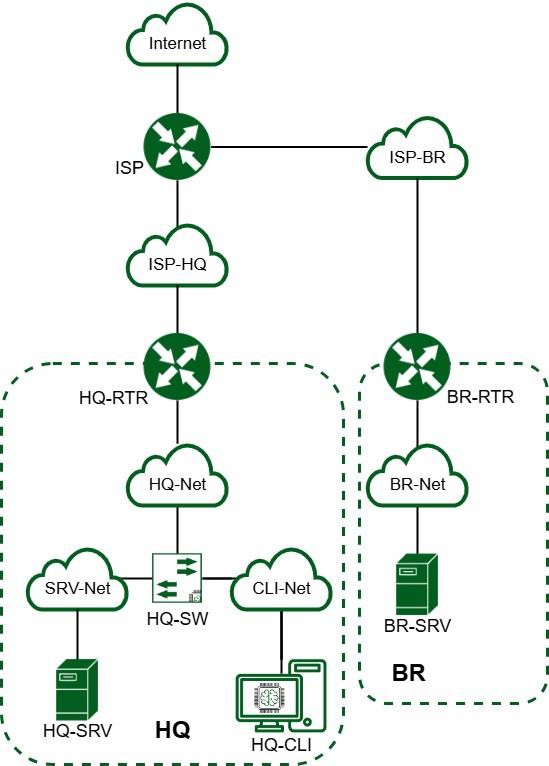
Разбор ДЭ 2025

**Сетевое и системное администрирование Модуль 2**



**Задание**

## Настройте доменный контроллер Samba на машине BR-SRV.

* + Создайте 5 пользователей для офиса HQ: имена пользователей формата user№.hq. Создайте группу hq, введите в эту группу созданных пользователей
  + Введите в домен машину HQ-CLI
  + Пользователи группы hq имеют право аутентифицироваться на клиентском ПК
  + Пользователи группы hq должны иметь возможность повышать привилегии для выполнения ограниченного набора команд: cat, grep, id. Запускать другие команды с повышенными привилегиями пользователи группы не имеют права
  + Выполните импорт пользователей из файла users.csv. Файл будет располагаться на вирту- альной машине BR-SRV в папке /opt

## Сконфигурируйте файловое хранилище:

* + При помощи трёх дополнительных дисков, размером 1Гб каждый, на HQ-SRV сконфигури- руйте дисковый массив уровня 5
  + Имя устройства – md0, конфигурация массива размещается в файле /etc/mdadm.conf
  + Обеспечьте автоматическое монтирование в папку /raid5
  + Создайте раздел, отформатируйте раздел, в качестве файловой системы используйте ext4
  + Настройте сервер сетевой файловой системы(nfs), в качестве папки общего доступа выбе- рите /raid5/nfs, доступ для чтения и записи для всей сети в сторону HQ-CLI
  + На HQ-CLI настройте автомонтирование в папку /mnt/nfs
  + Основные параметры сервера отметьте в отчёте

## Настройте службу сетевого времени на базе сервиса chrony

* + В качестве сервера выступает HQ-RTR
  + На HQ-RTR настройте сервер chrony, выберите стратум 5
  + В качестве клиентов настройте HQ-SRV, HQ-CLI, BR-RTR, BR-SRV

## Сконфигурируйте ansible на сервере BR-SRV

* + Сформируйте файл инвентаря, в инвентарь должны входить HQ- SRV, HQ-CLI, HQ-RTR и BR-RTR
  + Рабочий каталог ansible должен располагаться в /etc/ansible
  + Все указанные машины должны без предупреждений и ошибок отвечать pong на команду ping в ansible посланную с BR-SRV

## Развертывание приложений в Docker на сервере BR-SRV.

* + Создайте в домашней директории пользователя файл wiki.yml для приложения MediaWiki.
  + Средствами docker compose должен создаваться стек контейнеров с приложением MediaWiki и базой данных.
  + Используйте два сервиса
  + Основной контейнер MediaWiki должен называться wiki и использовать образ mediawiki
  + Файл LocalSettings.php с корректными настройками должен находиться в домашней папке пользователя и автоматически монтироваться в образ.
  + Контейнер с базой данных должен называться mariadb и использовать образ mariadb.
  + Разверните
  + Он должен создавать базу с названием mediawiki, доступную по стандартному порту, поль- зователя wiki с паролем WikiP@ssw0rd должен иметь права доступа к этой базе данных
  + MediaWiki должна быть доступна извне через порт 8080.

