**3 РАЗРАБОТКА ИНФОРМАЦИОННОГО ОБЕСПЕЧЕНИЯ**

Для проведения лабораторного практикума были выбраны Subversion, Mercurial и Git в качестве систем контроля версий и tcl/tk, zenity и HTML в качестве примеров для написания программ или разметки.

**3.1 Обмен данными в системах контроля версий**

В данном дипломном проекте при первоначальной разработке использовался аккаунт Microsoft Azure. В связи с тем, что качество интернет-соединения было достаточно невысокое (по сравнению с требованиями Azure), и оно на настоящий момент не может предоставить скорости, которой будет достаточно для качественного отображения удалённого рабочего стола, было принято решение тестировать и проводить отладку программного комплекса лабораторного практикума с использованием программы PuTTY, которая взаимодействует с удалённым сервером через SSH (без использования удалённого рабочего стола).

SSH (англ. Secure Shell — «безопасная оболочка») — сетевой протокол прикладного уровня, позволяющий производить удалённое управление операционной системой и туннелирование TCP-соединений (например, для передачи файлов). Схож по функциональности с протоколами Telnet и rlogin, но, в отличие от них, шифрует весь трафик, включая и передаваемые пароли. SSH допускает выбор различных алгоритмов шифрования. SSH-клиенты и SSH-серверы доступны для большинства сетевых операционных систем.

SSH позволяет безопасно передавать в незащищённой среде практически любой другой сетевой протокол. Таким образом, можно не только удалённо работать на компьютере через командную оболочку, но и передавать по шифрованному каналу звуковой поток или видео. Также SSH может использовать сжатие передаваемых данных для последующего их шифрования.

Кроме этого, при разработке использовался SSH-туннель — туннель, создаваемый посредством SSH-соединения и используемый для шифрования туннелированных данных. Он используется для того, чтобы обезопасить передачу данных в Интернете (аналогичное назначение имеет IPsec). При пересылке через SSH-туннель незашифрованный трафик любого протокола шифруется на одном конце SSH-соединения и расшифровывается на другом.

Все эти меры предосторожности были нужны потому, что аккаунт в Azure является платным, а в случае перехвата злоумышленниками пары логин-пароль и последующего несанкционированного доступа к аккаунту разработчика, поддержка Azure-аккаунта закрывается (однако, злоумышленник может потратить достаточно большое количество средств, которые всё ещё будут на счету разработчика).

Кроме непосредственно разработки и взаимодействия с удалённой ОС необходимо было использовать сетевой протокол для общения с системами контроля версий. Все используемые в лабораторном практикуме системы контроля версий умеют работать с четырьмя сетевыми протоколами для передачи данных: локальный, Secure Shell (SSH), HTTP и собственный (соответственно, SVN, Git или Mercurial). Важно понимать, что за исключением протокола HTTP, все эти протоколы требуют, чтобы система контроля версий была установлена и работала на сервере.

Базовым является локальный протокол, при использовании которого удалённый репозиторий — другой каталог на диске. Наиболее часто он используется, если все члены команды имеют доступ к общей файловой системе, например, к NFS, или, что менее вероятно, когда все работают на одном компьютере. Последний вариант не столь хорош, поскольку все копии вашего репозитория находятся на одном компьютере, делая возможность потерять всё более вероятной.

Наиболее часто используемый транспортный протокол — это SSH. Причина этого в том, что доступ по SSH уже есть на многих серверах, а если его нет, то его очень легко настроить. Кроме того, SSH — единственный из сетевых протоколов, предоставляющий доступ и на чтение, и на запись. Два других сетевых протокола (HTTP и Git) в большинстве случаев дают доступ только на чтение, поэтому даже если они вам доступны, вам всё равно понадобится SSH для записи. К тому же SSH это протокол с аутентификацией, и благодаря его распространённости обычно его легко настроить и использовать.

Огромный плюс протоколов HTTP и HTTPS в простоте их настройки. По сути, всё, что необходимо сделать — поместить пустой репозиторий внутрь каталога с HTTP документами и установить перехватчик post-update. В таком случае любой, имеющий доступ к веб-серверу, на котором был размещён репозиторий, может его склонировать.

Одним из примеров собственных протоколов является Git-протокол. Так, вместе с Git'ом поставляется специальный демон, который слушает порт 9418 и предоставляет сервис, схожий с протоколом SSH, но абсолютно без аутентификации. Чтобы использовать Git-протокол для репозитория, необходимо создать файл git-daemon-export-ok, иначе демон не будет работать с этим репозиторием, но следует помнить, что в протоколе отсутствуют средства безопасности. Соответственно, любой репозиторий в Git'е может быть либо доступен для клонирования всем, либо не доступен никому. Как следствие, обычно вы не можете отправлять изменения по этому протоколу. Вы можете открыть доступ на запись, но из-за отсутствия авторизации в этом случае кто угодно, зная URL вашего проекта, сможет его изменить.

При составлении лабораторного практикума был использован протокол SSH, как наиболее популярный, доступный и действенный способ соединения.

**3.2 Форматы файлов при использовании систем контроля версий**

В данном цикле лабораторных работ используются такие системы контроля версий, в которых при сохранении новых версий на сервере используется дельта-компрессия: система находит отличия новой версии от предыдущей и записывает только их, избегая дублирования данных. Таким образом, на сервере будут лежать уже сжатые файлы; прочитать из них информацию без предварительной переработки не получится.

Теперь рассмотрим ситуацию на других клиентах.

В случае системы контроля версий Git в каталоге .git находится вся нужная информация про файлы и папки, а в корне самой директории – непосредственно текущая версия репозитория. Подробный пример можно увидеть на рисунке 3.1:

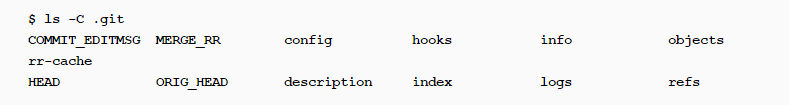


Рисунок 3.1 – Содержание .git репозитория на клиенте

В папке с базой данных объектов хранятся преимущественно каталоги, состоящие из двух символов – первых двух букв хэша sha1 объекта, хранящегося в Git. Внутри лежат сжатые и закодированные данные про объекты, хранящиеся в Git.

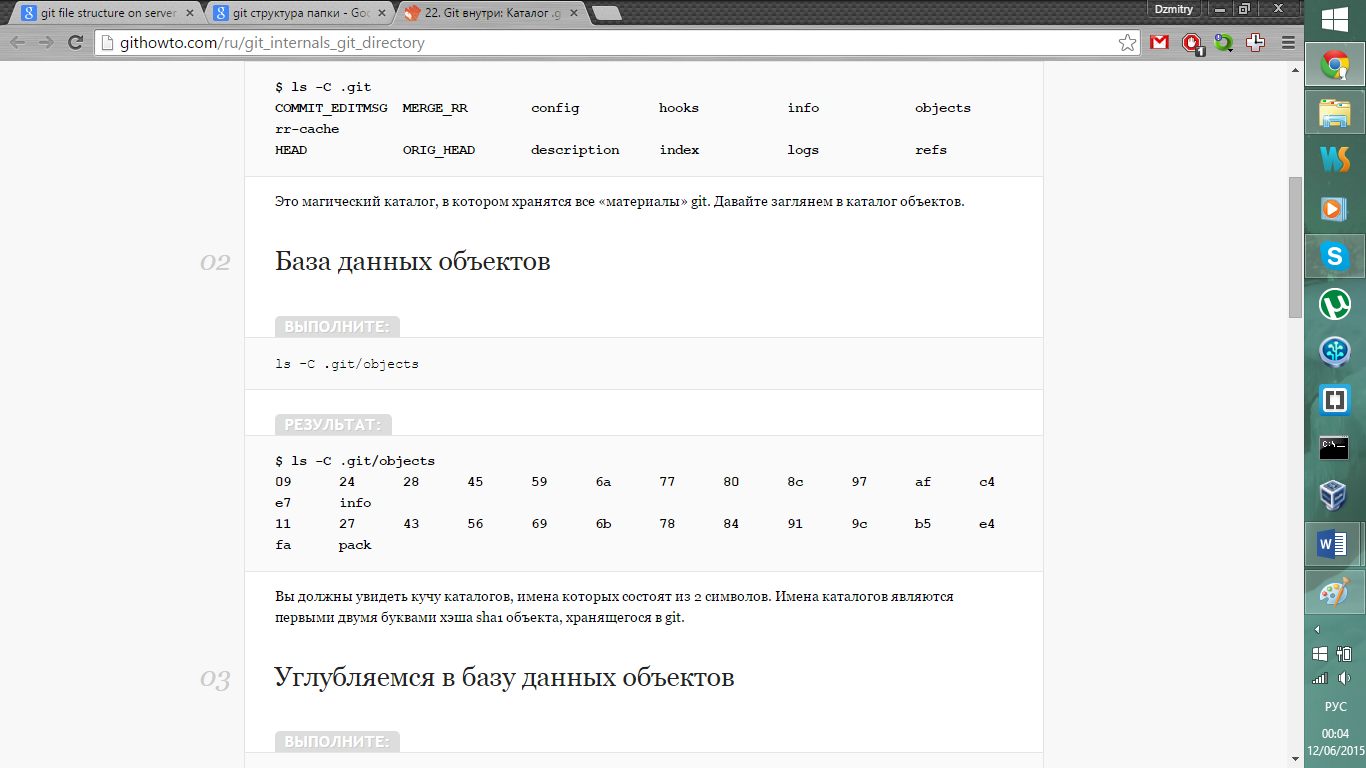


Рисунок 3.2 – Содержание папки object

При работе с Mercurial структура каталога будет

Когда Mercurial отслеживает изменения файла, он сохраняет историю этого файла в объекте метаданных называемом filelog. Каждая запись в filelog содержит достаточно информации чтобы восстановить одну ревизию отслеженного файла. Filelog'и хранятся в виде файлов в папке .hg/store/data. Они содержат два вида информации: данные о ревизиях и индексы, помогающие Mercurial эффективно искать ревизии.

