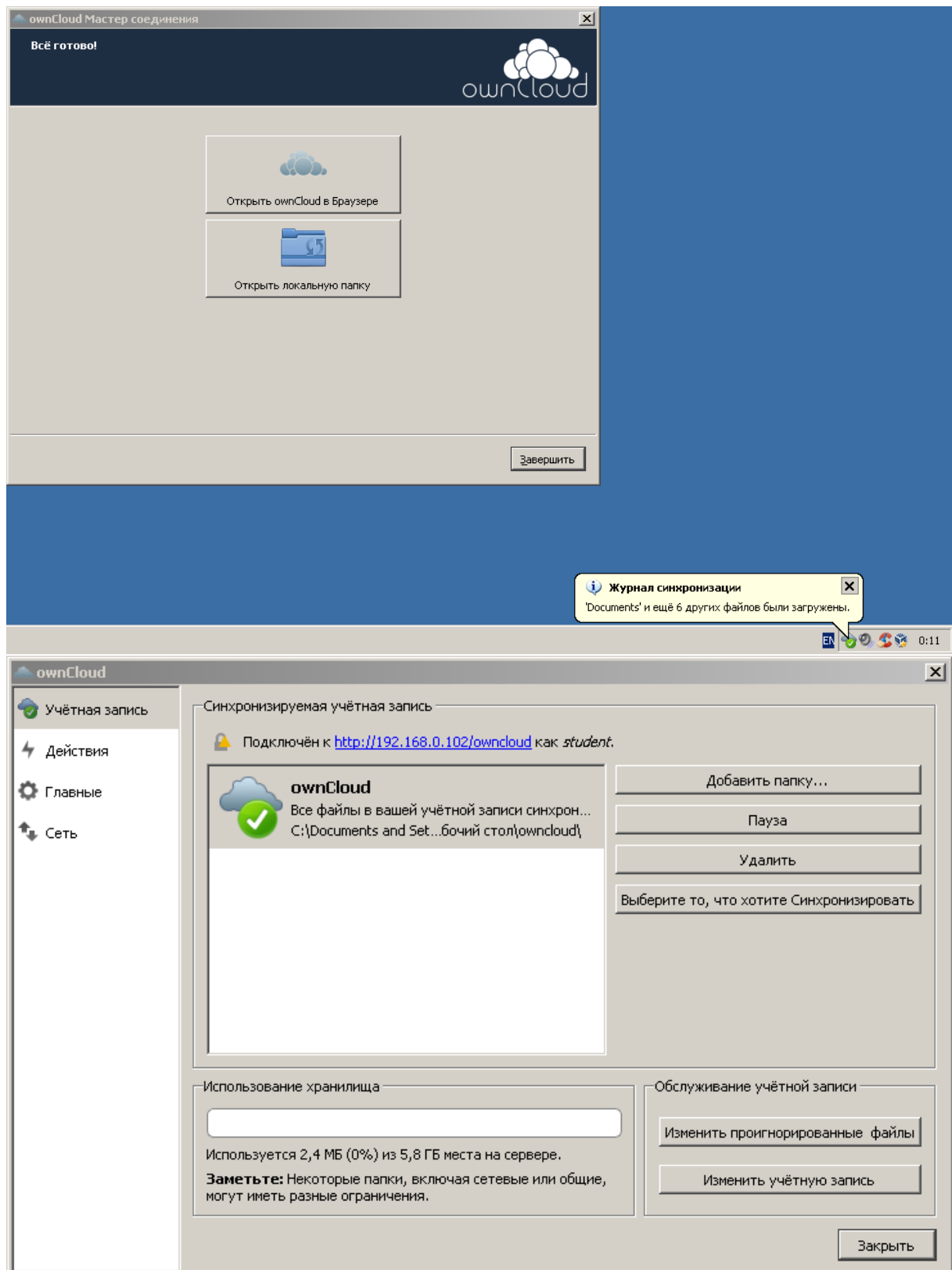
 ☒ Синхронизировать всё с сервера
☐ Выберите то, что хотите синхронизировать

Сервер



Локальная папка

Пропустить настройку папок < Назад Соединение...



Е Пример деплоя приложения на Heroku

Для регистрации в Heroku¹ необходима только электронная почта (рис. 10).

