Перед любым разработчиком ПО рано или поздно встает вопрос автоматизации тестирования, сборки и выкладывания проекта.

И из этой потребности постепенно выросла целая ниша ПО - системы непрерывной доставки и интеграции или **CI/CD**.

Эти системы позволяют быстро и удобно организовать автоматическое тестирование и деплой разрабатываемого приложения. При этом каждый этап контролируется машиной, а также визуализируется для человека.

10:16

Одной из таких систем является **Jenkins**, с которым ты в этом уроке будешь знакомиться.

Дальше

10:17

**Jenkins** написан на **Java**. А это значит что перед его установкой надо подготовить сервер.

Традиционно использовать в качестве ОС будем **Ubuntu 22.04.** Все шаги 100% сработают на ней. С другими дистрибутивами могут быть нюансы.

10:17

Если у тебя уже установлена и настроена **Java 11**, то можешь пропустить этот кусок урока.

Установить Java

10:17

**Перед началом обновите систему!**

Для работы всех этих программ требуется установить интерпретатор **Java** 11-й версии. В **Ubuntu** с этим все достаточно просто.

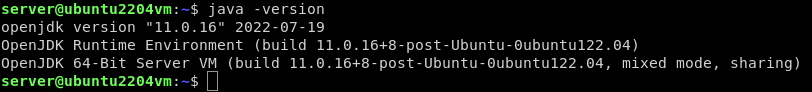
sudo apt install openjdk-11-jre

Проверим, что установка прошла успешно:

java -version

Ответ должен быть похож на тот, что на скриншоте.

10:17



Дальше

10:17

После установки интерпретатора надо провести еще ряд манипуляций, чтобы все работало корректно (почему это не делается

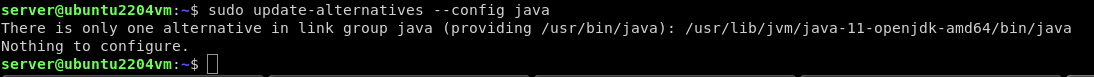
инсталляционным пакетом не представляю). Цель манипуляций - установка в системе переменной среды **JAVA\_HOME**.

А что же в нее писать? Сейчас узнаем с помощью следующей команды:

sudo update-alternatives --config java

Ты должен получить что-то похожее на скриншот ниже

10:17



Дальше

10:17

В списке может быть и несколько пунктов, если ты установил несколько версий **java**, то и пунктов в списке будет несколько.

Нужно выбрать тот, что соответствует 11-й версии и скопировать путь в скобках.

Теперь открывай текстовым редактором файл **/etc/environment** и давляй туда следующую строку (не забывай про **sudo** ;) ):

**JAVA\_HOME="тут\_надо\_вставить\_путь\_из\_предыдущего\_шага"**

10:17

Сохраняй файл. А теперь выполни вот эту команду:

source /etc/environment

Она применит внесенные нами изменения в текущем сеансе.

Дальше

10:17

10:17

Вот и все. **Java** установлена и готова к работе. Можно ставить написанные на этом языке приложения.

Дальше

10:17

10:17

Теперь перейдем к установке **Jenkins.**

Инструкция:

<https://www.jenkins.io/doc/book/installing/linux/>

и тут

<https://www.digitalocean.com/community/tutorials/how-to-install-jenkins-on-ubuntu-22-04>

Дальше

10:17

В официальном репозитории его нет. Зато разработчики поддерживают свой собственный репозиторий, который мы сейчас подключим.

wget -q -O - <https://pkg.jenkins.io/debian-stable/jenkins.io.key> |sudo gpg --dearmor -o /usr/share/keyrings/jenkins.gpg

sudo sh -c 'echo deb [signed-by=/usr/share/keyrings/jenkins.gpg] <http://pkg.jenkins.io/debian-stable> binary/ > /etc/apt/sources.list.d/jenkins.list'

sudo apt update

10:17

И установка:

sudo apt install jenkins

Дальше

10:18

В случае с этим программным пакетом никакие предварительные настройки не требуются.

Так что сразу его запускаем и прописываем в автозагрузку:

sudo systemctl enable jenkins

sudo systemctl start jenkins

Немного подожди. Это все же **Java**. Но в течение минуты все точно запустится.

Запущенный **Jenikns** будет слушать порт **8080 tcp.**

Ну и проверка будет выглядеть вот так:

sudo netstat -lpn | grep 8080

10:18



Дальше

10:18

По умолчанию в **ubuntu 22.04** файрвол не настроен. Но если вдруг у тебя он настроен, то предлагаю сразу открыть порт **8080** хотя бы для себя, чтобы подключаться к веб-интерфейсу Jenkins.

Но перед тем как перейти в веб-интерфейс, тебе нужно узнать одну строку. Она зовется **Admin Initial Password** и ее надо будет ввести, чтобы в первый раз войти в веб-интерфейс.

sudo cat /var/lib/jenkins/secrets/initialAdminPassword

10:18

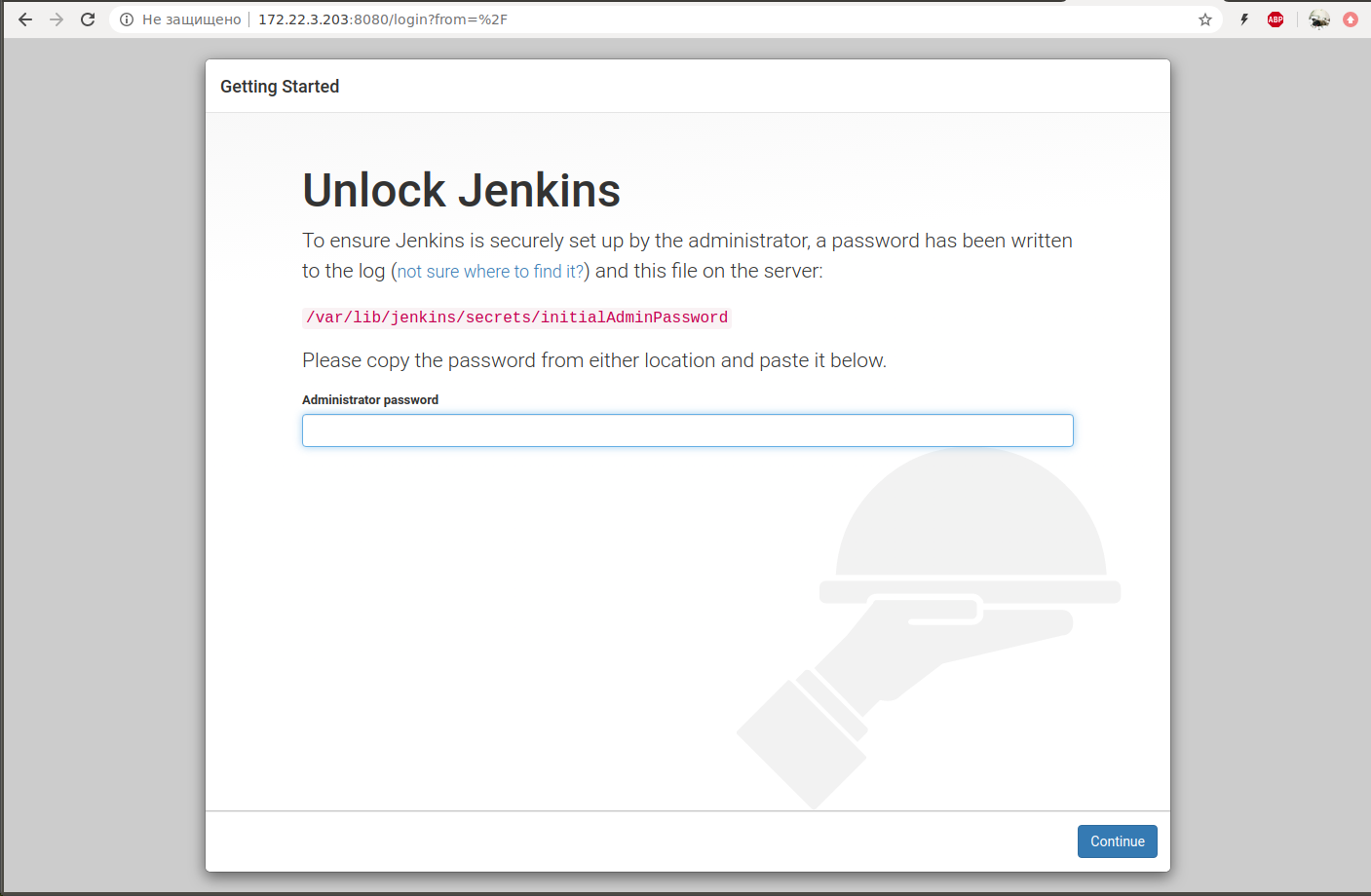


Дальше

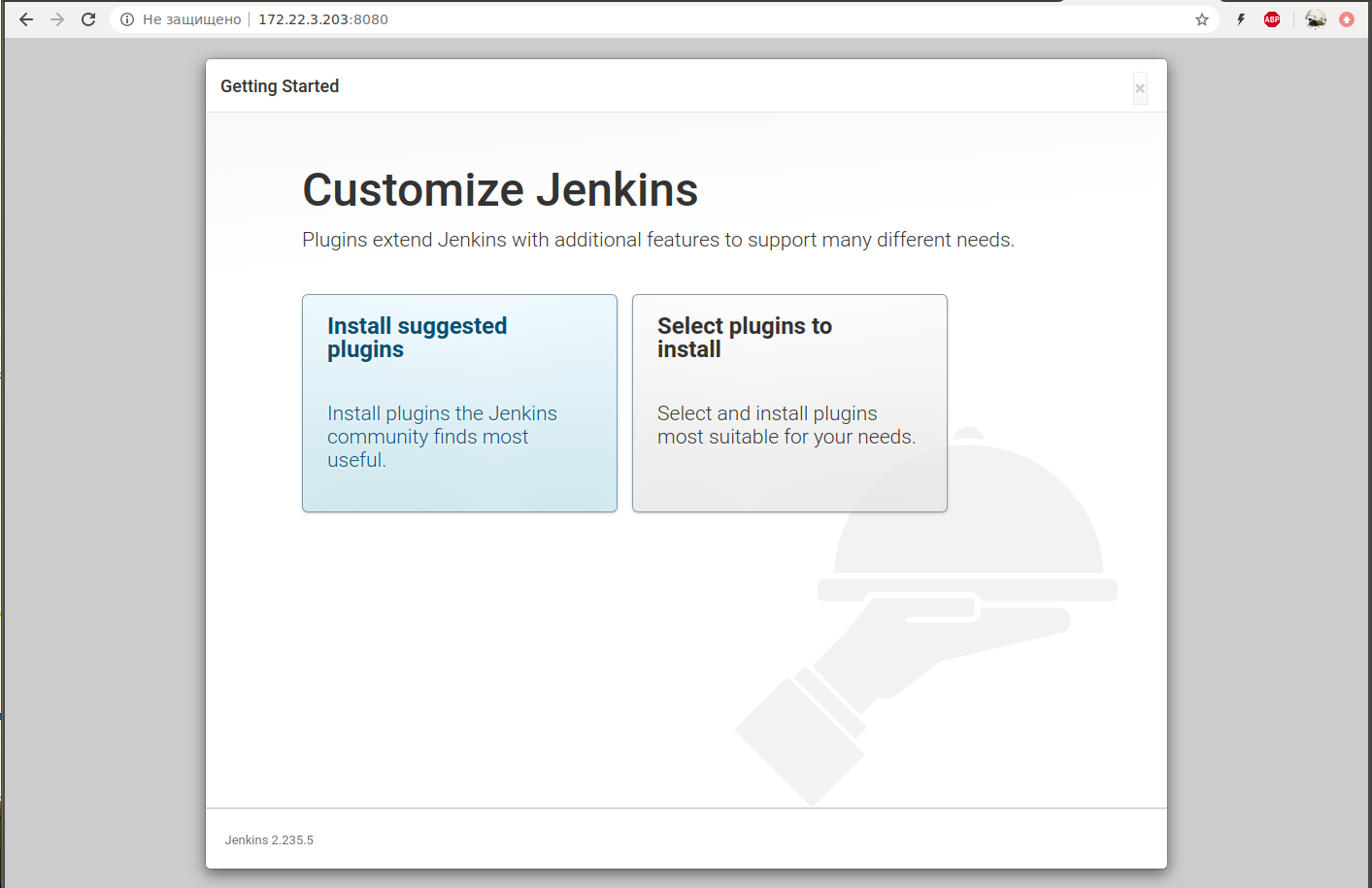
10:18

Пришло время подключиться к веб-интерфейсу.

Открывай в браузере адрес: [**http://ip\_твоего\_сервера:8080**](http://xn--ip__-83dkbgrcb6hc4aclk:8080/)



Вводи строку, которую ты получил чуть ранее. Так попадешь в дальнейшую настройку.



10:18

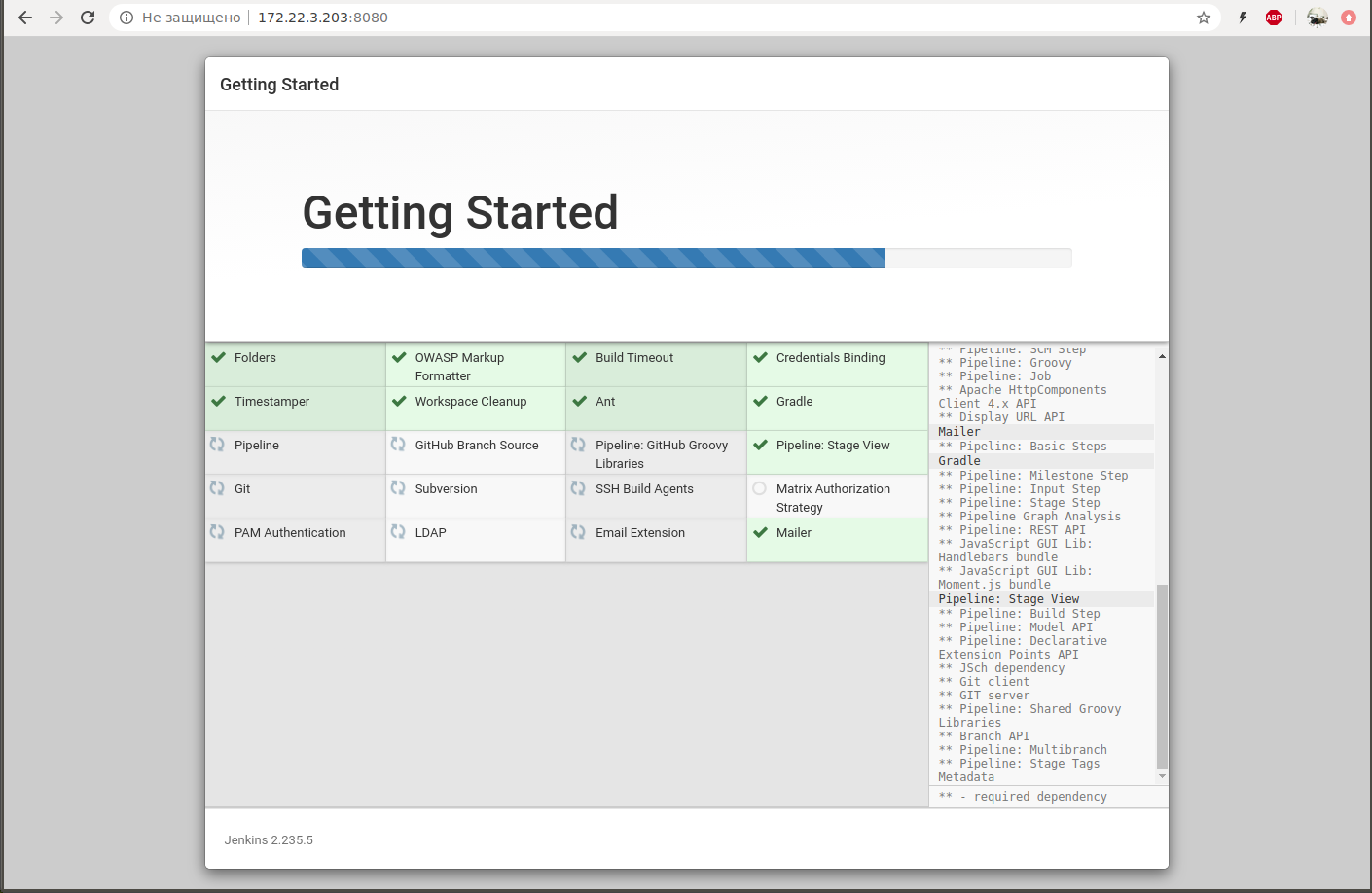
Тут чтобы сильно мозг не напрягать, жми квадратик слева. Пусть поставятся все плагины. В другой раз уже с опытом сможешь выбрать какие тебе нужны, а какие нет.