Для большого или имеющего длинную историю файла filelog хранится раздельно в файлах с данными (расширение .d) и с индексом (расширение .i). Для маленьких файлов с небольшой историей ревизионные данные и индекс хранятся в едином файле с расширением .i. Связь между файлом в рабочей директории и filelog'ом, который отслеживает его историю в хранилище, показана на рисунке 3.3:

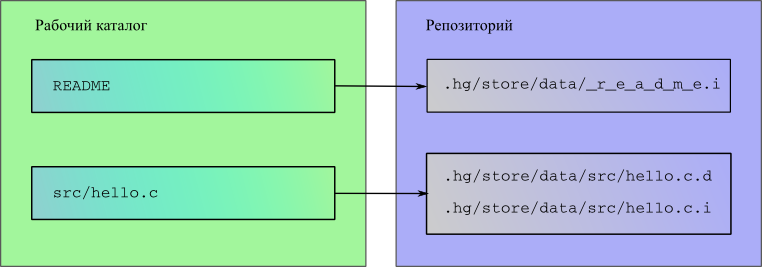


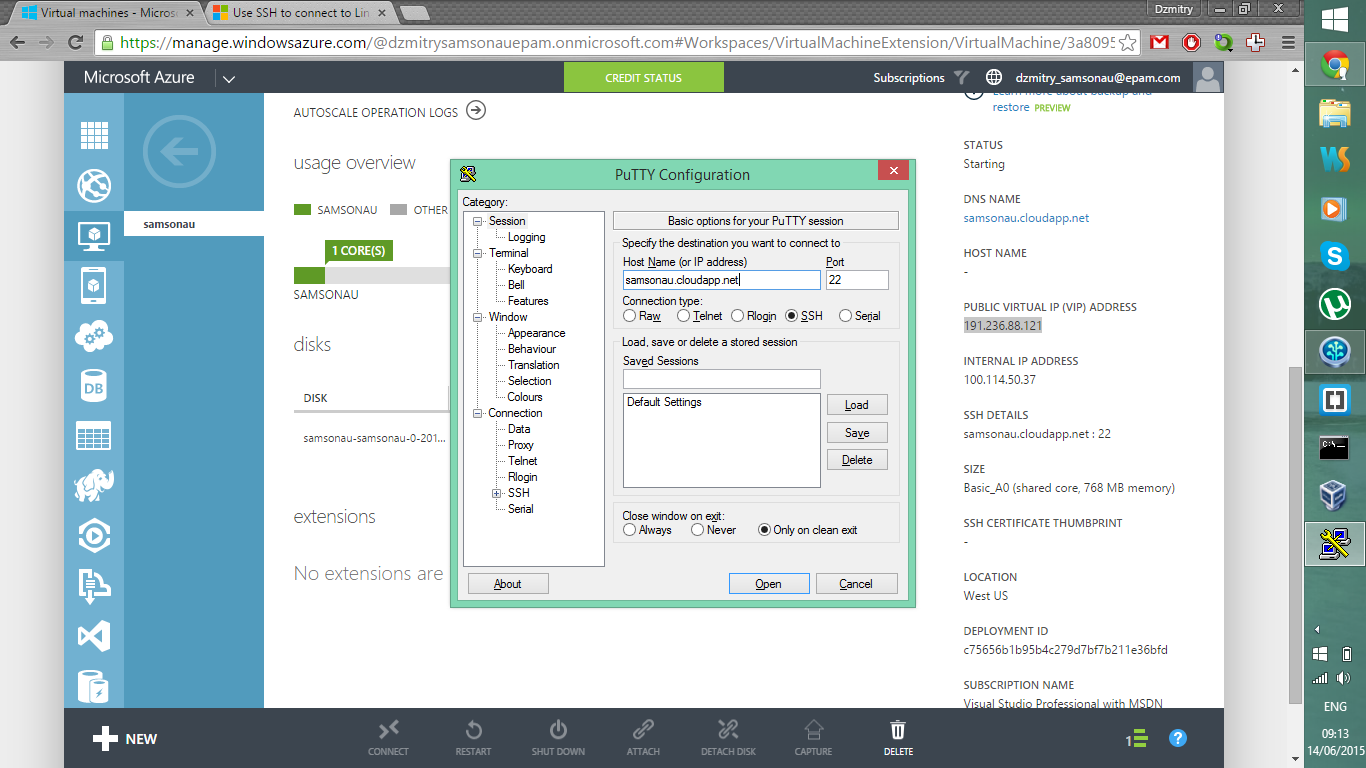
Рисунок 3.2 – Связь между файлами в рабочей директории и filelog'ом

Для сбора и хранения информации об отслеживаемых файлах Mercurial использует структуру называемую манифест. Каждый раздел в манифесте содержит информацию о файлах, входящих в один набор изменений. В разделе записано, какие файлы присутствуют в наборе изменений, ревизия каждого файла и некоторые другие части метаданных файла.

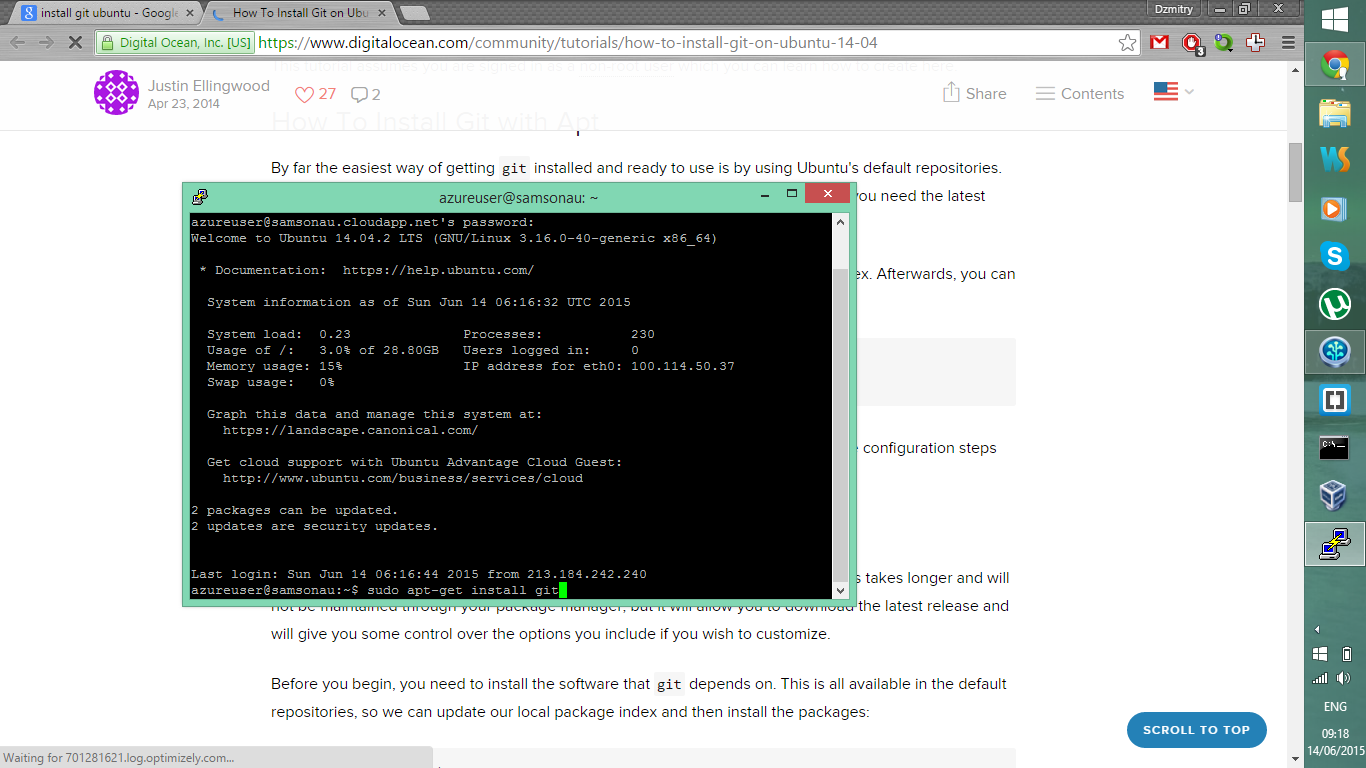
Для хранения внутренней информации Subversion также использует собственную папку – папку .svn. В этой папке хранятся системные файлы и копии рабочих файлов до внесения нами правок. Когда запускася команда status, система SVN сверяет наши файлы с образцами, которые хранятся внутри папки .svn, поэтому для выявления изменений не нужно подключаться к удаленному серверу.