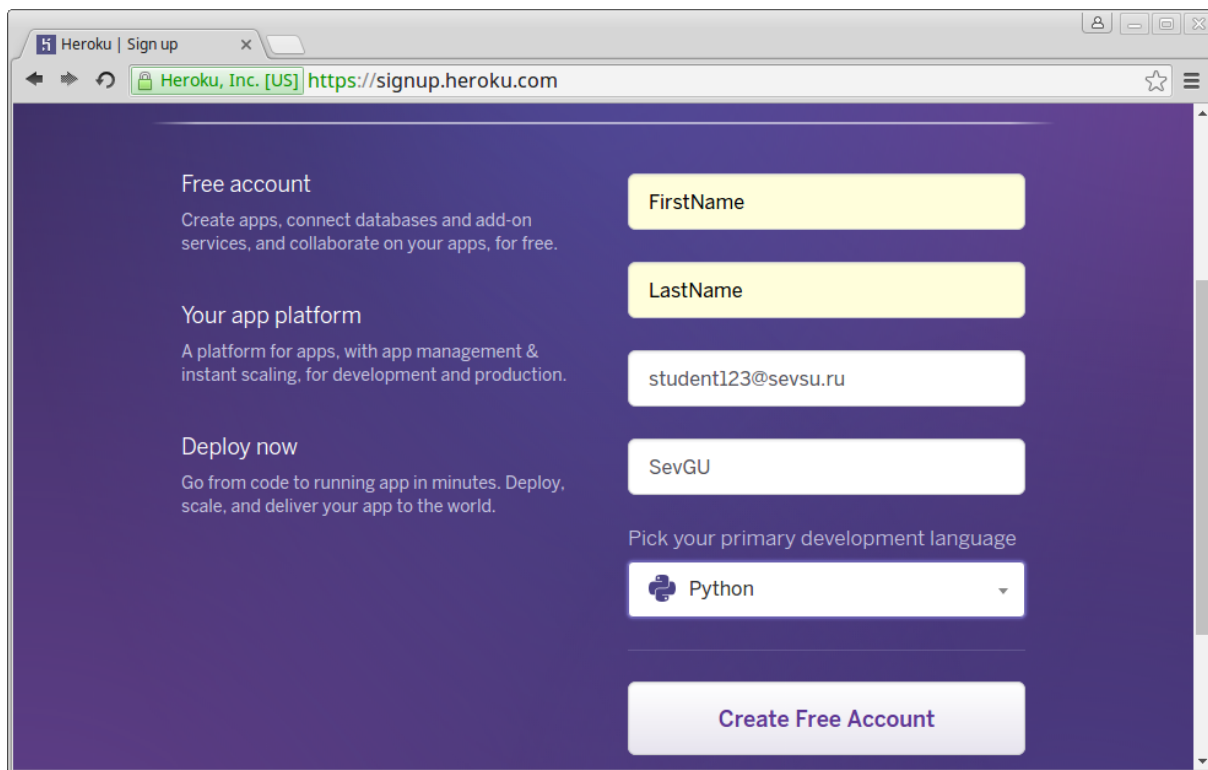


Рис. 10: Регистрация в Heroku

После регистрации, на указанный почтовый ящик приходит письмо с подтверждением регистрации, необходимо перейти по ссылке из письма и подтвердить регистрацию.

После подтверждения регистрации можно войти в свою учетную запись и ознакомиться с интерфейсом платформы.

Создадим новое приложение в разделе «Personal Apps» (рис. 11).