Дальше

10:18

Увидишь вот такую картину. Теперь (как писали в инсталляторе Win98) ты можешь отдохнуть и выпить чашечку чая, т.к. скачивание и установка плагинов займет некоторое время.

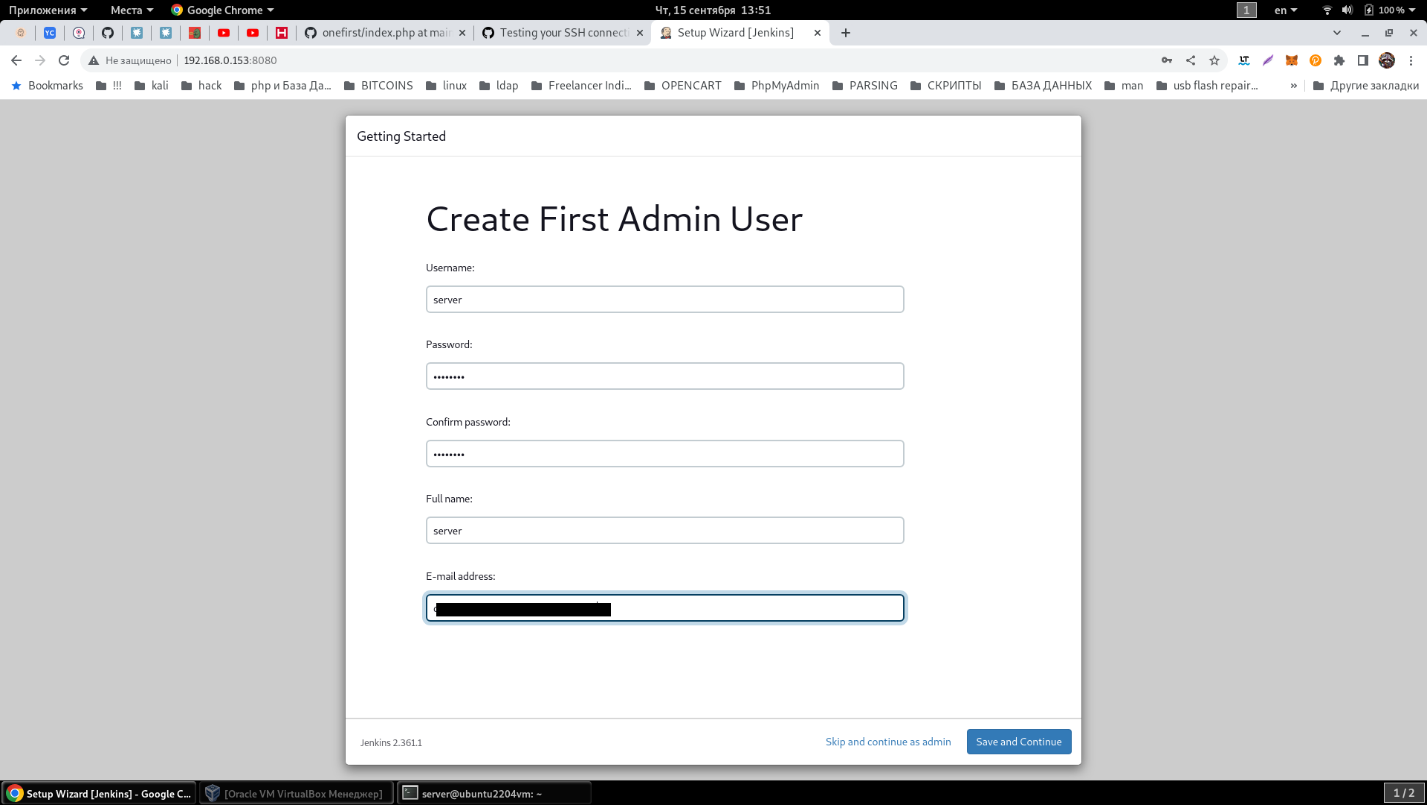
10:18



Дальше

10:18

Как только установка завершится тебе будет предложено создать первого администратора сервиса.



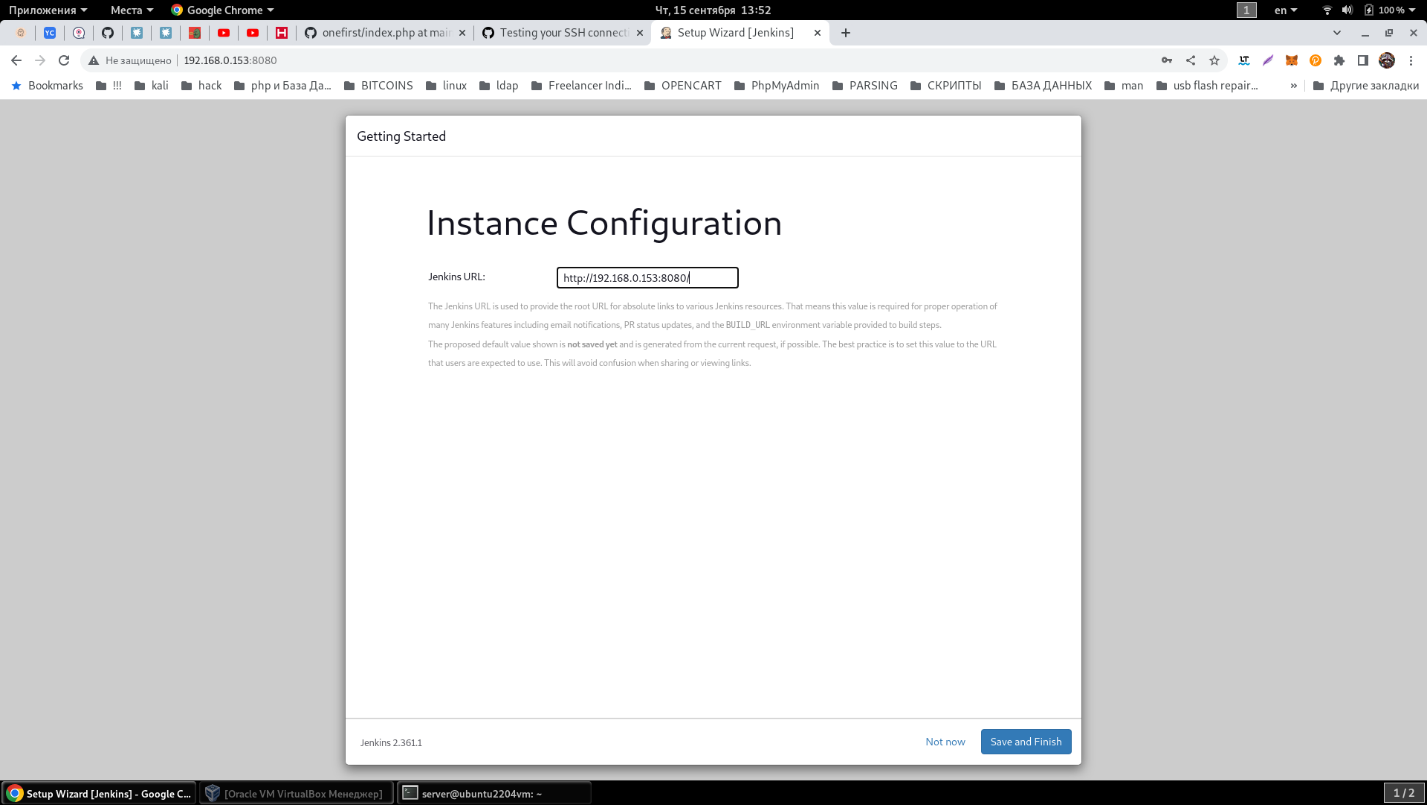
10:18

Не обязательно повторять за мной. Придумывай свои учетные данные.

Дальше

10:18

На следующем шаге тебе будет предложено задать адрес, по которому будет доступна панель **Jenkins**



Ты можешь оставить как есть, а можешь завернуть ее на **127.0.0.1** и предоставлять доступ через nginx и доп. пароль (либо вообще по специальному сертификату).

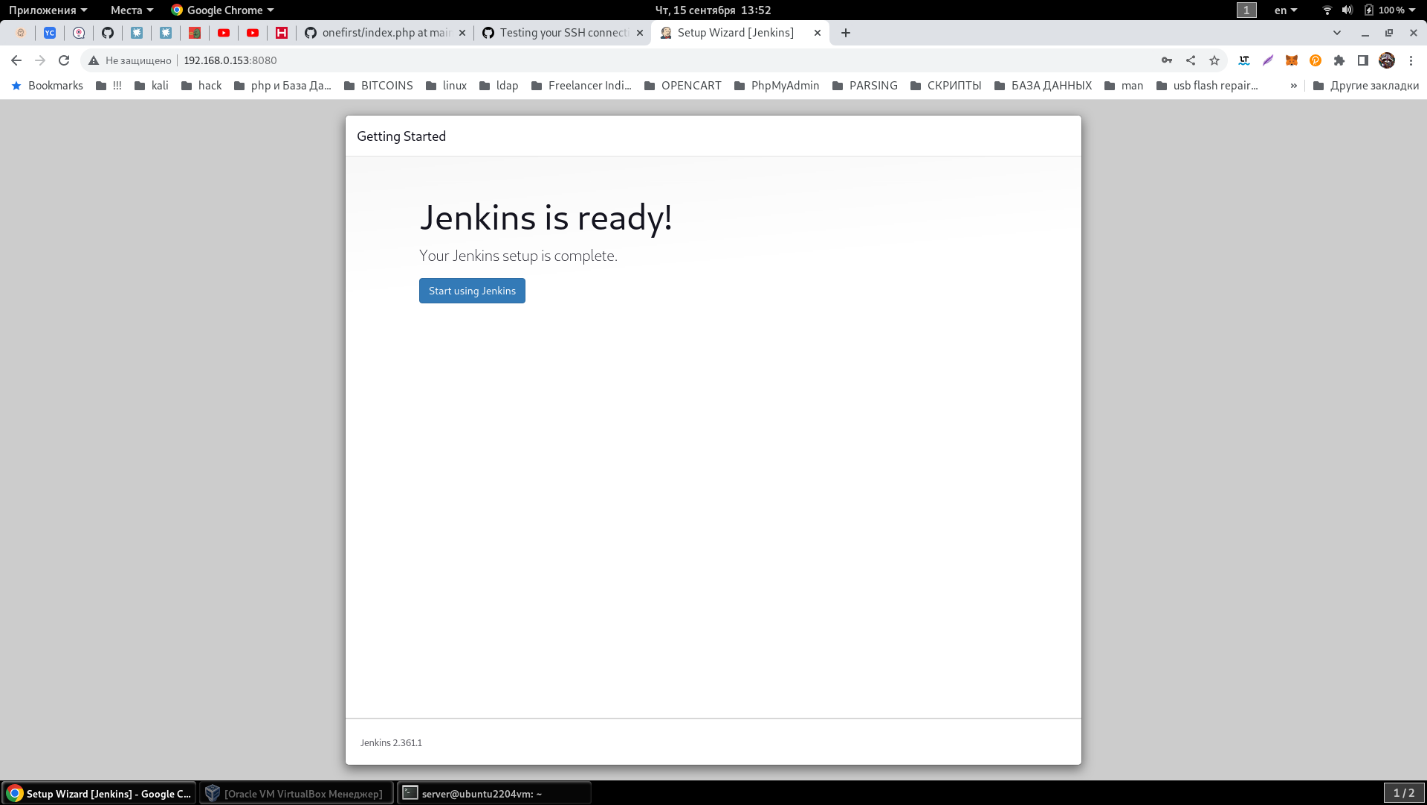
10:18

Оставлять веб-морду на белом ip строго не рекомендую. Несмотря на то, что тут доступ по логину и паролю, существуют еще и уязвимости, через которые тебя могут взломать.

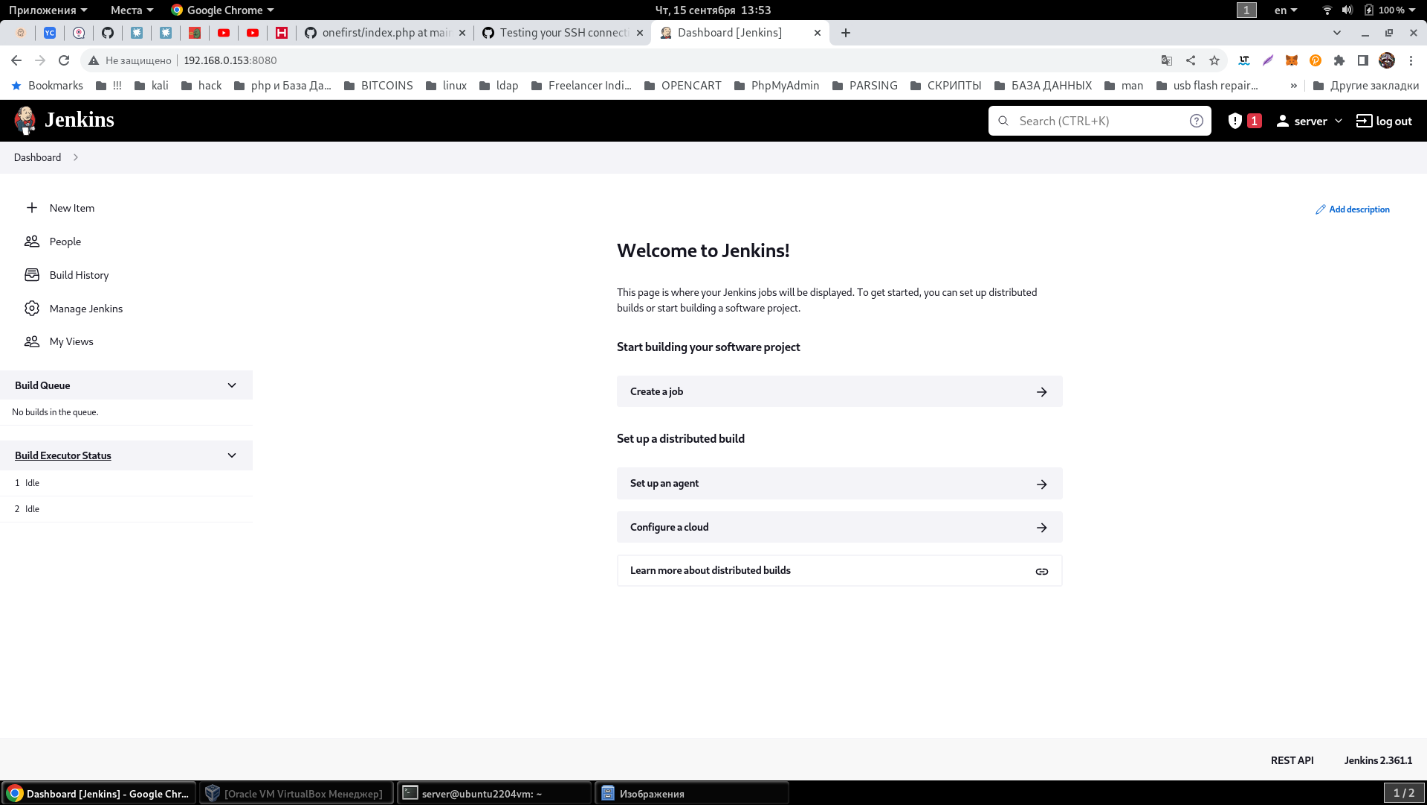
Дальше

10:18

А вот и завершение установки =).