**3.3 Установка Git на сервер**

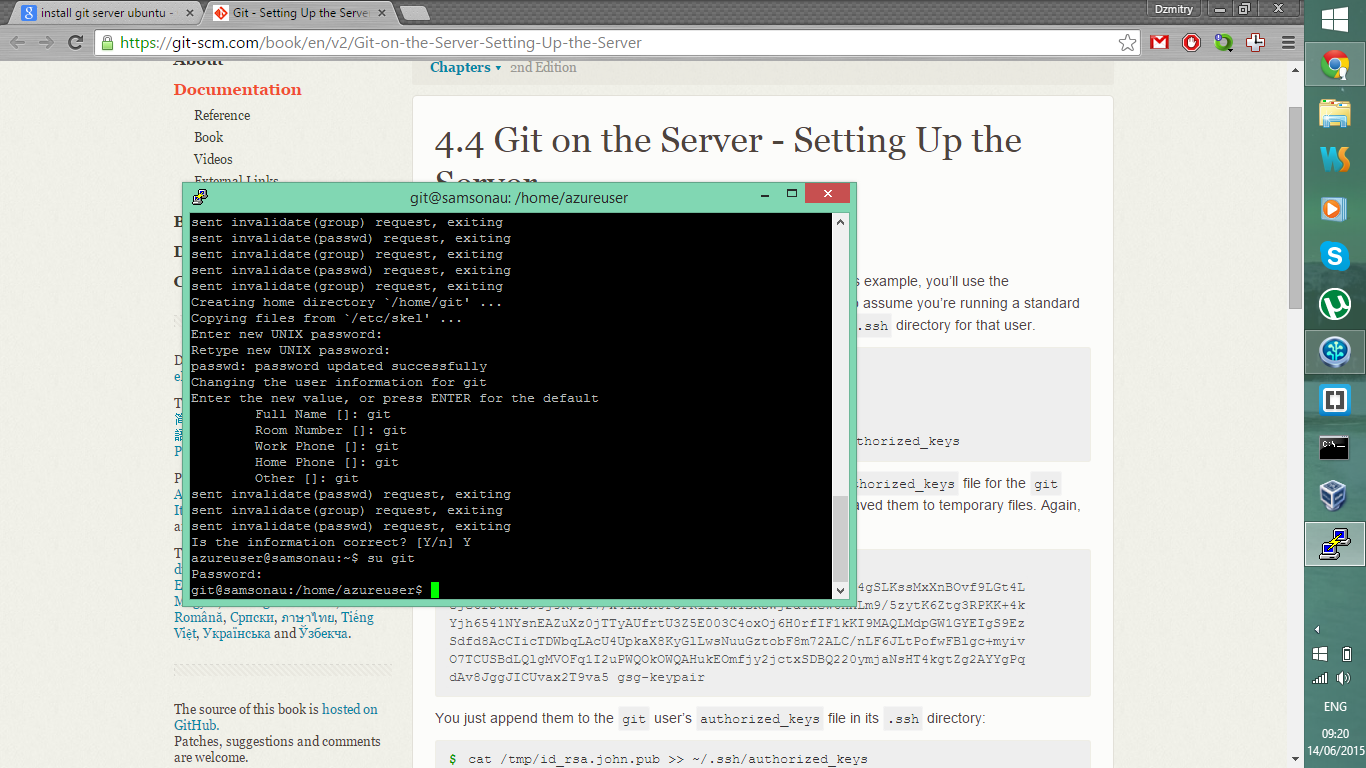
В данном разделе будет описано, как установить Git-систему на виртуальный сервер Windows Azure Ubuntu 14.04 LTS. Несмотря на специфичность, данные шаги можно воспроизвести на любом другом сервере.