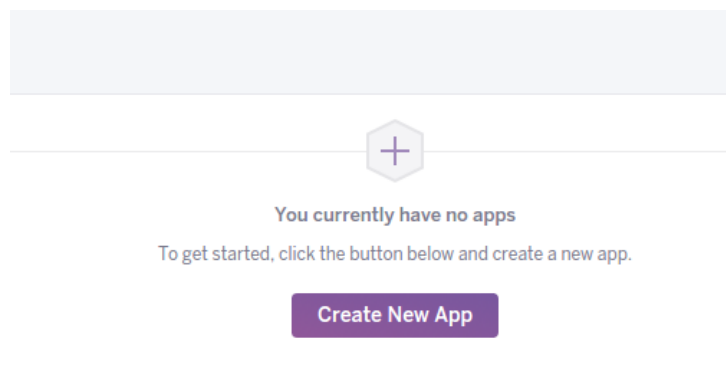
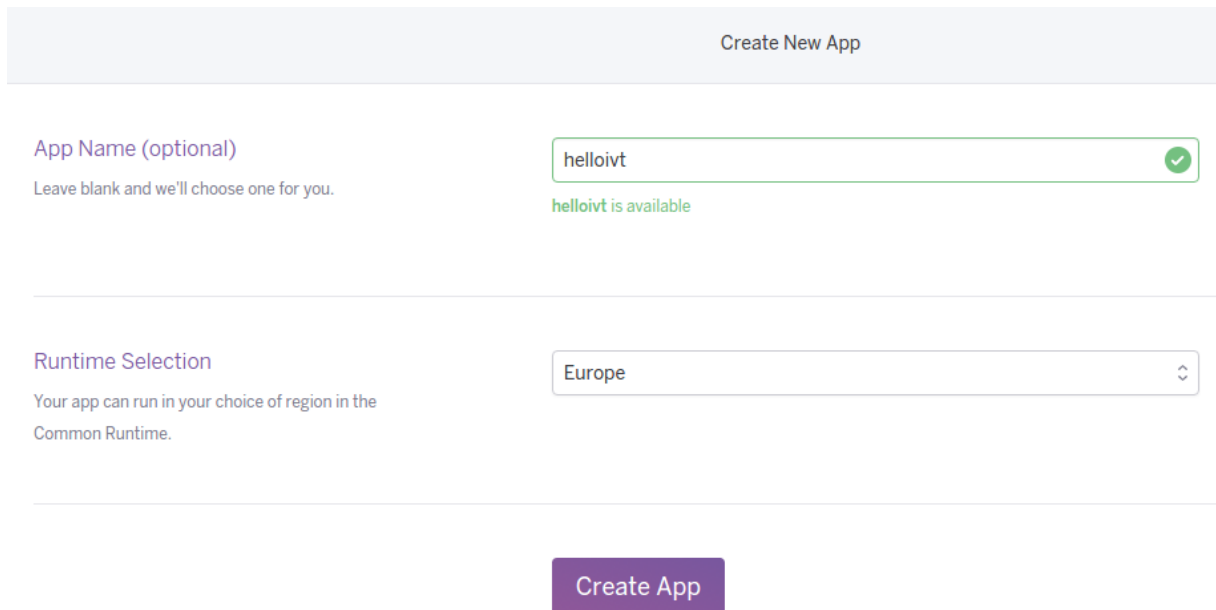


Рис. 11: Список созданных приложений в Heroku

Пусть имя приложения будет «helloivt» (рис. 14).

¹<https://signup.heroku.com/login>



Create New App

App Name (optional)
Leave blank and we'll choose one for you.

helloivt ✓
helloivt is available

Runtime Selection
Your app can run in your choice of region in the Common Runtime.

Europe

Create App

Рис. 12: Создание нового приложения

Авторизуемся на сервере с Debian, устанавливаем `heroku-cli` и авторизуемся в Heroku:

```
$ wget -O- https://toolbelt.heroku.com/install-ubuntu.sh | sh
$ heroku login
heroku-cli: Installing CLI... 21.83MB/21.83MB
Enter your Heroku credentials.
Email: student123@sevsu.ru
Password (typing will be hidden):
Logged in as student123@sevsu.ru
```

Добавляем контактные данные разработчика:

```
$ git config --global --add user.email "student123@sevsu.ru"
$ git config --global --add user.name "Name Surname"
```

Клонируем репозиторий с тестовым приложением на Flask:

```
$ git clone https://github.com/craigkerstiens/flask-helloworld
$ cd flask-helloworld/
```

Реинициализируем репозиторий:

```
$ git init
Reinitialized existing Git repository in /home/student/flask-
helloworld/.git/
$ heroku git:remote -a helloivt
set git remote heroku to https://git.heroku.com/helloivt.git
```

Деплоим приложение в Heroku:

```
$ git push heroku master
```

Переходим на страницу приложения¹ и проверяем, страница должна выводить надпись «Hello from Python!» (рис. 13).

¹<https://helloivt.herokuapp.com/>, где «helloivt» — это имя приложения, заданное в веб-интерфейсе Heroku