После нажатия на кнопку ты увидишь вот что



10:18

Все. **Jenkins** установлен и можно переходить к самой сути урока.

Дальше

10:18

10:18

Для того, чтобы проиллюстрировать работу с **CI/CD** на примере **Jenkins** мы с тобой разберем следующую задачу:

Допустим у тебя есть сайт, который ты разрабатываешь. Ты хранишь файлы в **git**-репозитории и хочешь, чтобы после коммита происходило тестирование свежих данных и автодеплой на **веб-сервер**.

Дальше

10:18

10:18

Для выполнения работы потребуется:

1. **Git**-репозиторий, доступный из нашей учебно-виртуальной среды.

2. Отдельный сервер или виртуалка с **nginx+php-fpm** на борту. Это мелочь. На ней я останавливаться не буду. **Тебе придется самостоятельно организовать такой сервер. Не забудьте создать на сервере пользователя jenkins .**

3. **Jenkins** - он у нас есть и ждет настройки.

Дальше

10:18

10:18

У тебя есть **Git**-репозиторий, которым ты можешь пользоваться для урока?

Да

10:18

Итак. У тебя есть репозиторий.

Надо будет в него выложить простенький проект на php. Можно из одного файла. К примеру такой, который я сейчас тебе продемонстрирую.

10:18

<https://pastebin.com/Lz1nCW5n>

Дальше

10:18

Над этим файлом и будем издеваться.

Алгоритм будет такой:

1. Ты загружаешь файл в репозиторий.

2. **Jenkins** по расписанию раз в 5 минут проверяет репозиторий. Если там появились изменения, то:

2.1 Выполняется проверка файла с помощью **php -l**

2.2 Если синтаксис в порядке, то файл заливается на веб-сервер.

10:18

Но для начала создадим ssh ключ. Он потребуются для работы с git.

<https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent>

и создадим токен.

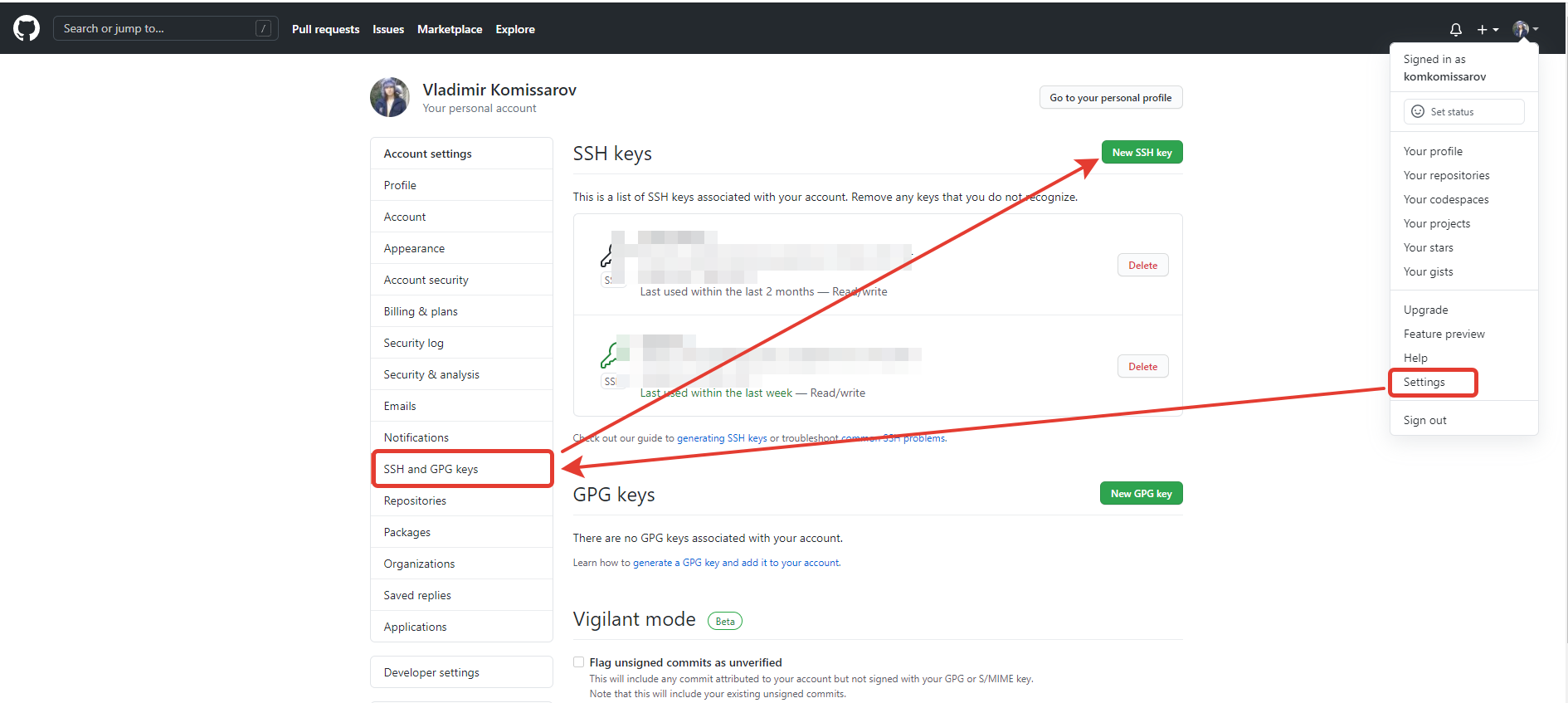
<https://docs.github.com/en/enterprise-server@3.4/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token>

Дальше

10:19

Потом на Github, зайди в Settings, раздел SSH ключи, как на скрине. И добавь туда содержимое своего публичного ключа.

10:19



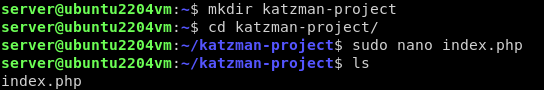
Далее

10:19

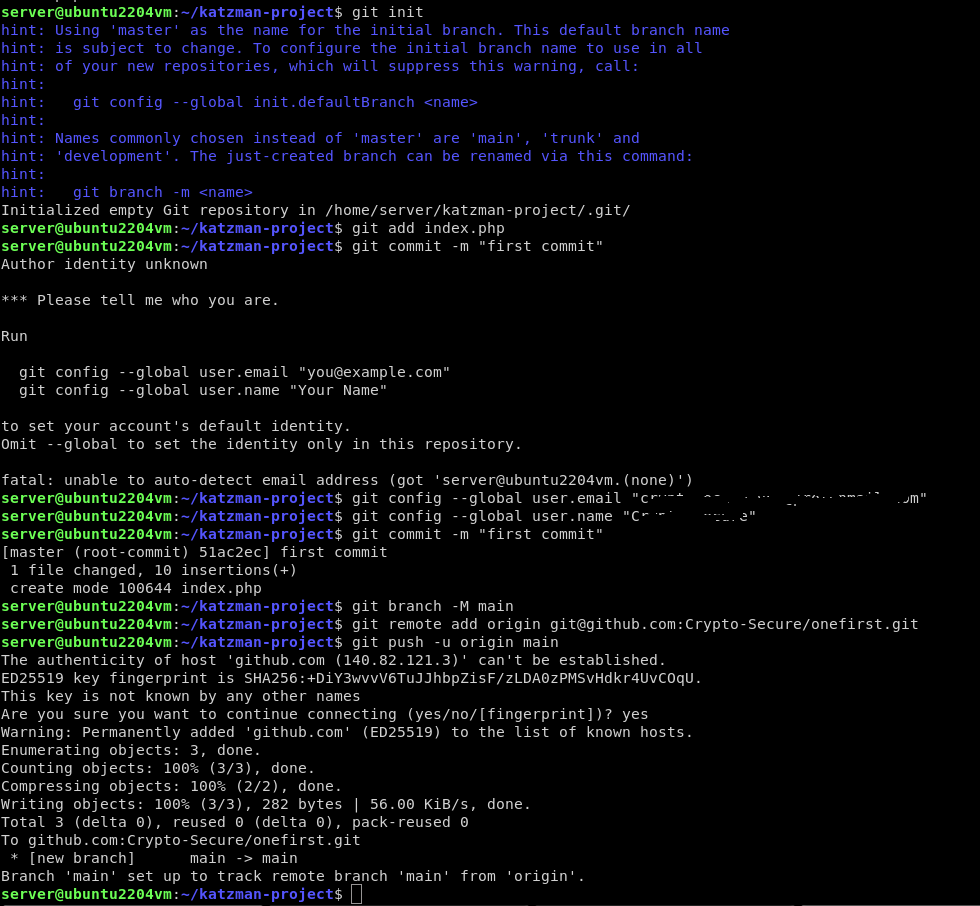
Итак. Начнем с файла и репозитория.

Создавай папку для проекта, заходи в нее, создавай файл, инициализируй проект и загружай его в **git**-репозиторий.

Примерно вот так, как на скриншотах:



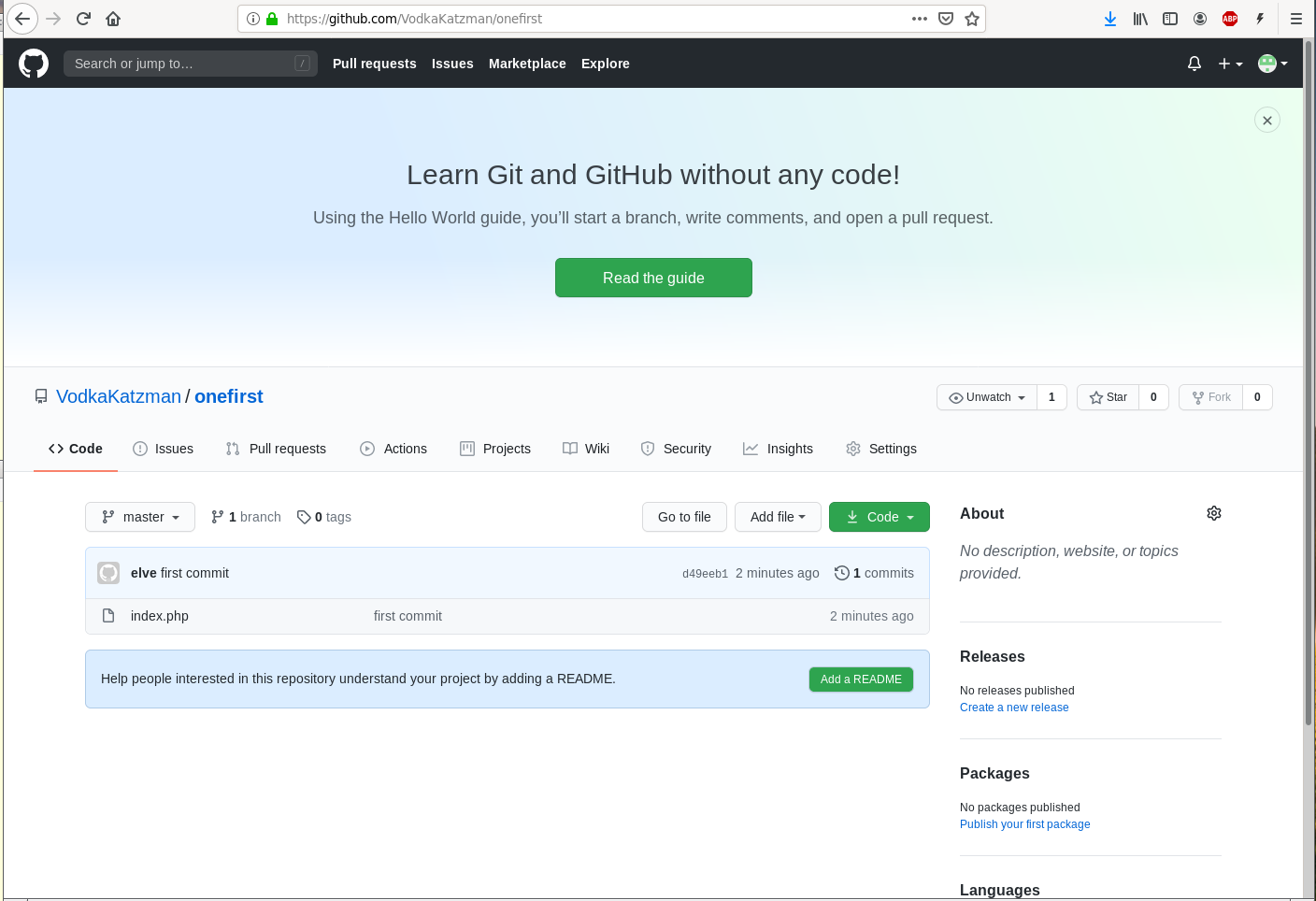
10:19



Дальше

10:19

Если ты все сделал правильно, то увидишь следующее:



10:19

Теперь у тебя есть проект и репозиторий. Можно настраивать разворачивание проекта через **Jenkins**.

Дальше

10:19

Выполнять работы мы будем с помощью bash-скриптов.

Т.к. у **Jenkins** нет агентов, то подключение к **веб-серверу** будет происходить по **ssh**.

А т.к. требуется автоматическое выполнение всех команд, то нужно настроить безпарольный вход без ключа.

10:19

При установке **Jenkins** в системе был создан одноименный пользователь jenkins. От имени этого пользователя и будут работать скрипты (его можно и в sudoers добавлять, если надо).

Предлагаю зайти на сервер и принудительно залогиниться под **jenkins**

sudo su

su jenkins

cd

Дальше

10:19

Итак. Ты пользователь **jenkins** в его домашней папке (**/var/lib/jenkins**)

Теперь надо скопировать наши ssh-ключи, которые мы делали для **git** в **/var/lib/jenkins/.ssh**

**командой pwd можно поверить в какой директории находитесь**

pwd

Создаем директорию:

mkdir .ssh

Теперь, чтобы нам перенести два наших ключа мы их копируем командой:

cp /home/YOUR\_USER/.ssh/id\_ed25519\* /var/lib/jenkins/.ssh/

**если не удается скопировать , тогда перейдите в пользователя (или вернитесь) в root пользователя и повторите попытку.**

10:19

В результате у тебя появится два файла:

**id\_ed25519;**

**id\_**[**ed25519.pub**](https://ed25519.pub/)**;**

**id\_ed25519** ты никуда не девай. Он должен быть только тут.

**id\_**[**ed25519.pub**](https://ed25519.pub/) нужно раскладывать по серверам, куда у **Jenkins** должен быть доступ.

**эти файлы должны иметь права пользователя и группу jenkins**

Дальше

10:19

В принципе ты можешь подключаться из скриптов **Jenkins** любым пользователем. Но для красоты, создай на веб-сервере пользователя **jenkins** с зубодробительным (или вообще отключенным) паролем и **NOPASSWD** в **sudoers**.