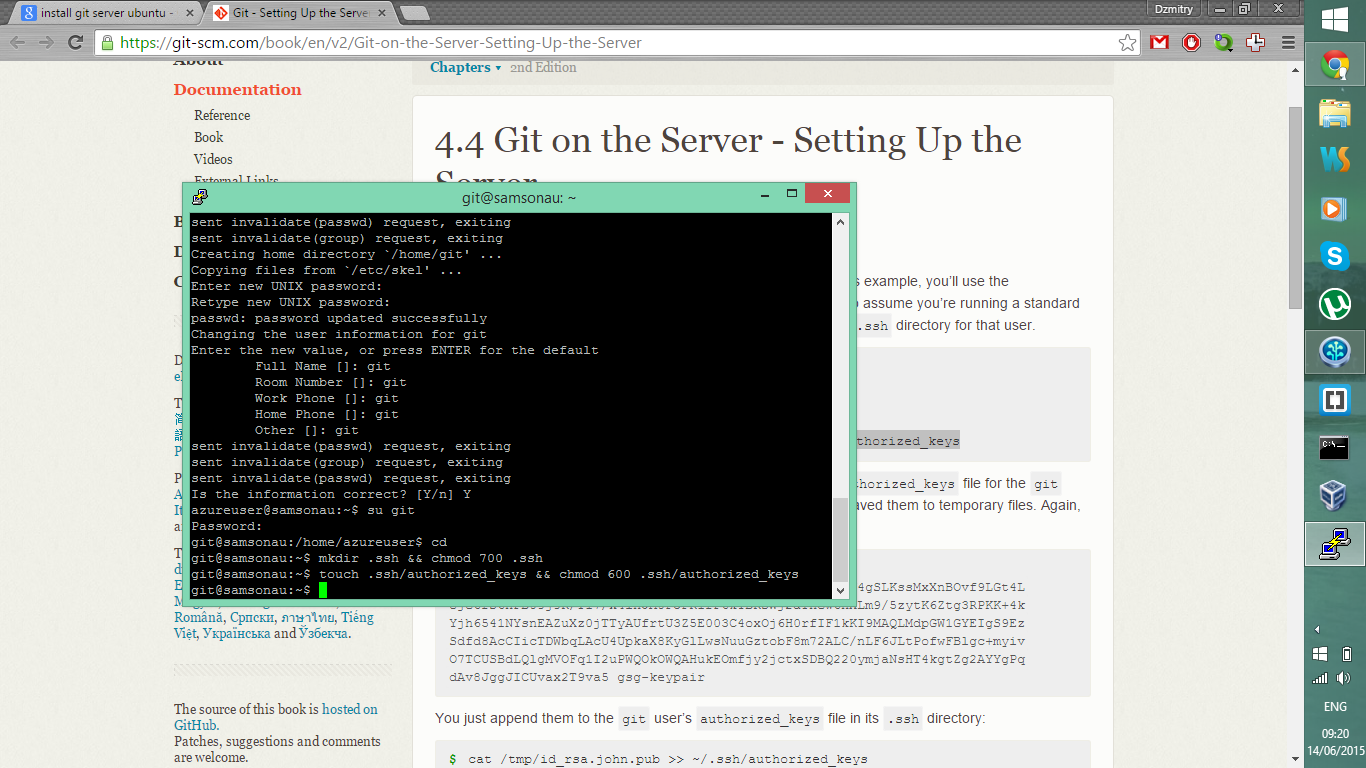


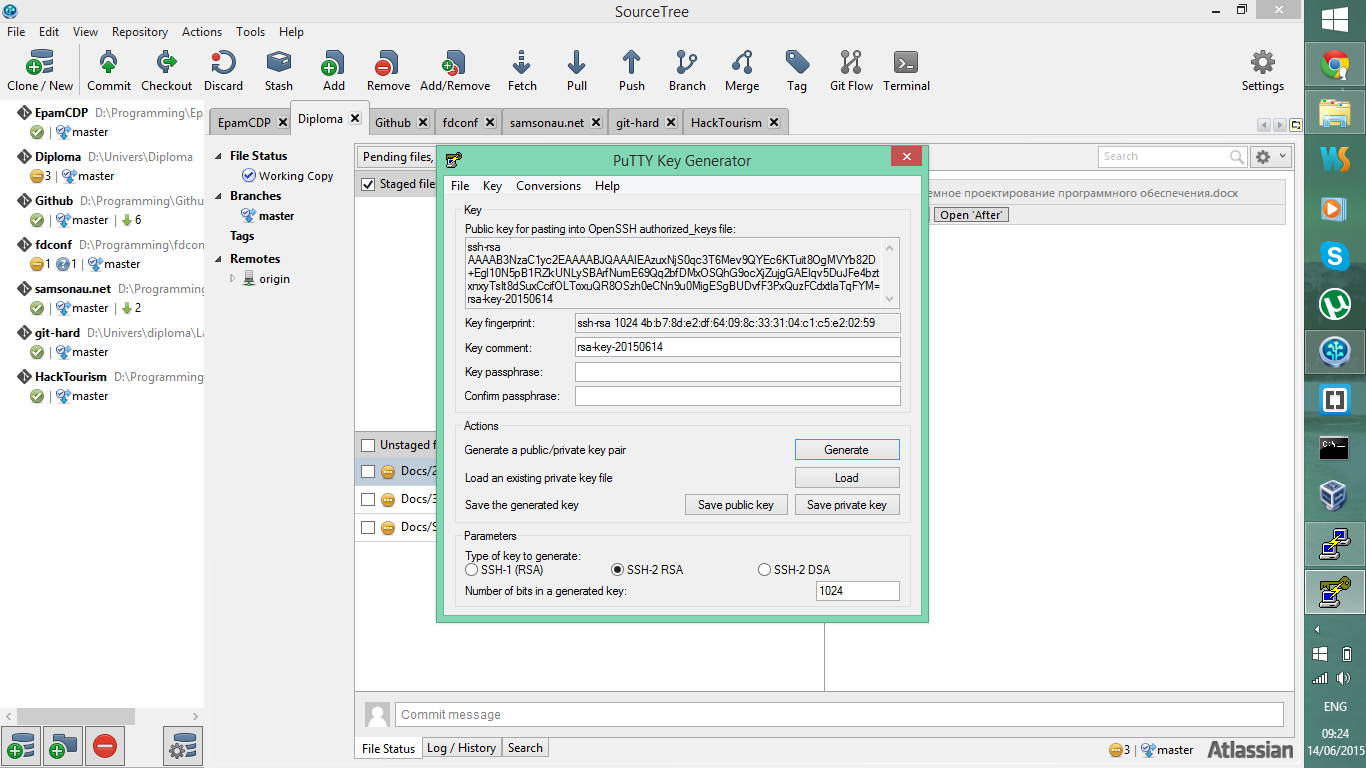


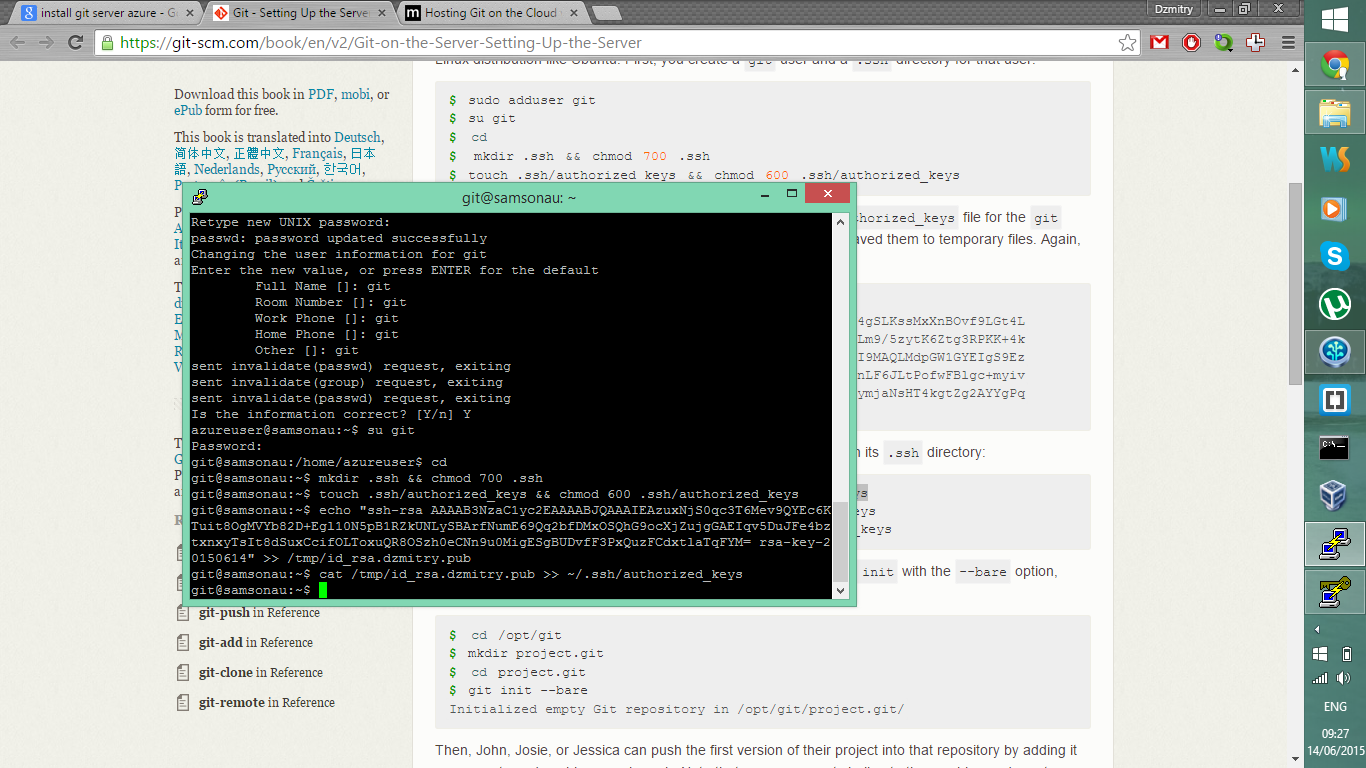


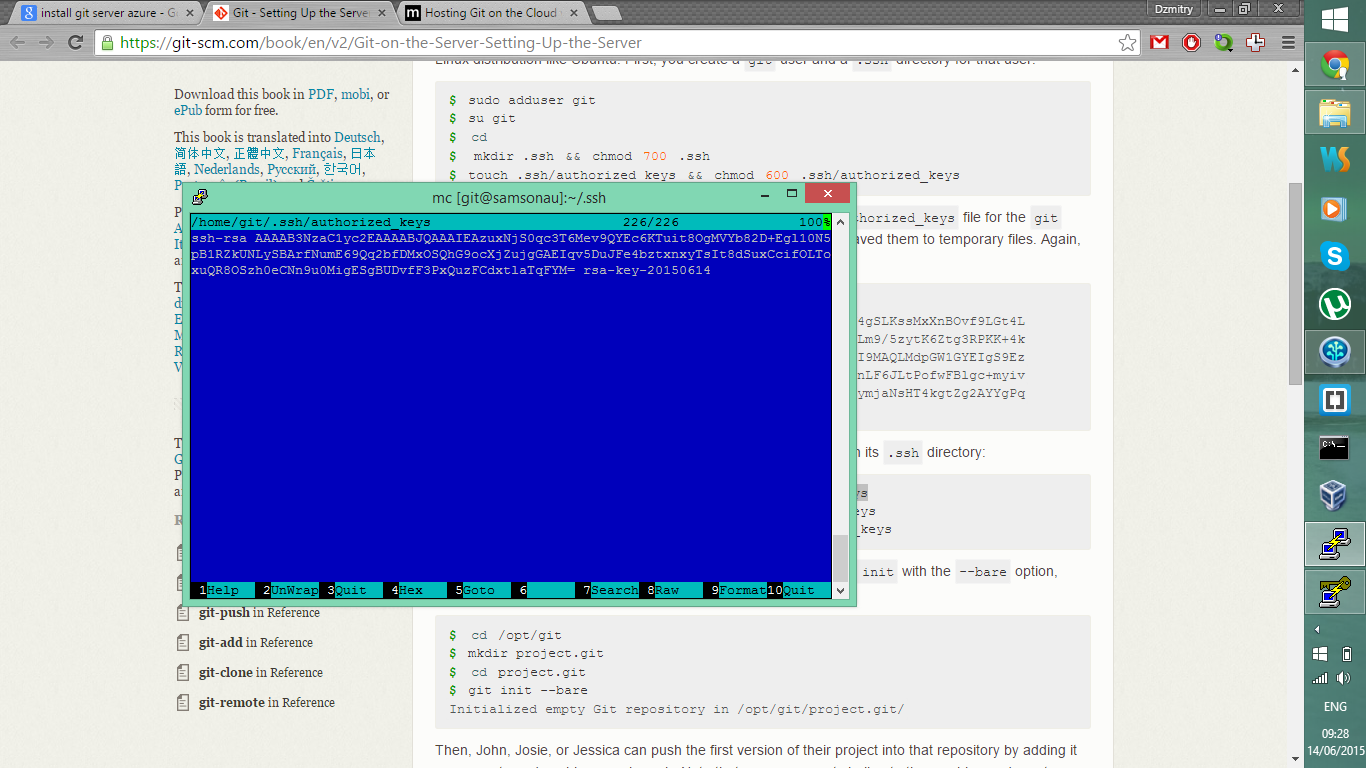


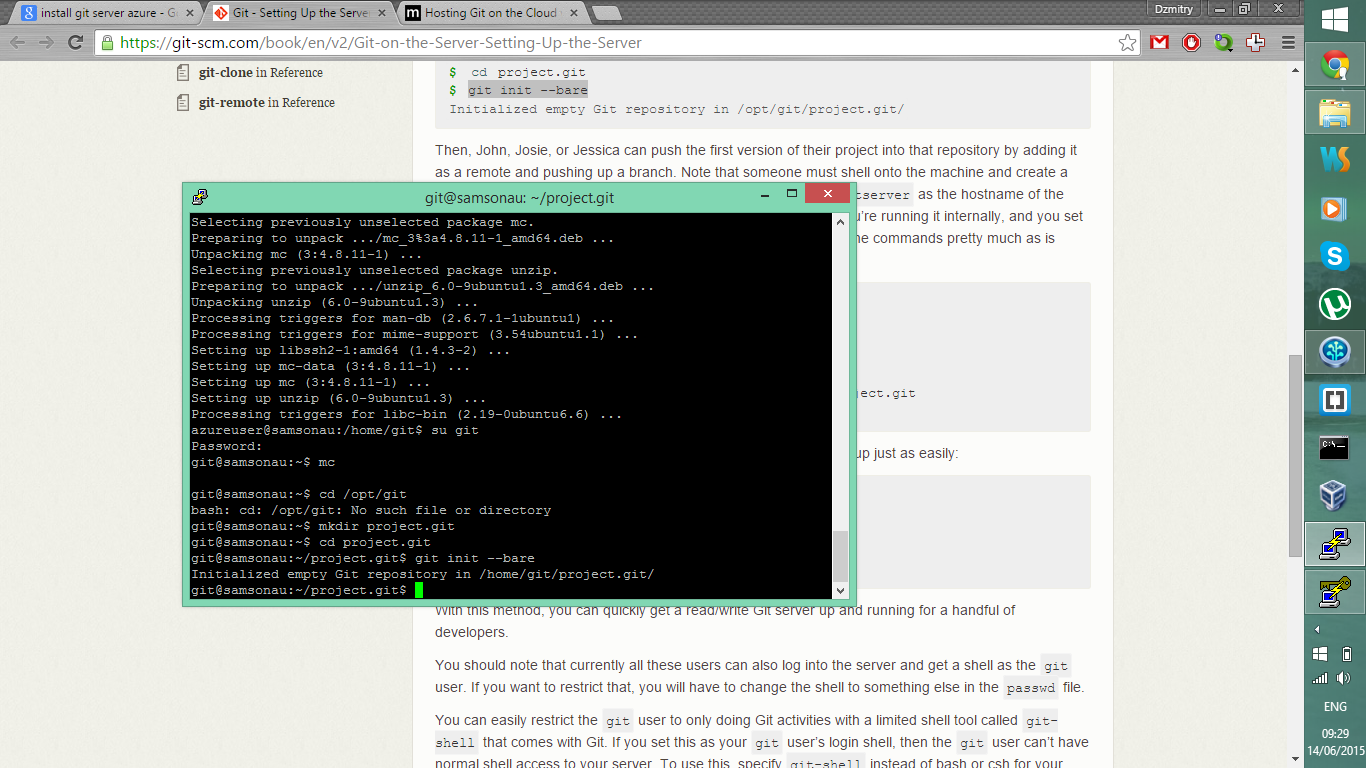


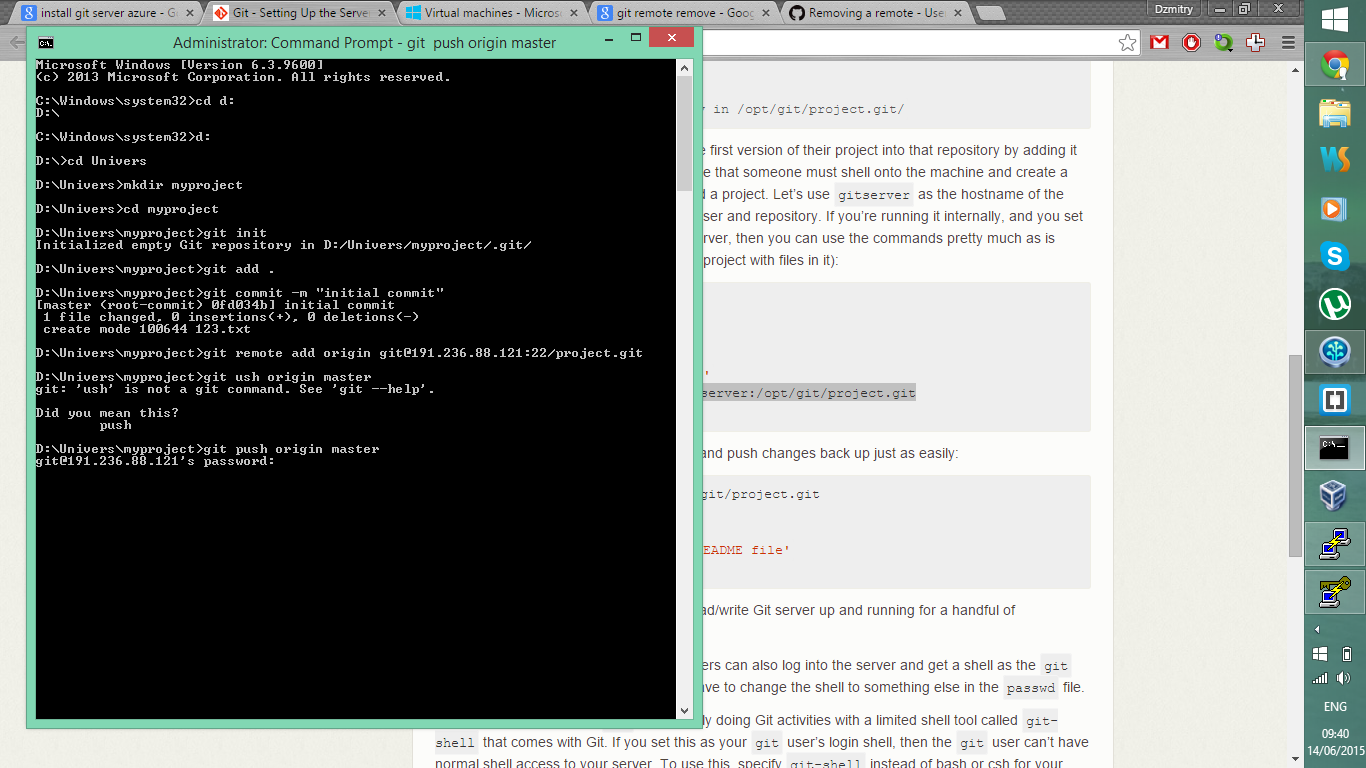


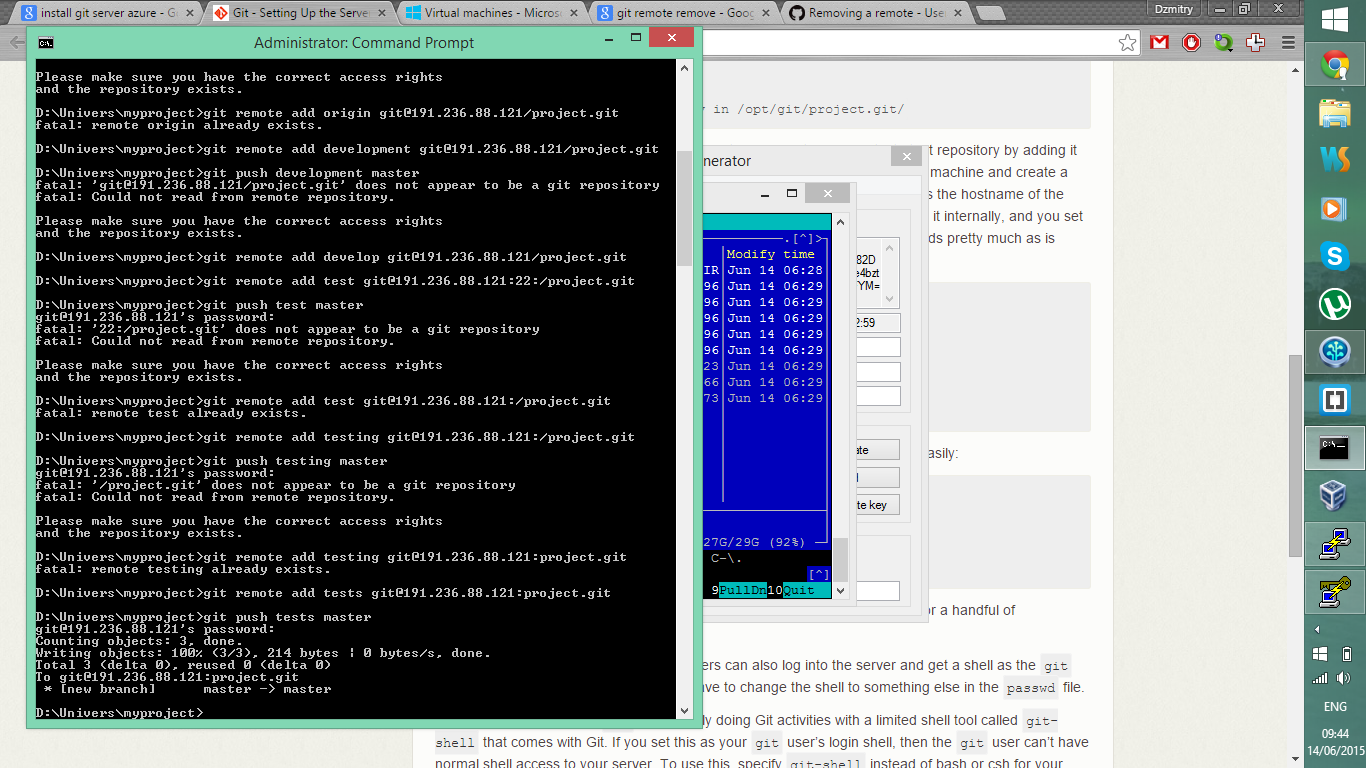


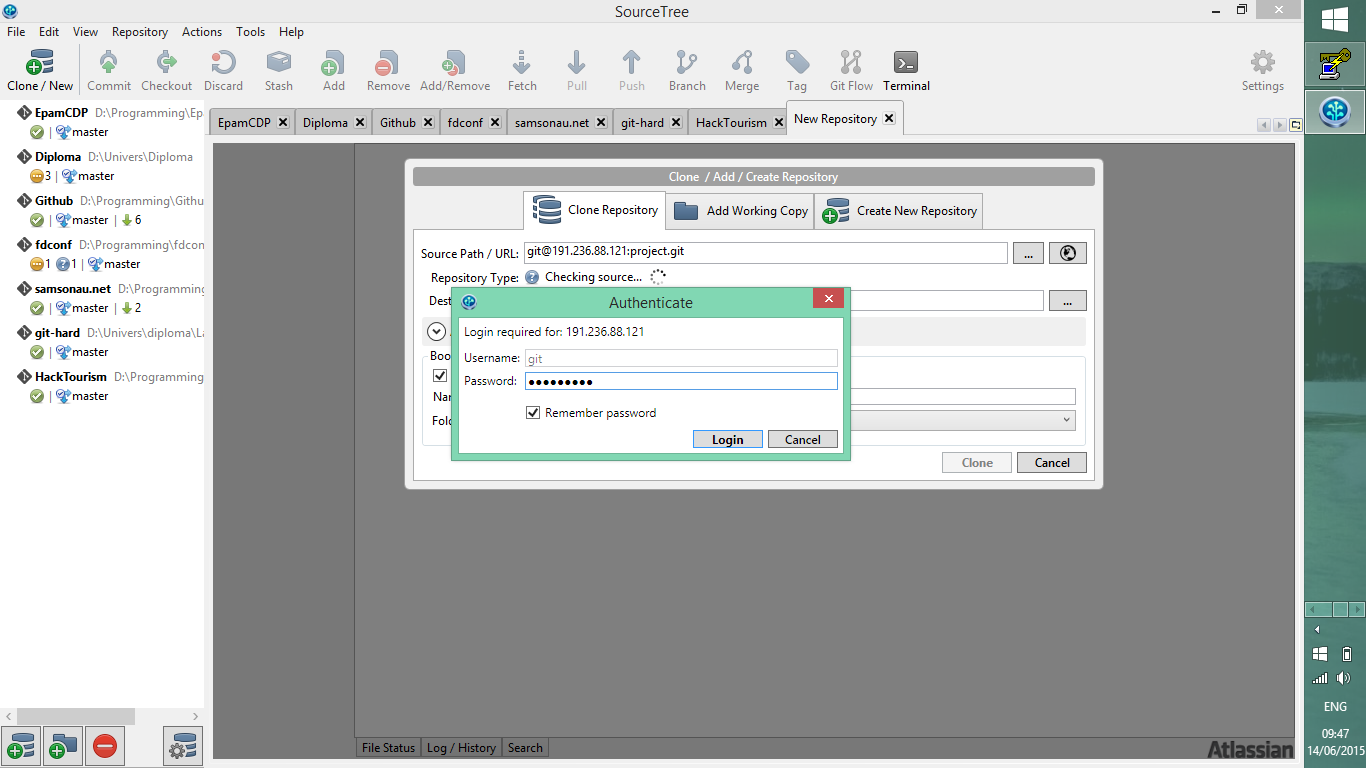


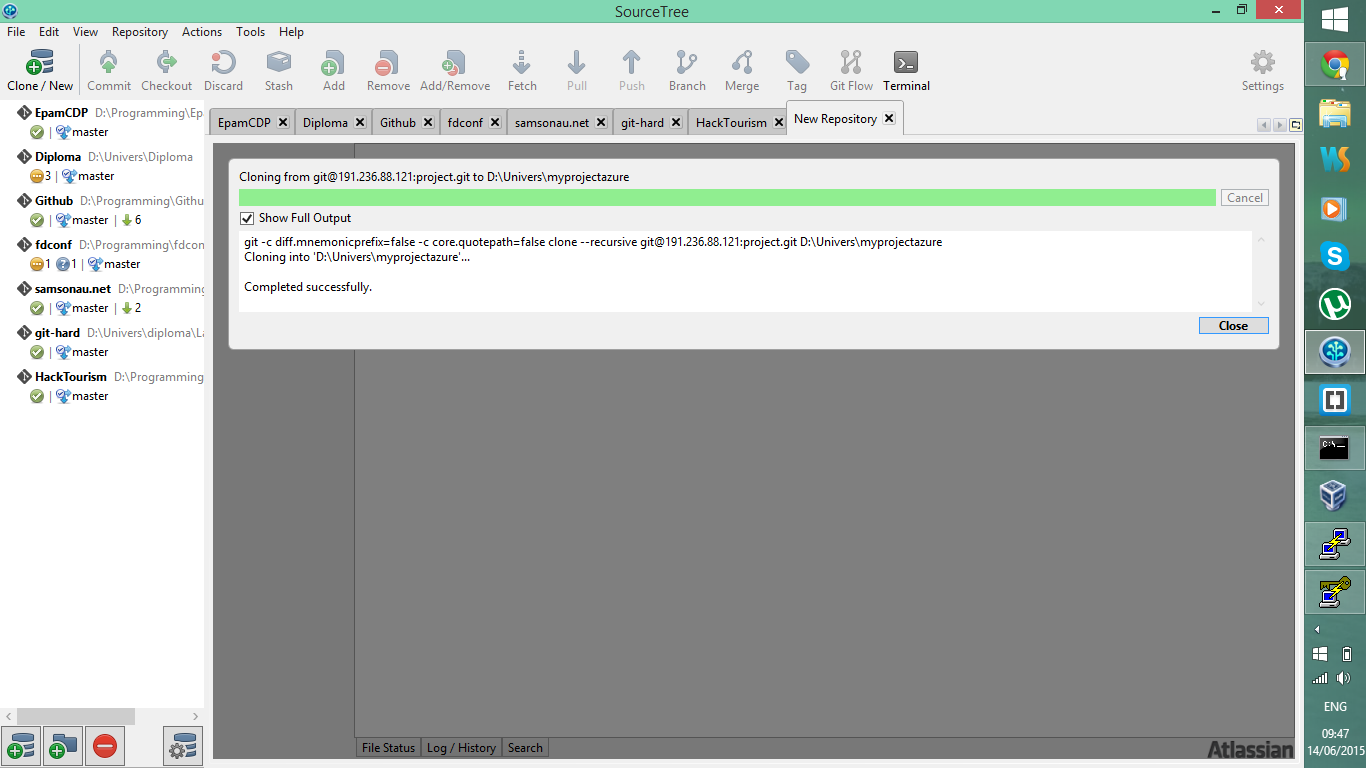


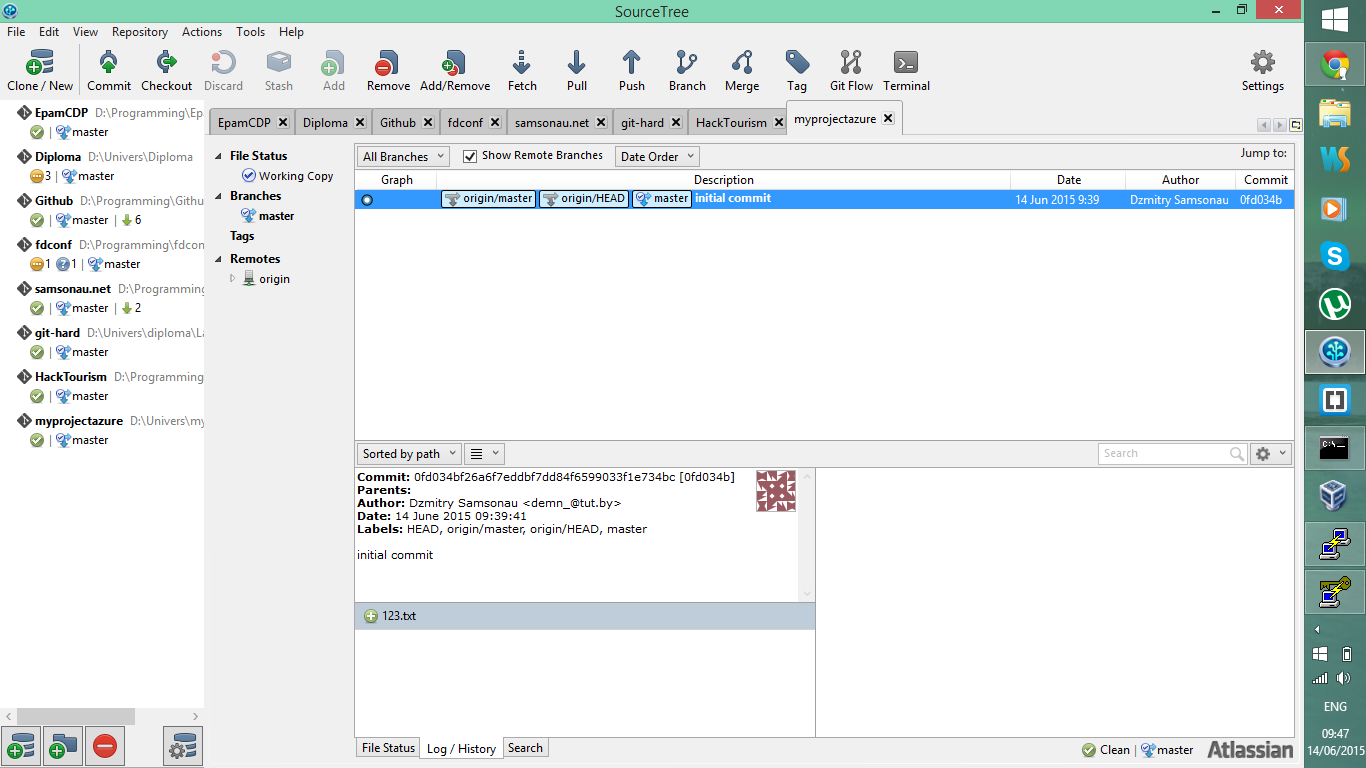