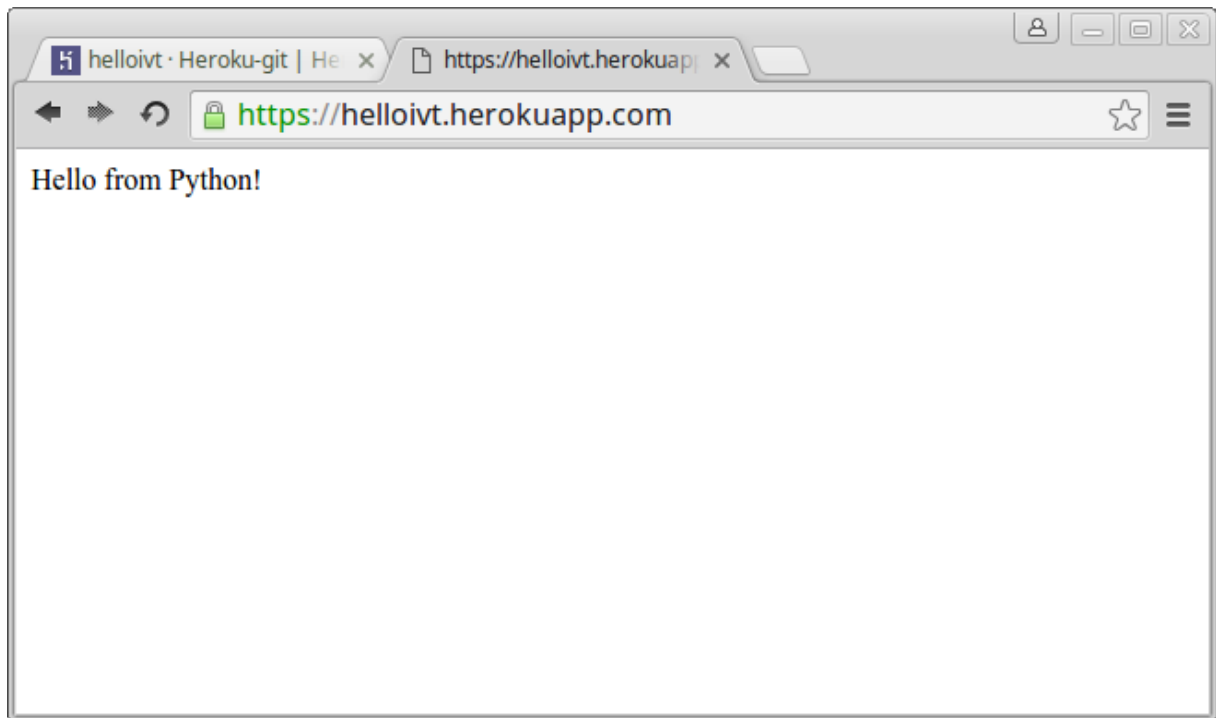


Рис. 13: Создание нового приложения

Внесем изменения в приложение, отредактировав приветствие:

```
$ nano app.py
    return "Hello from IVT student!"
$ git commit -am "First commit!"
[master 898f7d8] First commit!
1 file changed, 1 insertion(+), 1 deletion(-)
$ git push heroku master
```

Обновляем страницу с приложением (рис. 14).

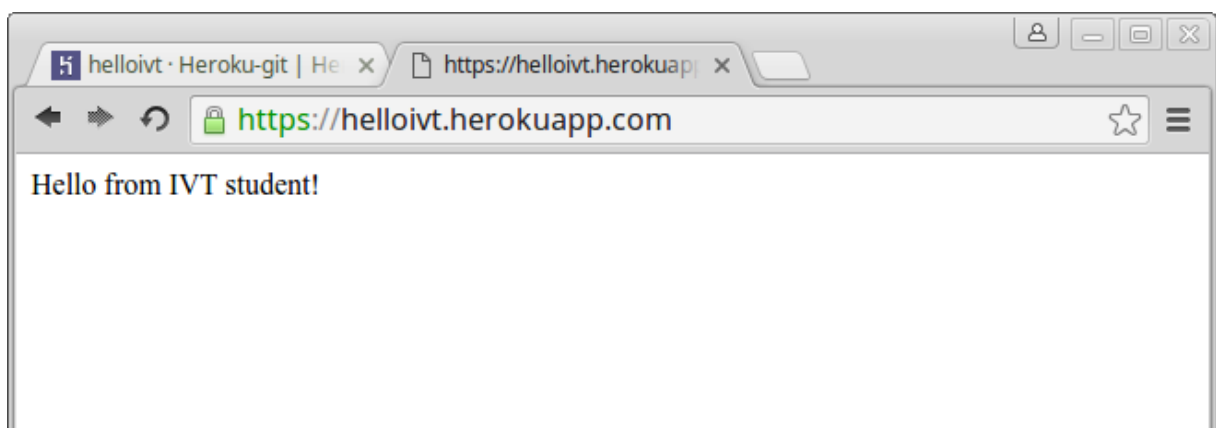


Рис. 14: Изменения в приложении вступили в силу