Вот так:

**jenkins ALL=(ALL) NOPASSWD:ALL**

От имени этого пользователя **jenkins** на веб-сервере надо будет создать папку **.ssh** и закинуть туда ключ:

mkdir ~/.ssh

chmod 0700 ~/.ssh

Публичный ключ это текстовый файл. Поэтому ты можешь его скопировать по **scp**, а можешь с помощью текстового редактора создать новый файл и скопировать текст.

scp root@172.22.3.203:/var/lib/jenkins/.ssh/id\_[ed25519.pub](https://ed25519.pub/) ~/.ssh/authorized\_keys

chmod 0600 ~/.ssh/authorized\_keys

или

Отрываем на сервере authorized\_keys и копируем туда содержимое **id\_**[**ed25519.pub**](https://ed25519.pub/)

10:19

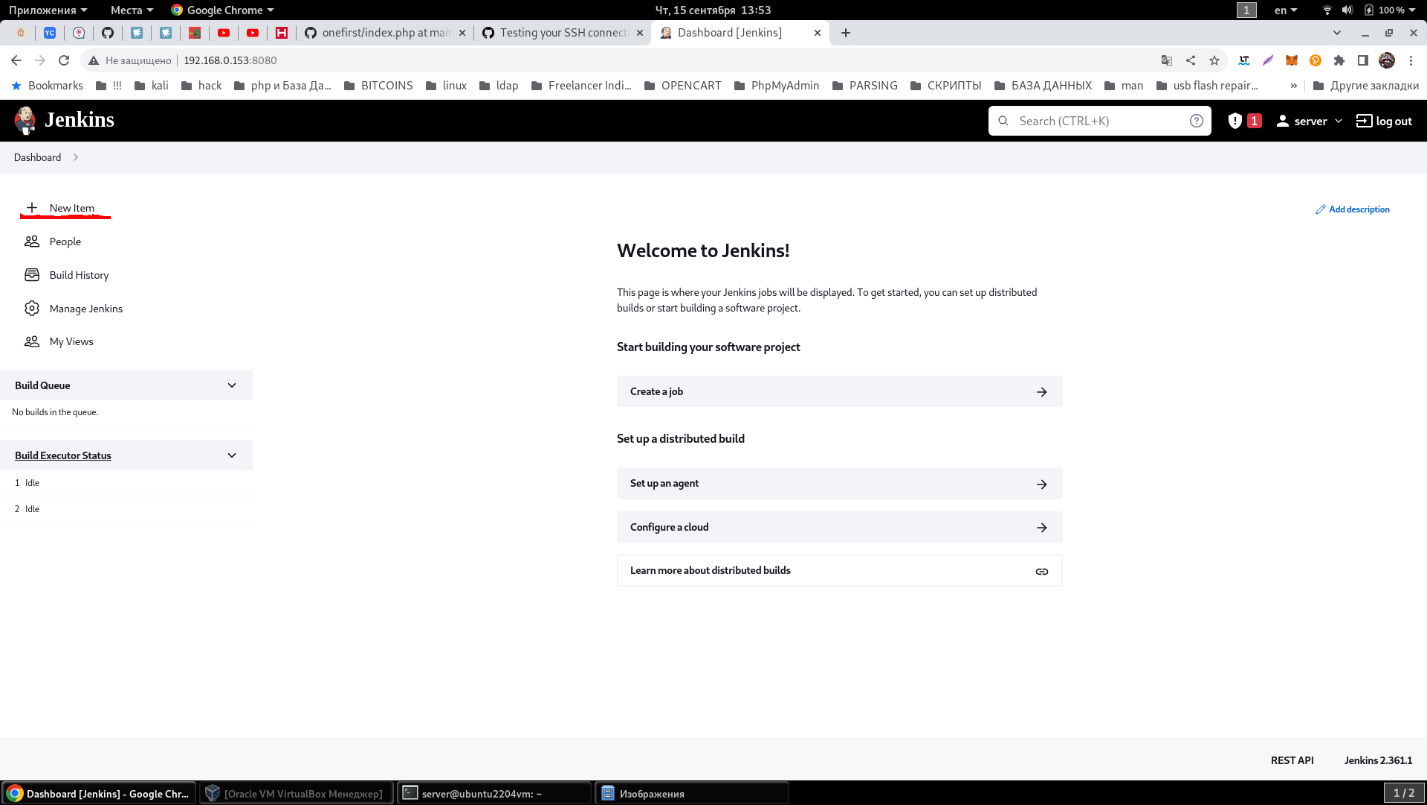
Обязательно убедись, что авторизация проходит. Чтобы дальше не отвлекаться на ненужные проблемы.

Дальше

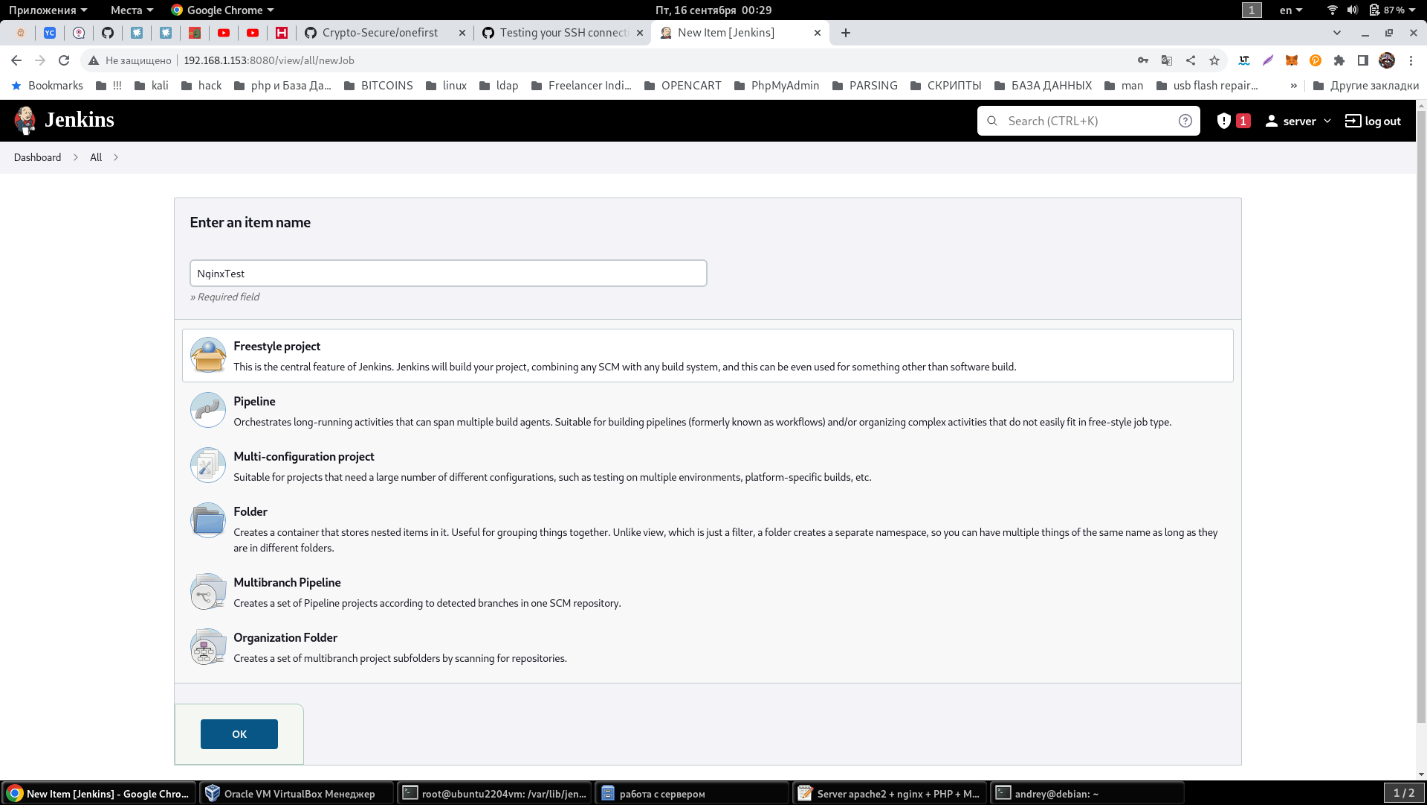
10:19

И вот пришло время вернуться в веб-интерейс **Jenkins**.

Жми кнопку "**Создать Item**"



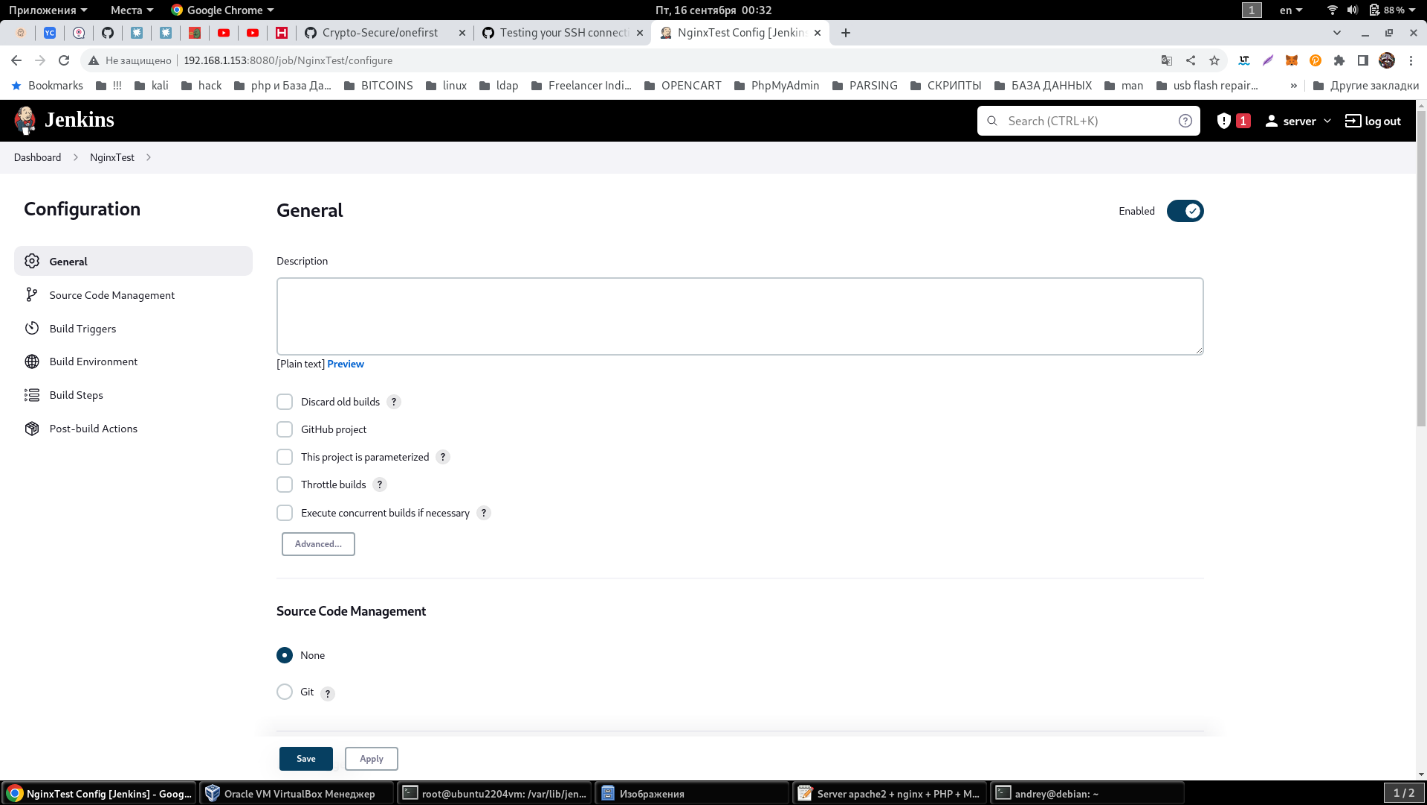
Создаем имя ( name ) и выбираем Создать задачу со свободной конфигурацией ( Freestyle project )



Появится форма создания задачи для дженкинса.

Тут можно просто прокручивать страницу вниз, а можно жать кнопки справа и перемещаться по кускам конфига.