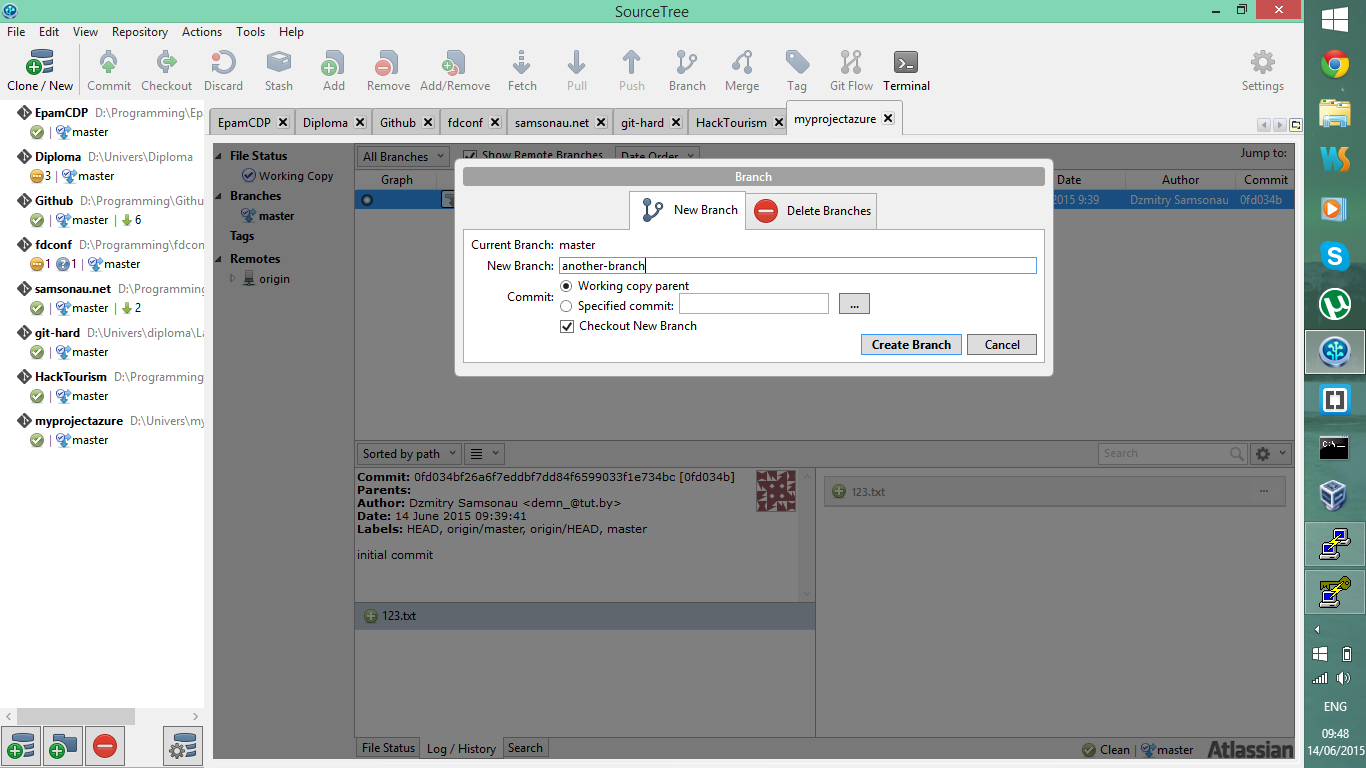


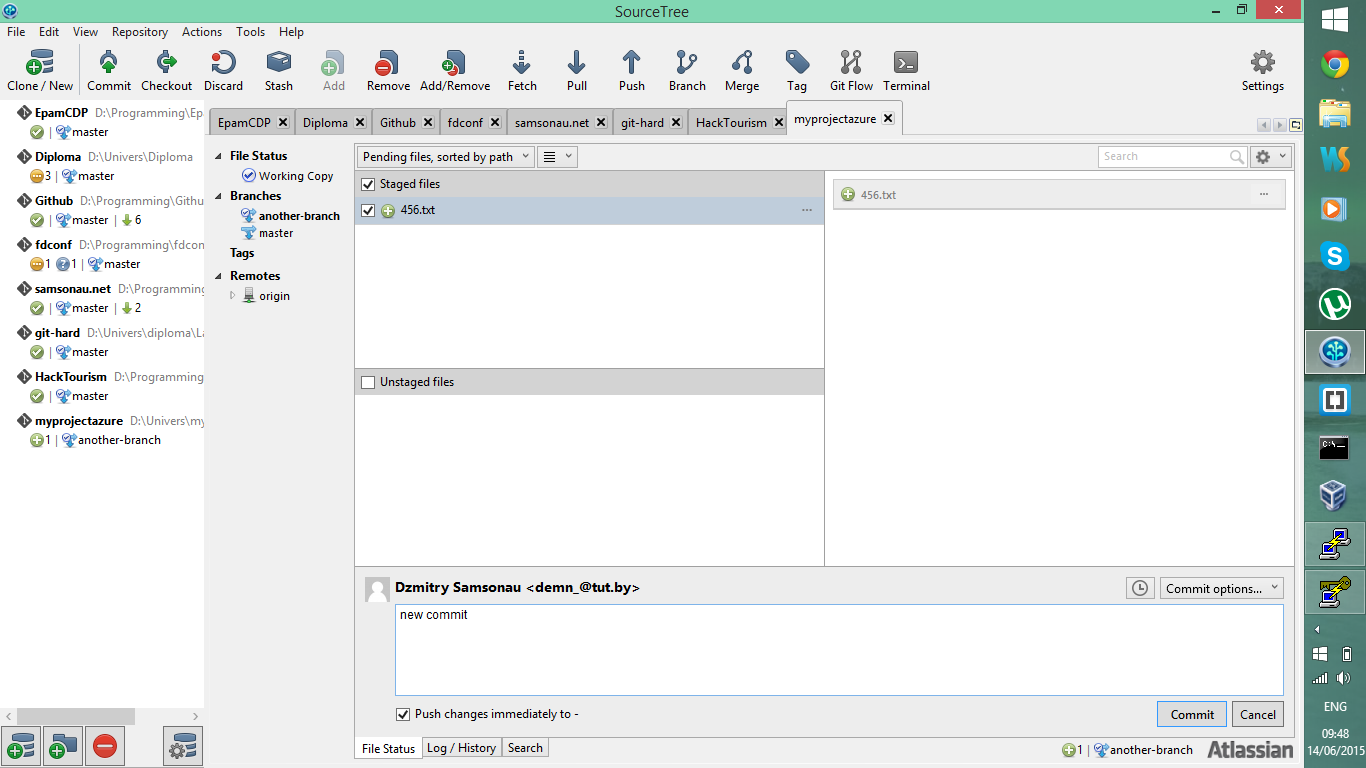


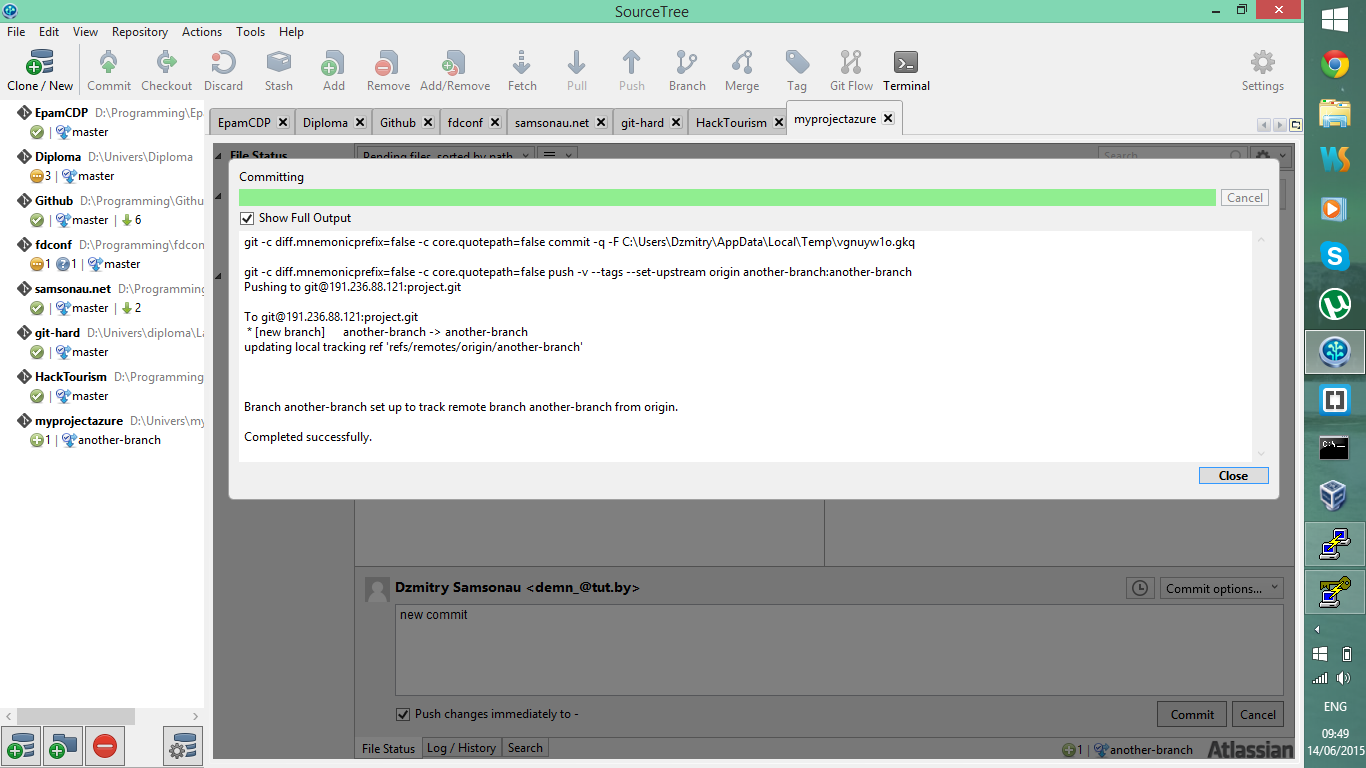














**3.4 Установка Subversion на сервер**

В данном разделе будет пошагово описано, как установить Subversion-систему. В этом примере также будем использовать сервер Ubuntu 14.04 LTS.

Для установки Subversion необходимо сначала установить все необходимые библиотеки. Это делается