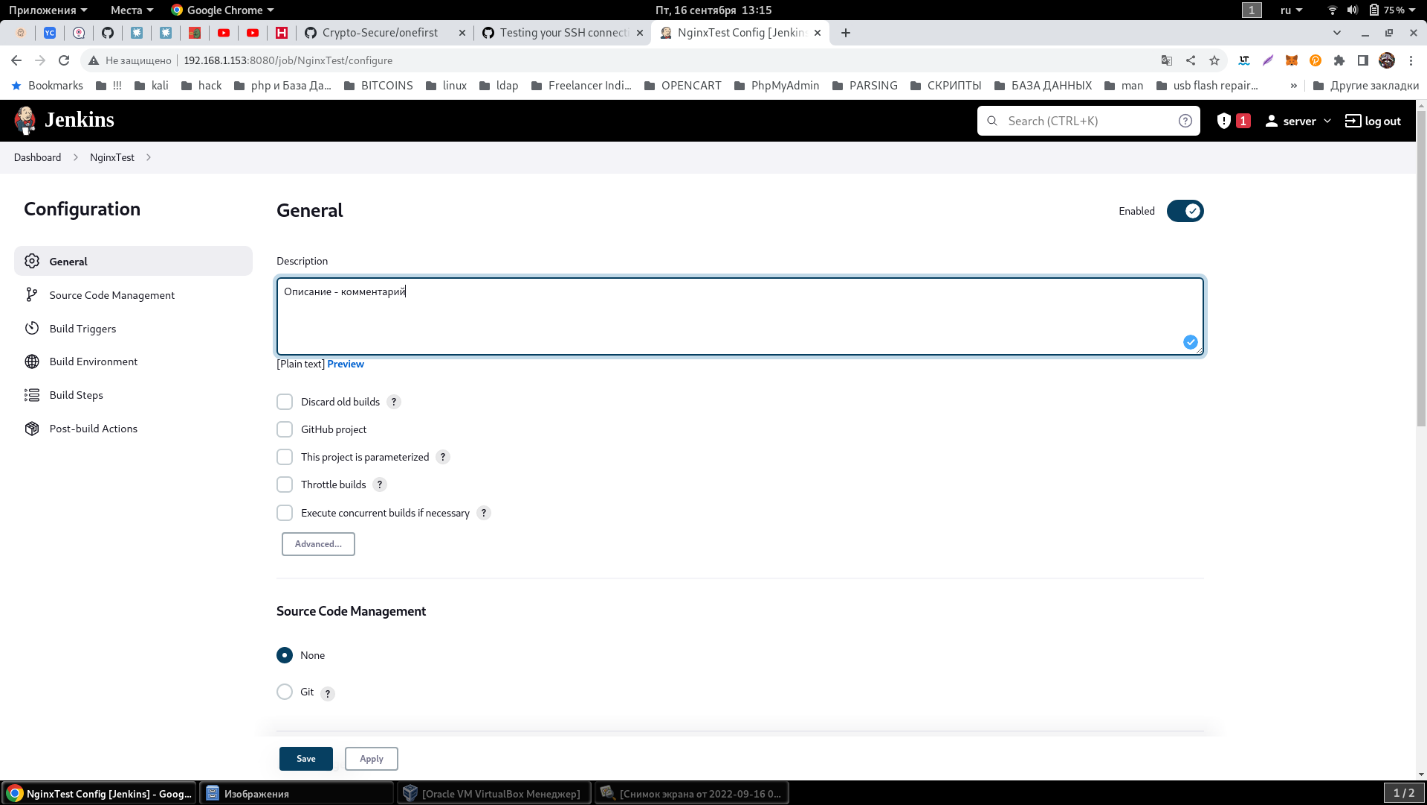
10:19

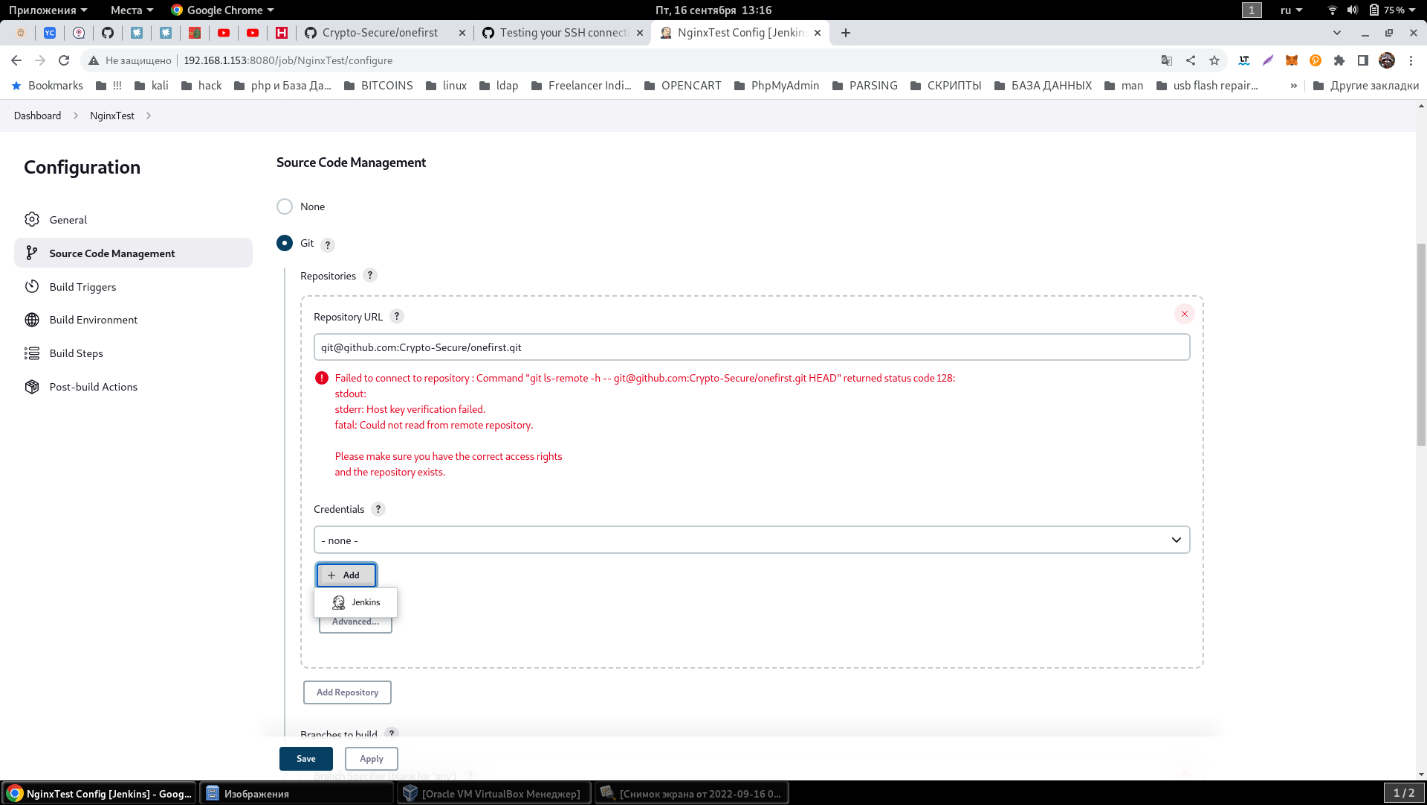


Дальше

10:19

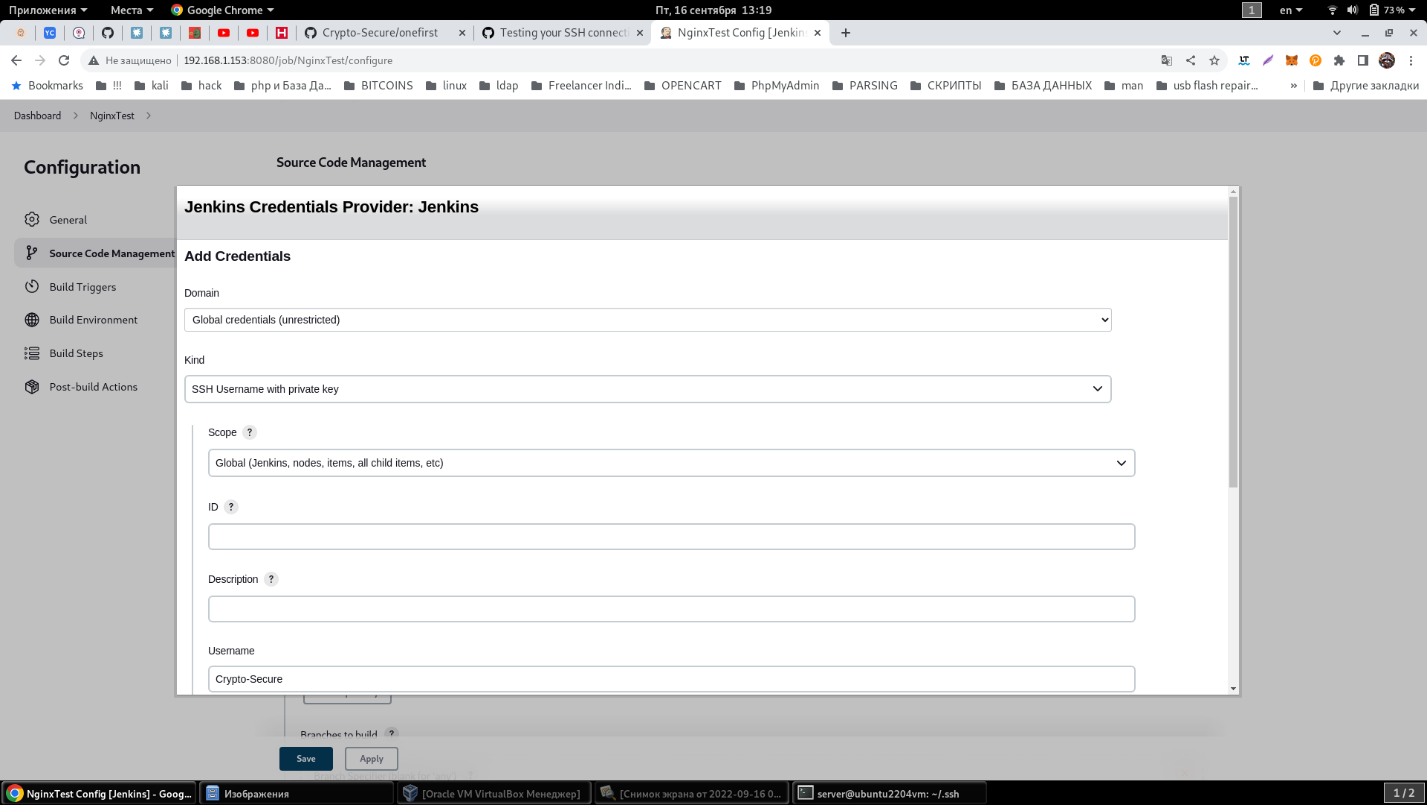
Начинаем заполнять.





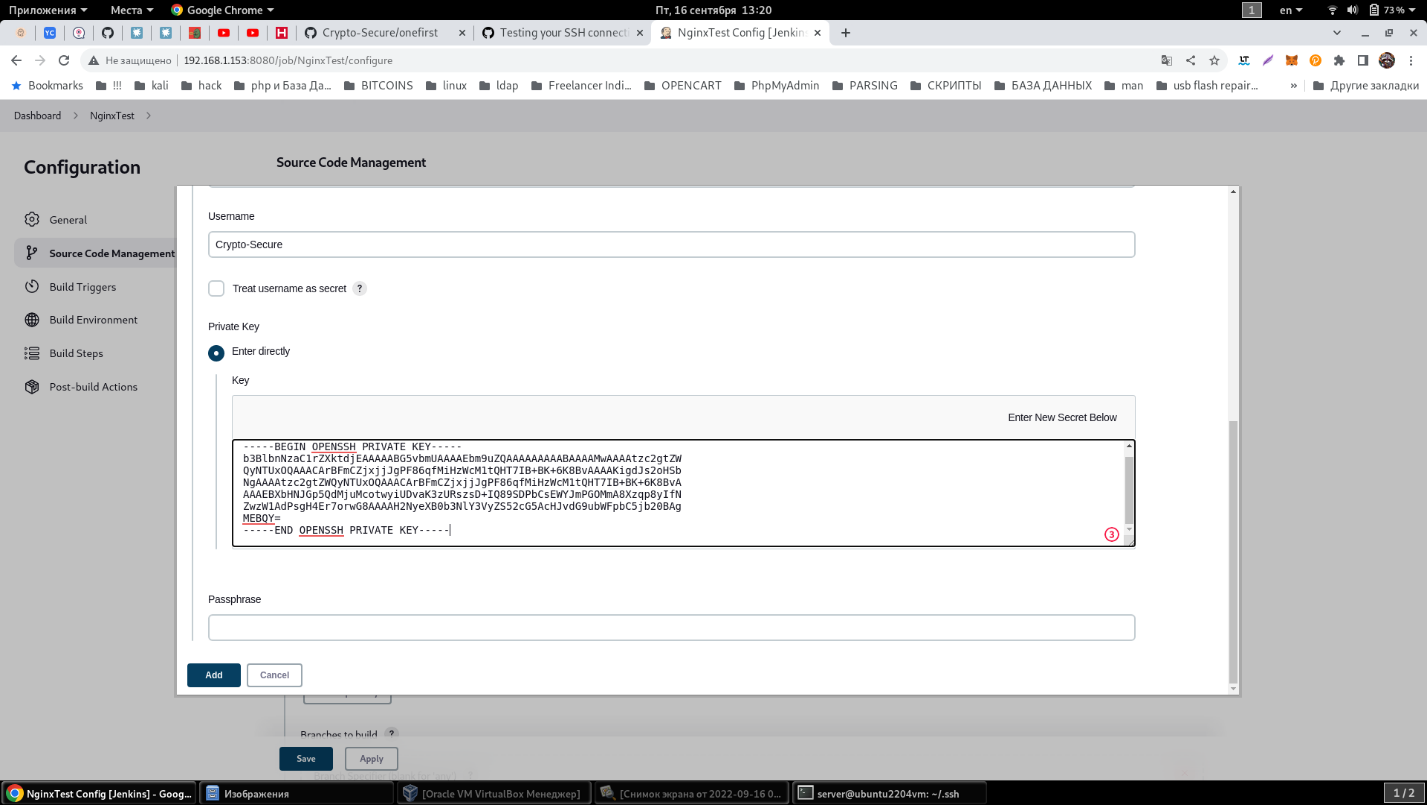
На этапе подключения репозитория надо будет указать учетные данные, с которыми ты в него входишь.

Жмешь **Add**, выбираешь пункт **Jenkins** (это внутренняя база **jenkins** для учетных данных) и заполняешь по аналогии со скриншотом ниже:



ключ вписываем от **id\_ed25519**

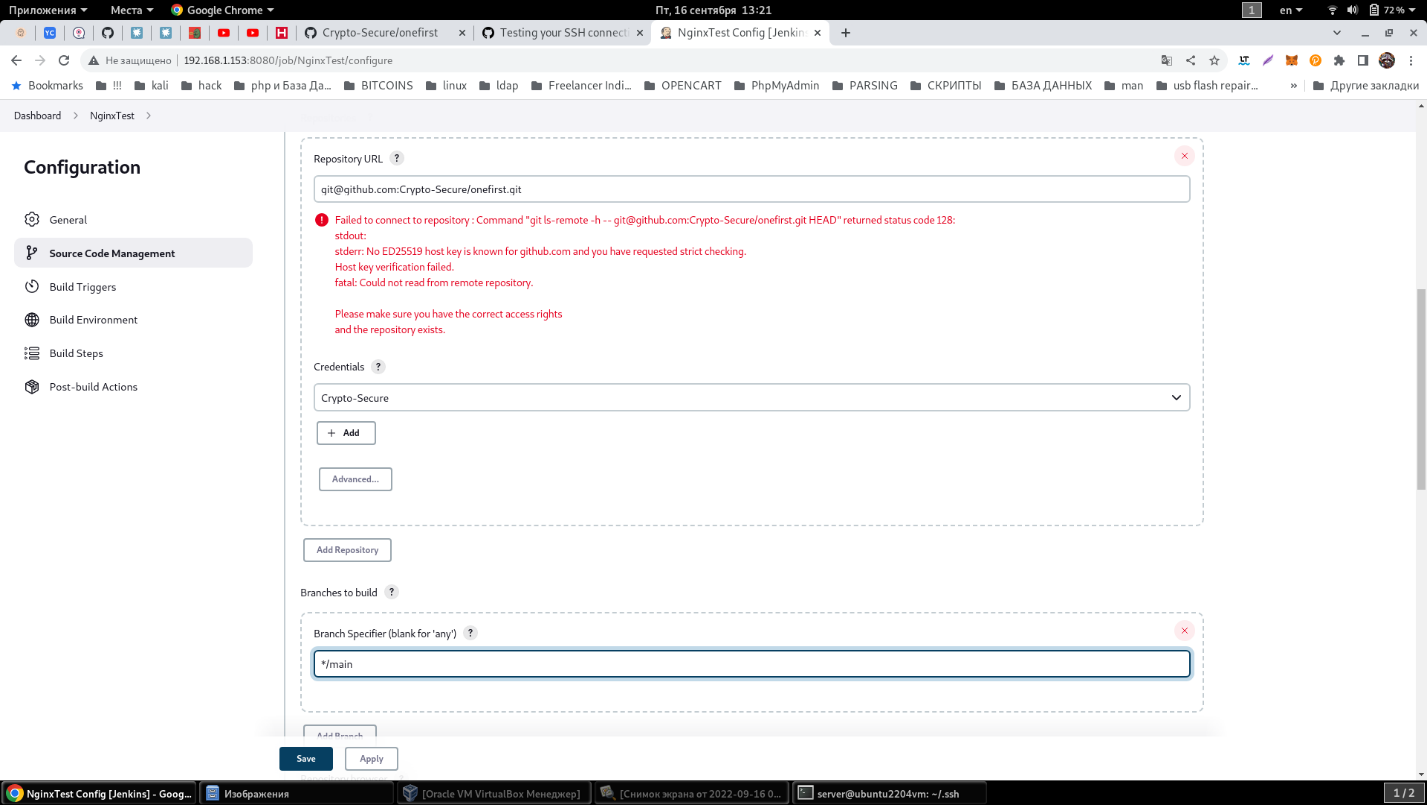
10:19



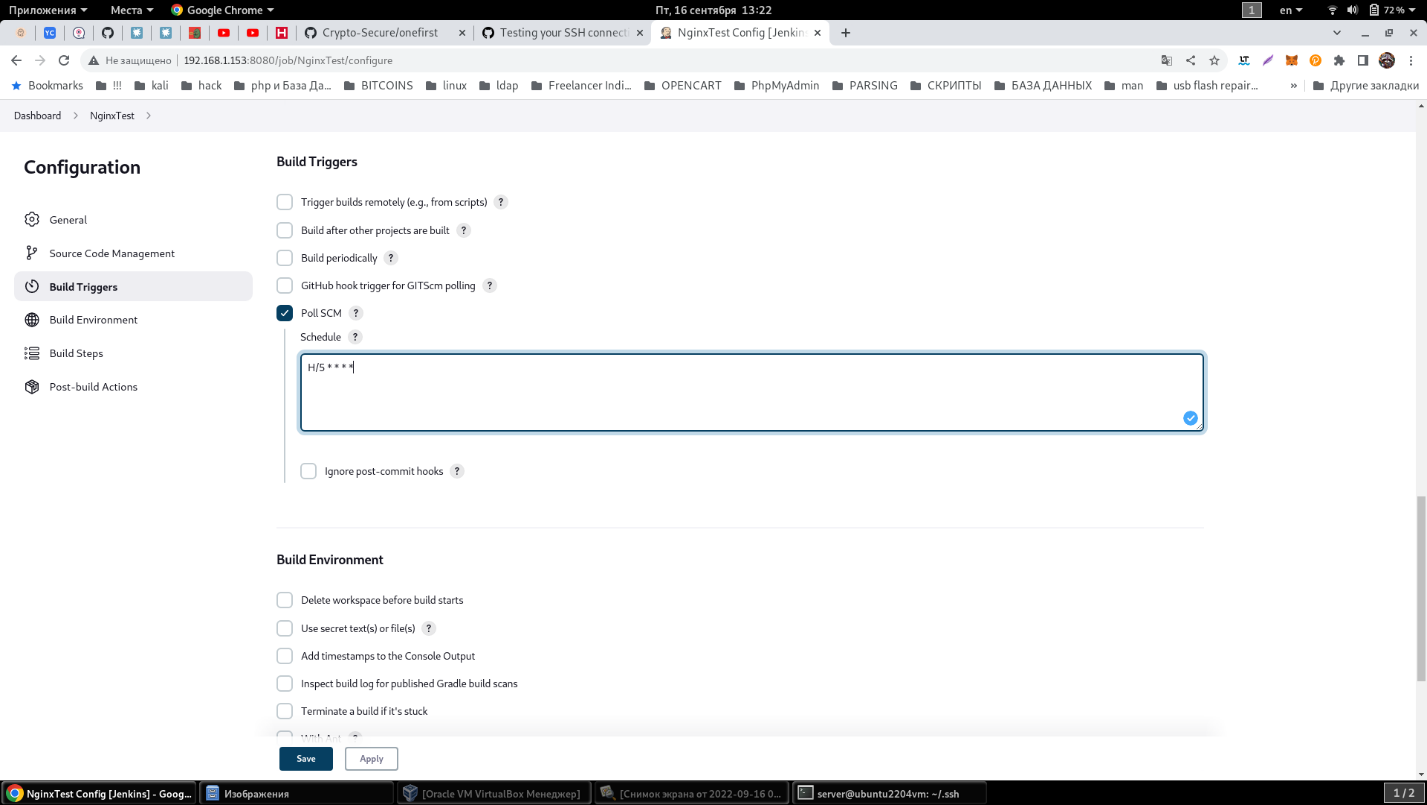
Дальше

10:19

Теперь можно выбрать эти учетные данные из выпадающего списка



Теперь определим условия, по которым проект будет автособираться.



10:19

Там был еще пункт "Запускать периодически". Но по нему проект будет без вопросов собираться каждый раз.

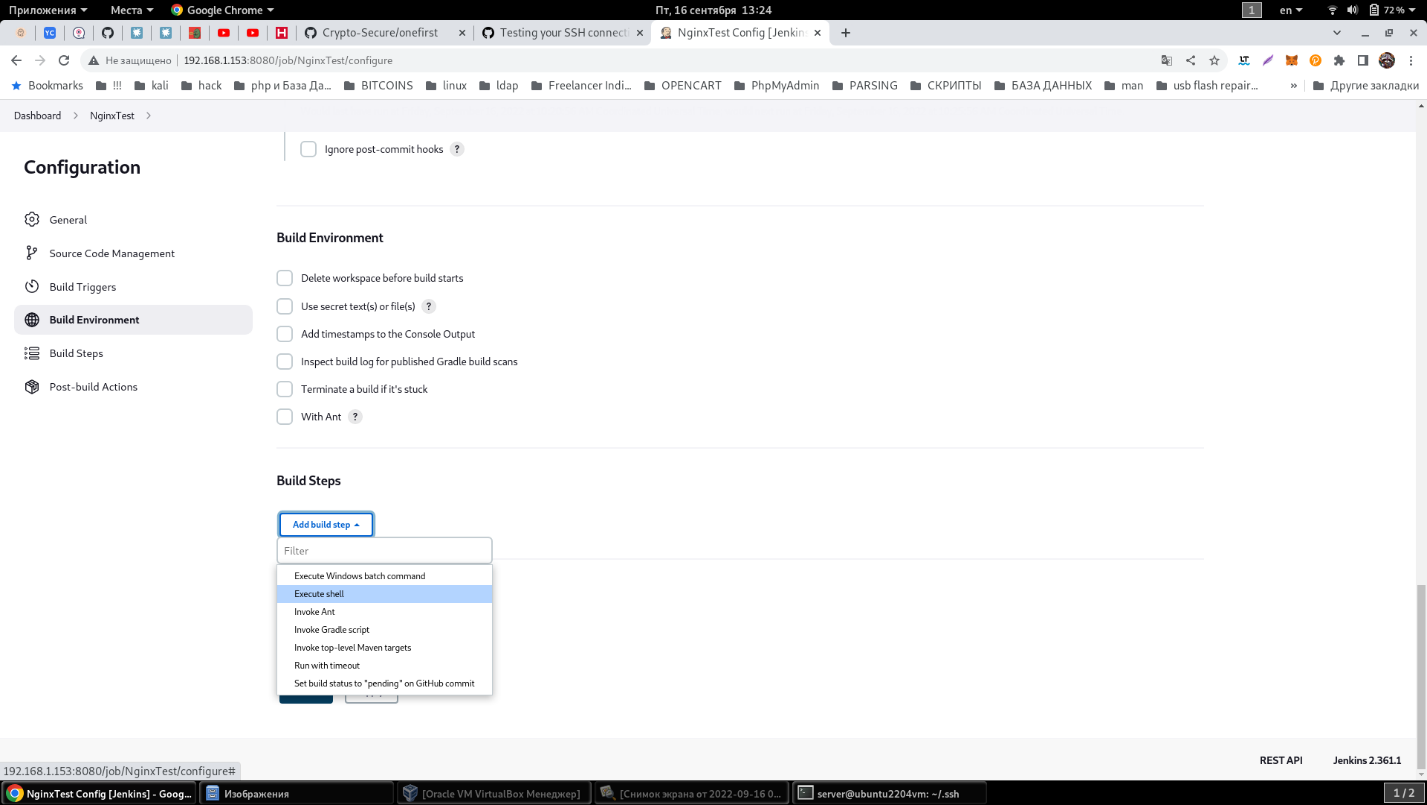
А **SCM** будет по расписанию опрашивать репозиторий. Но собирать проект только если есть изменения.

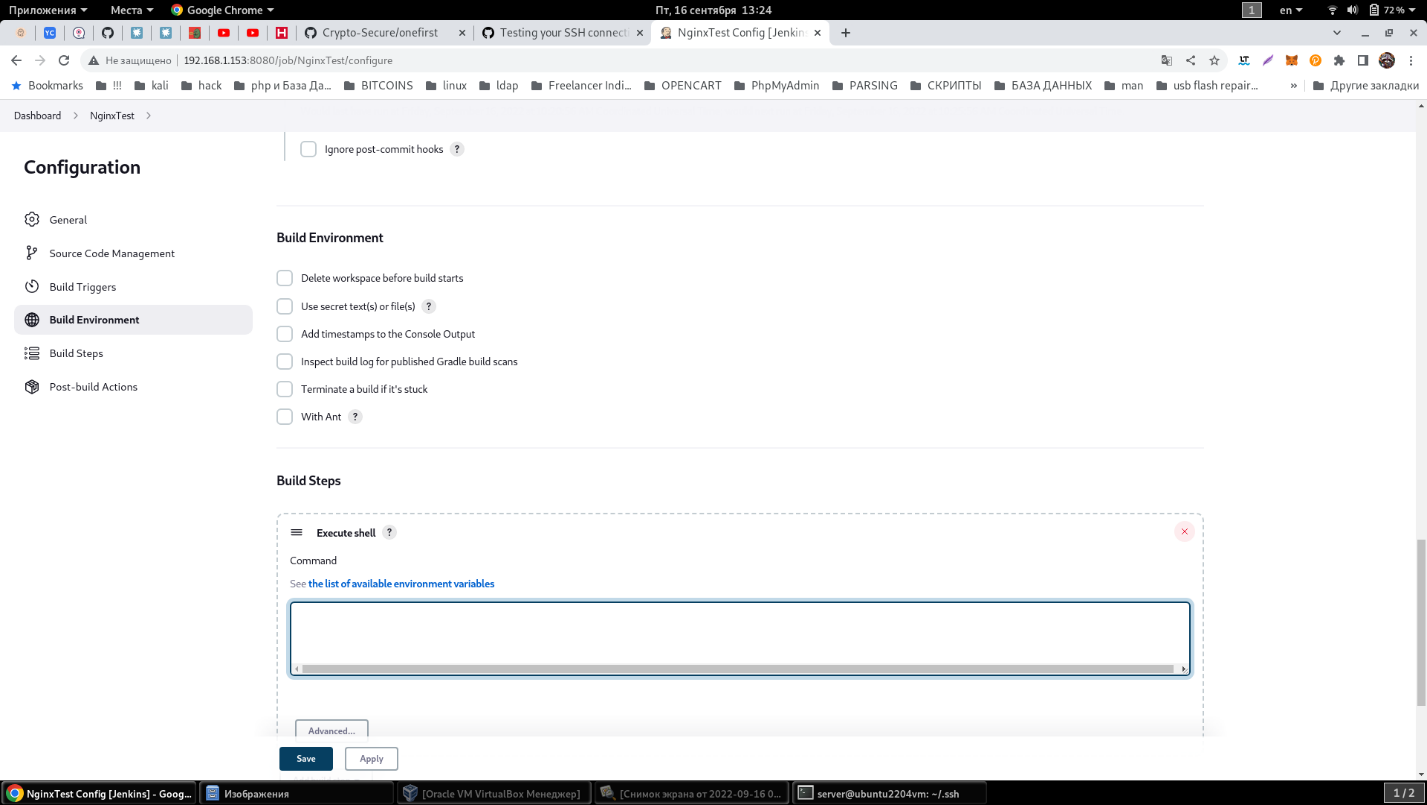
Дальше

10:19

И вот мы подошли к самому главному - сборке проекта.

Для начала надо выбрать метод сборки:





10:19

И вот в это окно можно уже писать **bash**-скрипт.

Дальше

10:19

Вот мой скрипт:

**. /etc/profile**

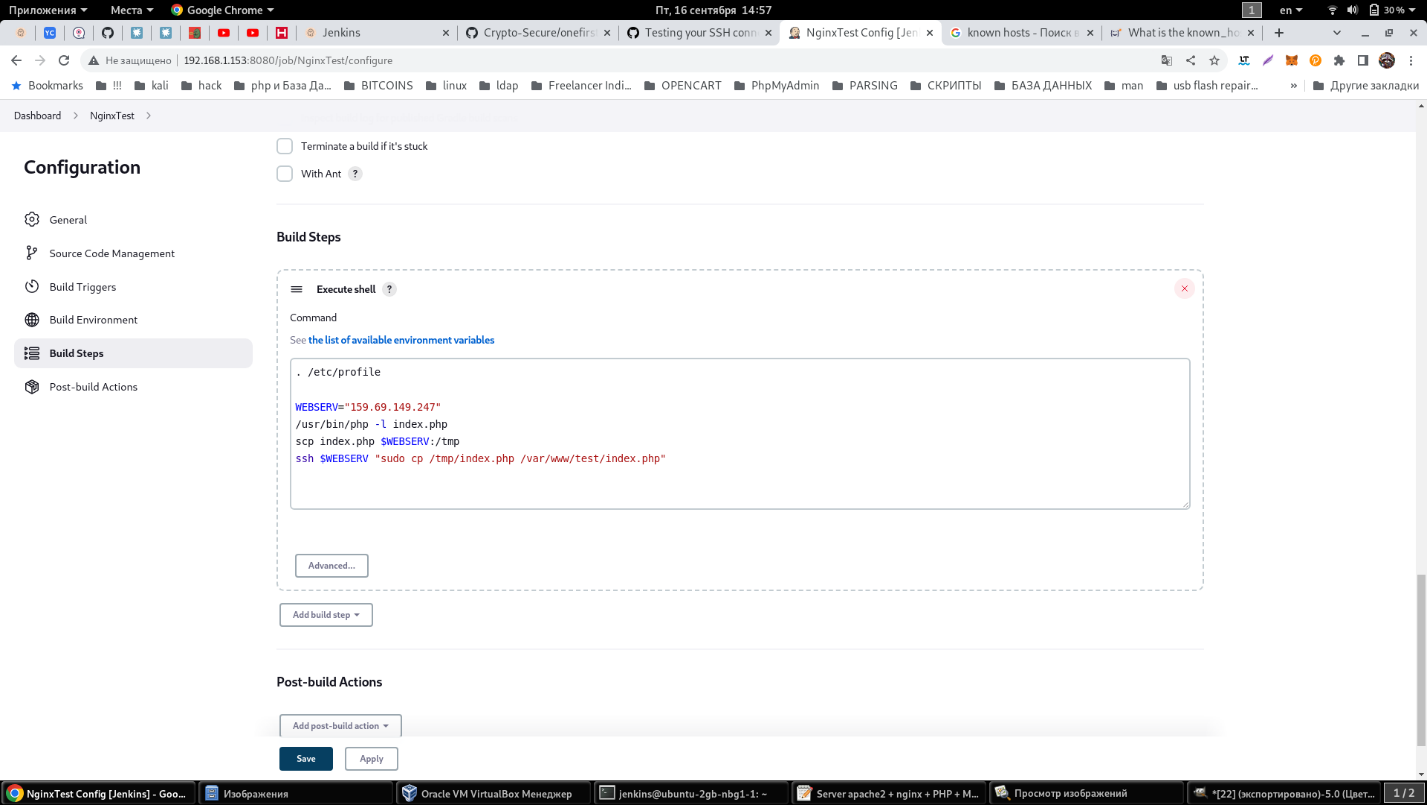
**WEBSERV="159.69.149.247"**

**/usr/bin/php -l index.php**

**scp index.php $WEBSERV:/tmp**

**ssh $WEBSERV "sudo cp /tmp/index.php /var/www/test/index.php"**

10:19



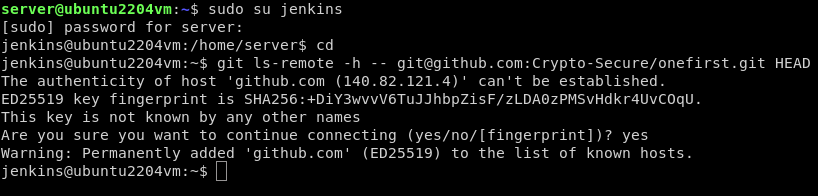
Дальше

10:19

Сохраняешь проект.