командой sudo apt-get install subversion apache2 libapache2-svn apache2-utils

После этого необходимо создать новую папку и новый репозиторий (в данном примере -- testrepo)

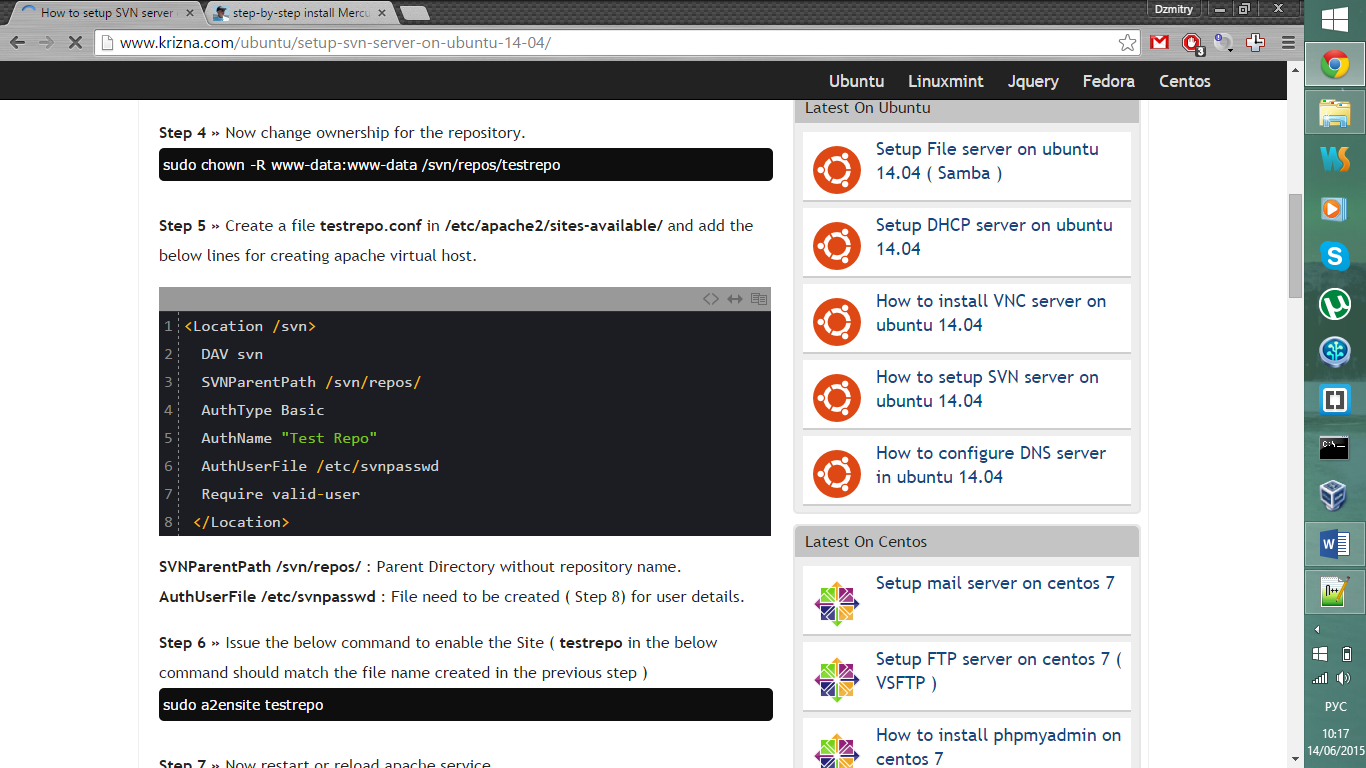
sudo mkdir -p /svn/repos/

sudo svnadmin create /svn/repos/testrepo

Далее нужно подменить владельца данного файла:

sudo chown -R www-data:www-data /svn/repos/testrepo

Теперь создадим файл testrepo.conf в /etc/apache2/sites-available/ и добавим строки таким же образом, как на рисунке:



где

SVNParentPath /svn/repos/ -- родительская папка без имени репозитория

AuthUserFile /etc/svnpasswd -- файл, который нужно создать для добавления данных пользователя

Теперь можно добавить сайт:

sudo a2ensite testrepo

И перезапустить Apache-сервер:

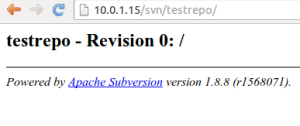
sudo service apache2 reload

Добавление данных в файл с доступом делается следующим образом

sudo htpasswd -cm /etc/svnpasswd user1 -- для добавления первого пользователя

sudo htpasswd -m /etc/svnpasswd user2 -- для добавления последующих пользователей

Теперь можно запустить браузер по адресу http://yourip/svn/testrepo (где yourip -- адрес сервера) и проверить правильность работы сервера:



Если сайт загрузился, значит можно начинать непосредственную работу с репозиторием.