У нас есть еще предупреждение, которое светилось красным. Сейчас исправим.. Идем в консоль

git ls-remote -h -- [git@github.com](mailto:git@github.com):Crypto-Secure/onefirst.git HEAD



и введем еще одну команду

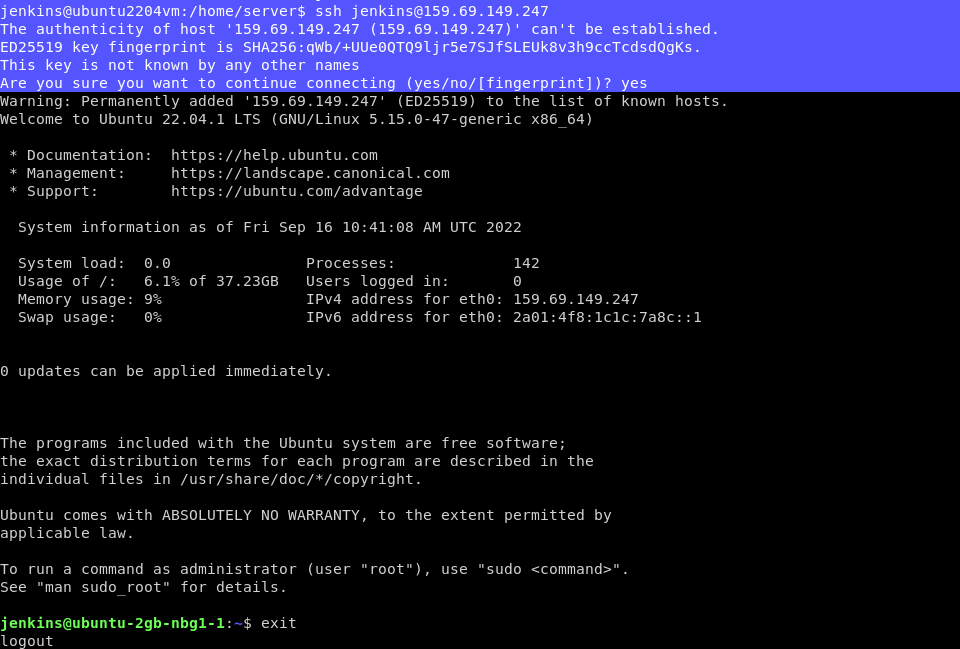
ssh jenkins@159.69.149.247

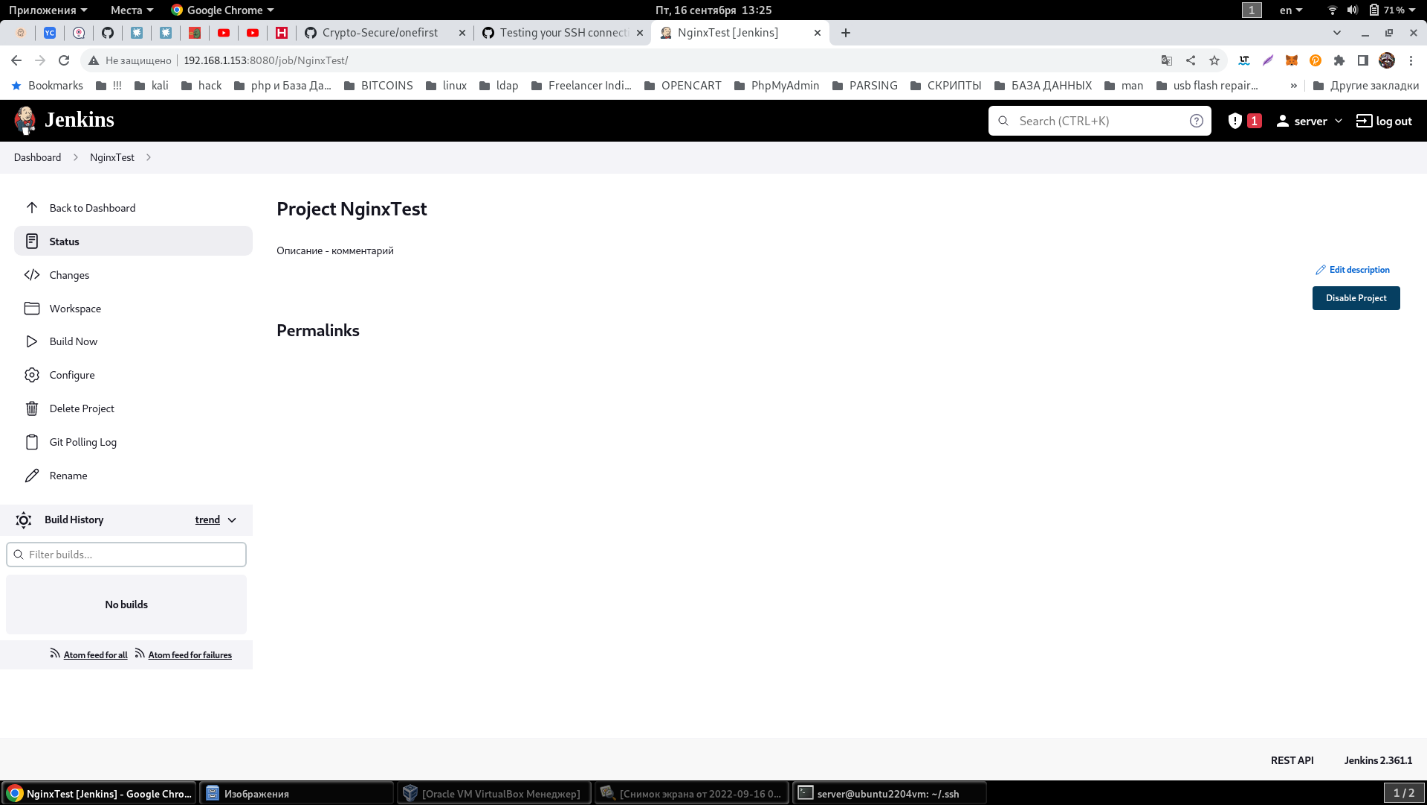
Это мы подключились к нашему серверу с nginx+php-fpm.

**это для того, чтоб прошла** аутентификация в файле know hosts.

Он перейдет в консоль сервера **jenkins**. (смотрите скрин)

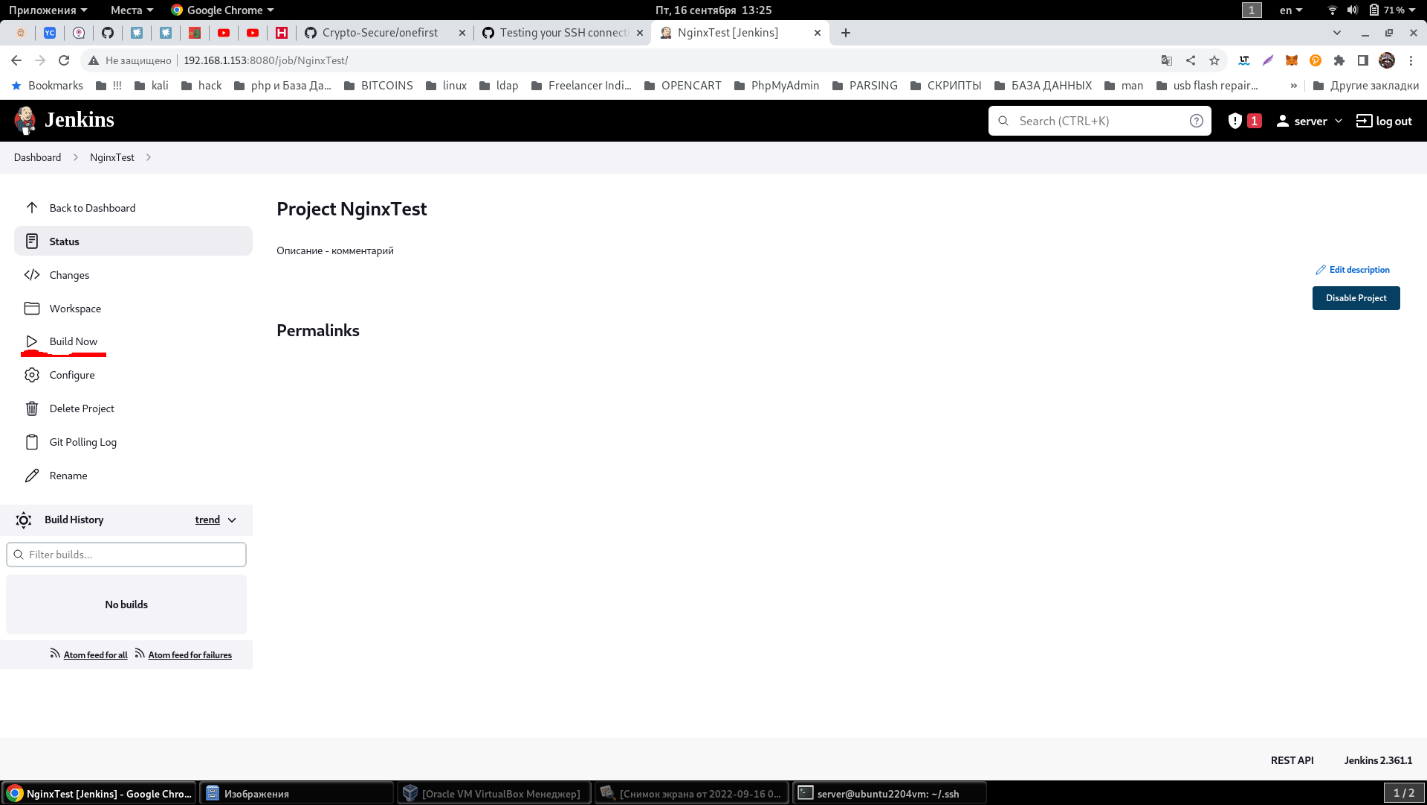
Наберите **exit, чтоб вернутся в нашу консоль**





Там есть кнопочка "**Собрать проект**". Жми ее, чтобы собрать проект принудительно - тут ты решаешь, а не **SCM**.

10:19



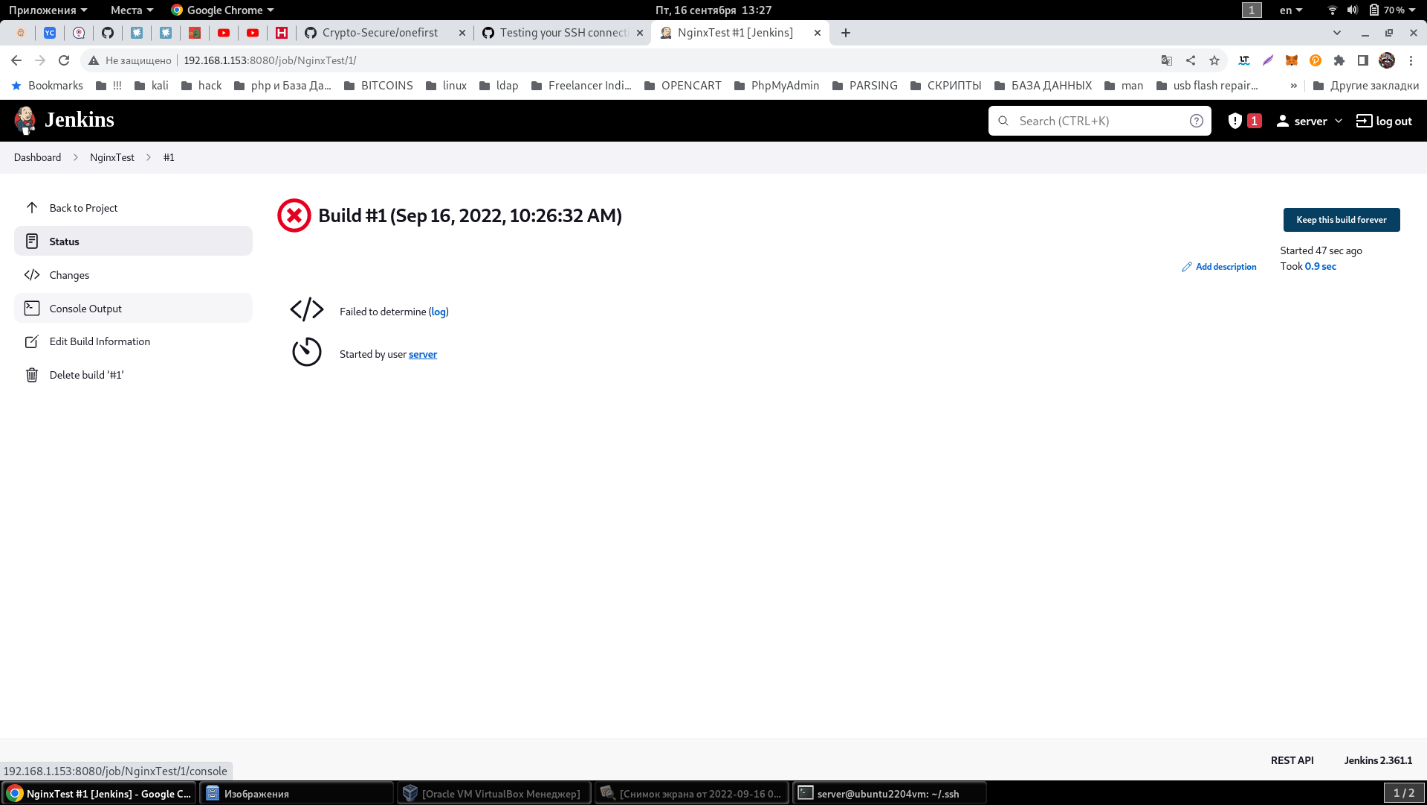
Дальше

10:19

Как ты заметил на предыдущем шаге - значок сборки красного цвета. Это означает одно - сборка завершилась с ошибкой.

Чтобы узнать по какой причине сборка не завершилась - жми прямо на этот красный кружок.

Увидишь вот такое окно:



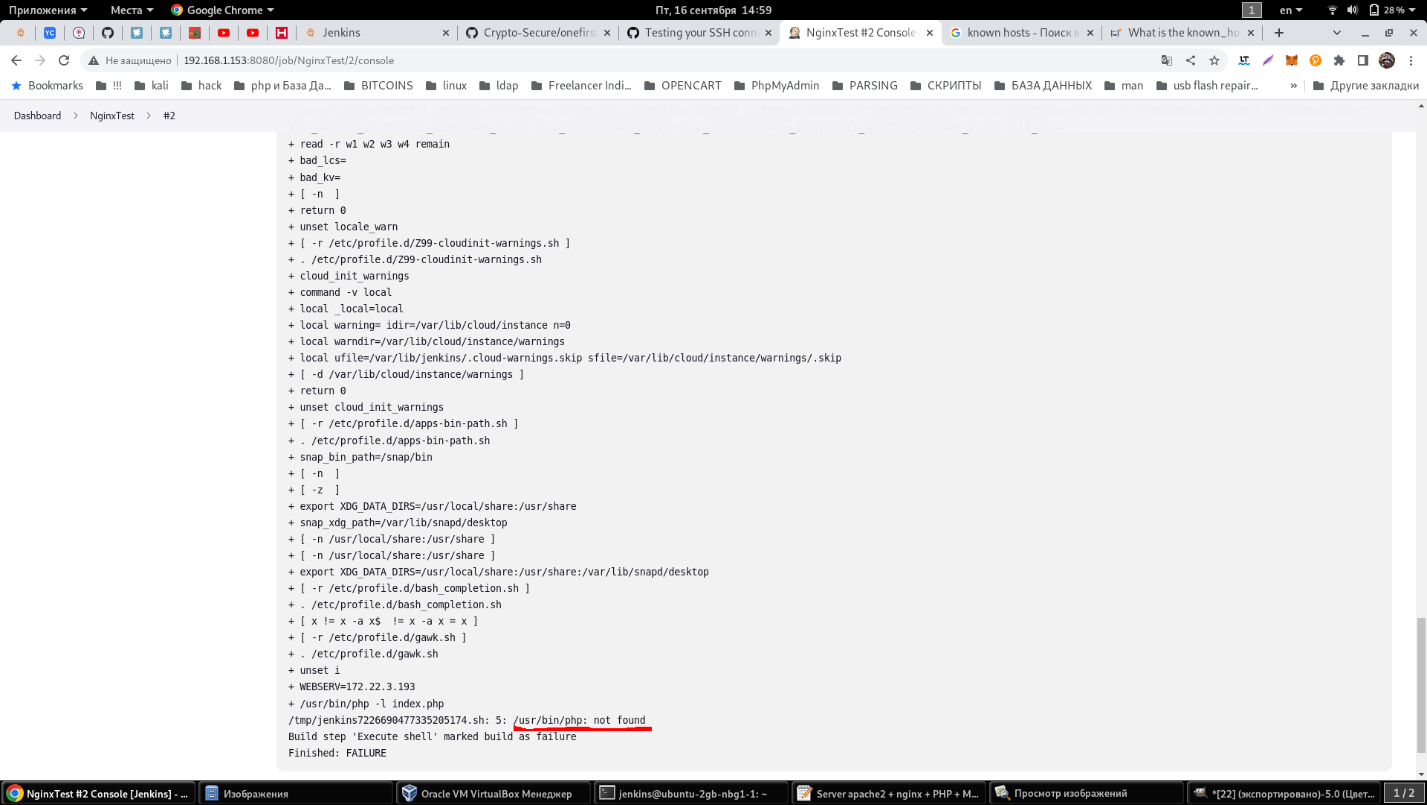
10:20

И теперь жми "**Вывод консоли**".

Дальше

10:20

Возможно когда-то тебе потребуется весь вывод. Но сейчас листай в самый низ, чтобы найти сообщение об ошибке, на которой падает сборка.



10:20

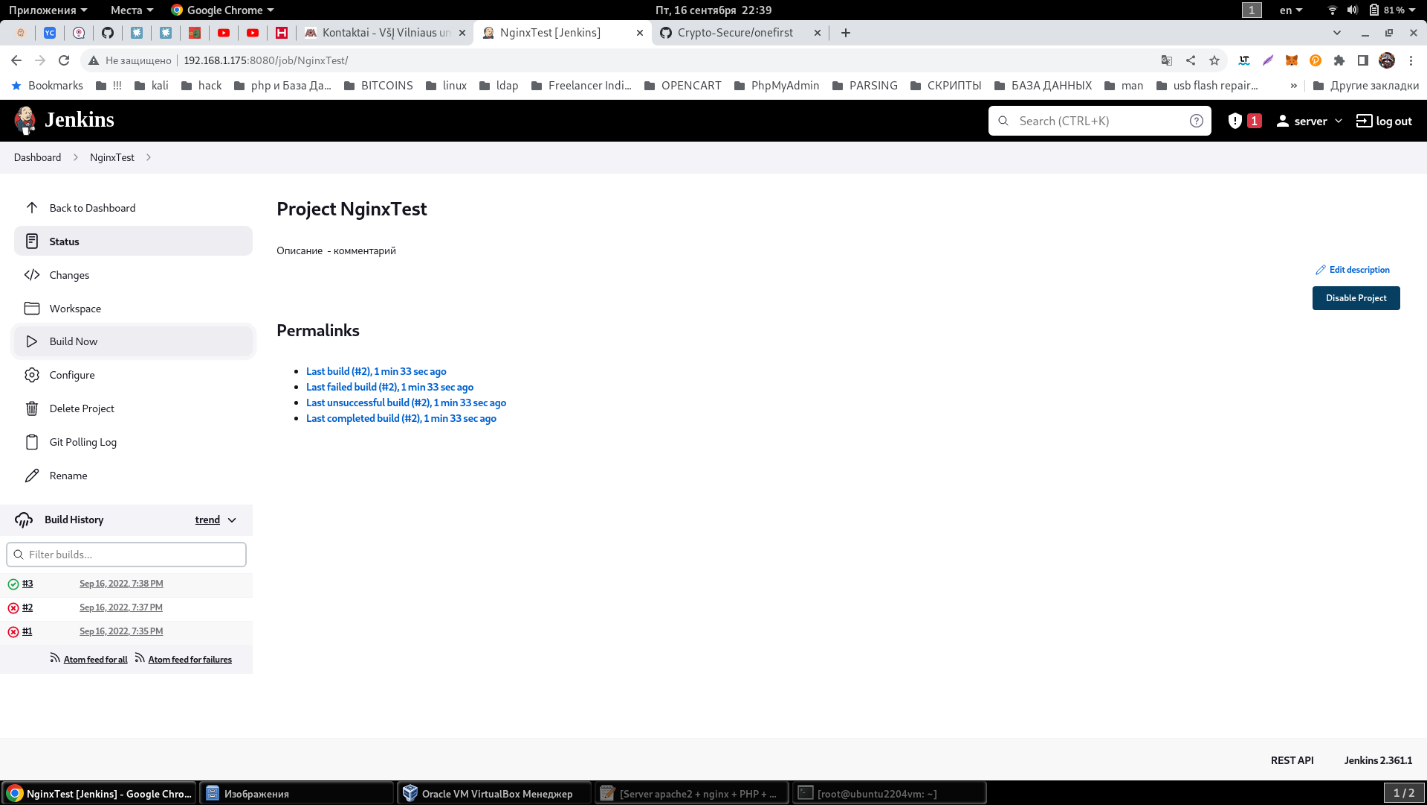
Хм... а и правда. Jenkins то мы поставили, а вот **php** на сервере нет. Ошибка вполне логична.

Доустанови пакет php на сервер с **Jenkins** и попробуй собрать проект снова.

Дальше

10:20

Ну вот. Совсем другое дело. Зеленый кружок означает, что сборка прошла успешно.



В твое отстуствие всем этим будет заведовать SCM =).

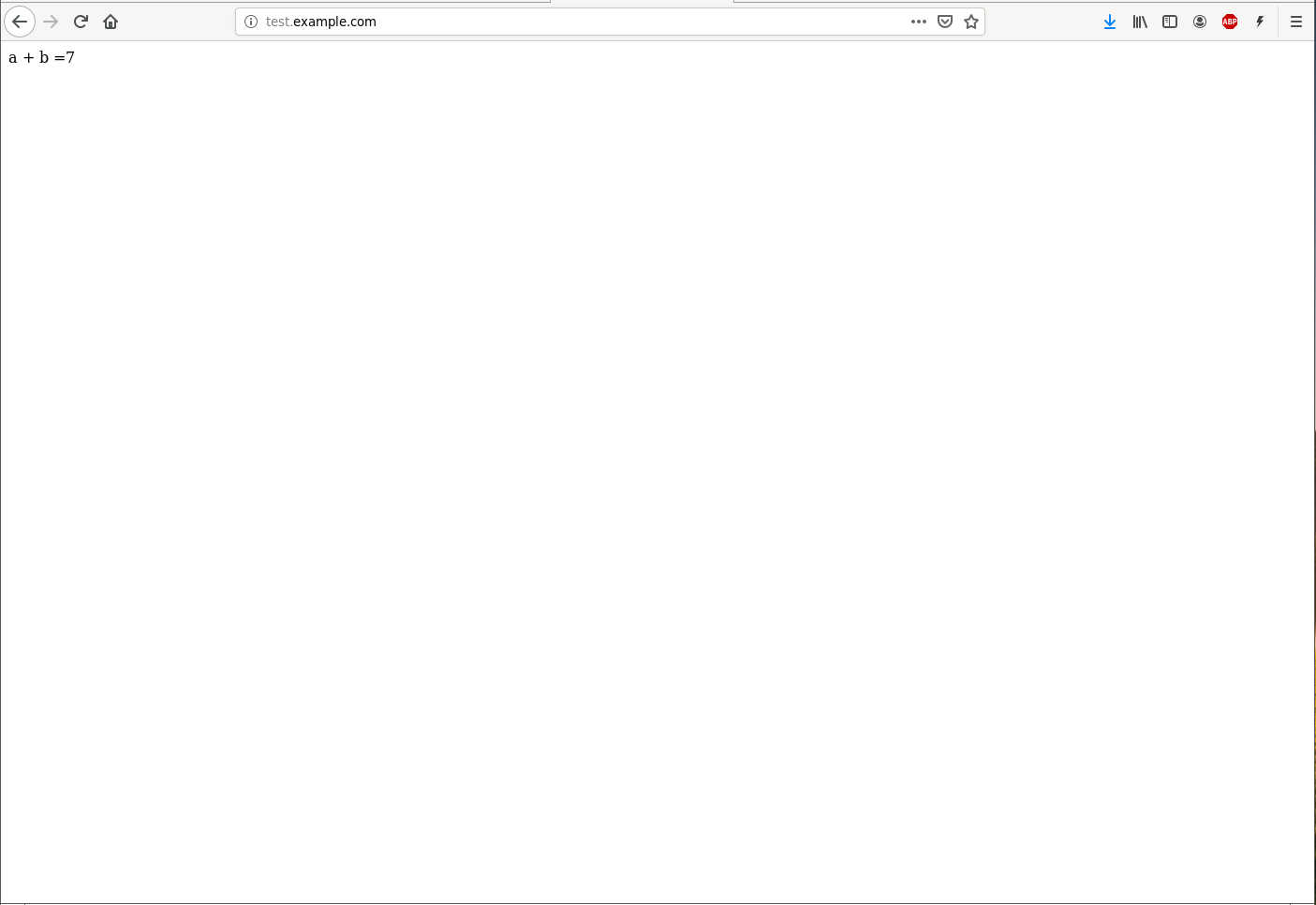
10:20

Кстати, там же в настройках проекта, можно настроить отсылку уведомлений о неудачных сборках - это для того, чтобы быть в курсе без захода в панель **Jenkins**.

Дальше

10:20

Если ты зайдешь на веб-сервер, то увидишь вот это:



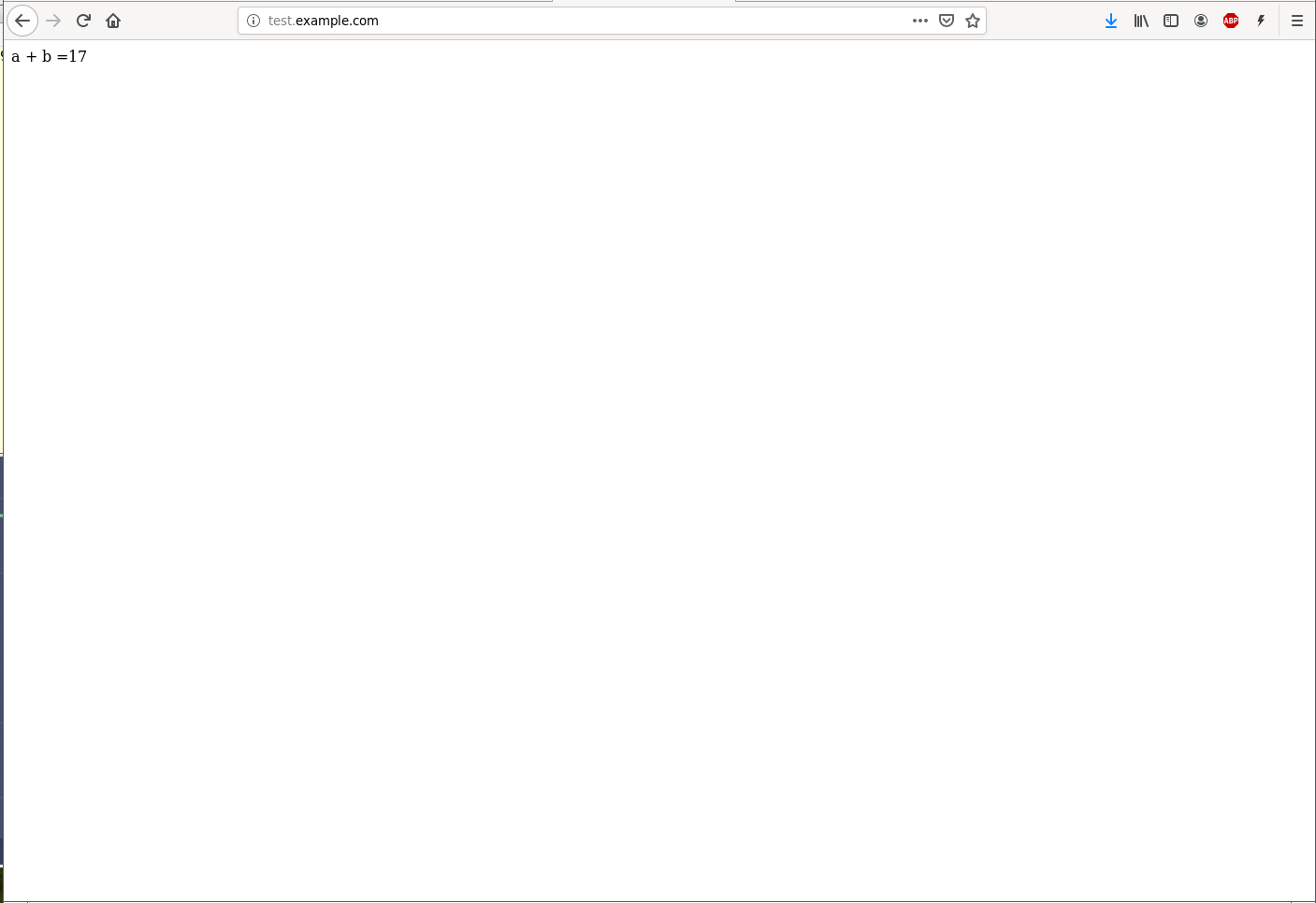
10:20

А теперь давай поменяем значения переменных в **index.php**, загрузим его в репозиторий и отвлечемся на чашку чая (и съешь еще этих мягких французских булок =))))) ). Пусть Jenkins поработает самостоятельно =).

Ок. Вернусь после чашечки чая.

10:20

Итак. Пришло время обновить страницу. И циферка изменилась - т.е. jenkins увидел изменения, протестил файл на орфографические ошибки и залил на веб-сервер.



10:20

Ну вот и все.

Хоть такое использование дженкинса и игнорирует 99% его возможностей, но при этом дает тебе возможность ввести **CI/CD** в рабочий процесс и тем самым освободить часть времени для самостоятельного изучения остальных возможностей дженкинса.