## На маршрутизаторах сконфигурируйте статическую трансляцию портов

* + Пробросьте порт 80 в порт 8080 на BR-SRV на маршрутизаторе BR- RTR, для обеспечения работы сервиса wiki
  + Пробросьте порт 2024 в порт 2024 на HQ-SRV на маршрутизаторе HQ-RTR
  + Пробросьте порт 2024 в порт 2024 на BR-SRV на маршрутизаторе BR-RTR

## Запустите сервис moodle на сервере HQ-SRV:

* + Используйте веб-сервер apache
  + В качестве системы управления базами данных используйте mariadb • Создайте базу данных moodledb
  + Создайте пользователя moodle с паролем P@ssw0rd и предоставьте ему права доступа к этой базе данных
  + У пользователя admin в системе обучения задайте пароль P@ssw0rd
  + На главной странице должен отражаться номер рабочего места в виде арабской цифры, дру- гих подписей делать не надо
  + Основные параметры отметьте в отчёте

## Настройте веб-сервер nginx как обратный прокси-сервер на HQ-RTR

* + При обращении к HQ-RTR по доменному имени moodle.au-team.irpo клиента должно пере- направлять на HQ-SRV на стандартный порт, на сервис moodle
  + При обращении к HQ-RTR по доменному имени wiki. au-team.irpo клиента должно перена- правлять на BR-SRV на порт, на сервис mediwiki

## Удобным способом установите приложение Яндекс Браузере для организаций на HQ-CLI

* + Установку браузера отметьте в отчёте

***Решение:***

1. **​НАСТРОЙКА ДОМЕННОГО КОНТРОЛЛЕРА SAMBA НА МАШИНЕ BR-SRV**

**BR-SRV:**

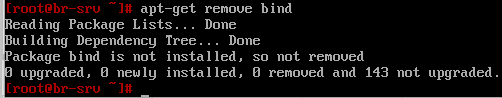
Перед настройкой самого контроллера домена проверим наличие пакета bind на нашем сервере, для этого попробуем его удалить командой:

## apt-get remove bind



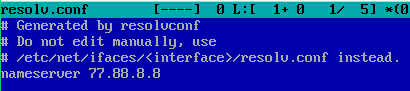
Делаем, чтобы не было конфликтов базового bind и нашего нового доменного.

Если команда выводит, что нет пакета для удаления, значит все хорошо и можем приступать дальше



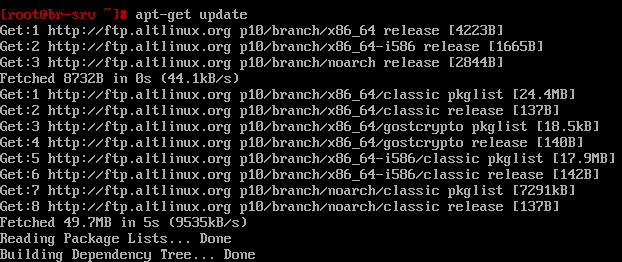
Проверим наш файл /etc/resolv.conf и наличие в нем строки с удаленным сервером DNS nameserver 77.88.8.8

## mcedit /etc/resolv.conf



Теперь установим службу samba на наш сервер BR-SRV

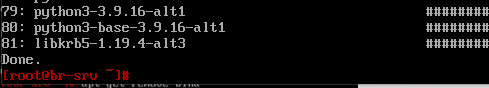
## apt-get update



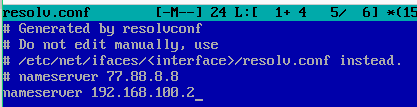
**apt-get install task-samba-dc**



Подтверждаем установку и дожидаемся окончания операции



Теперь файл **/etc/resolv.conf** приведем к виду **nameserver 192.168.100.2**, закомментировав dns- сервер яндекса (поставить знак # в начале строки) и дописав в него локальный dns-сервер их подсети HQ, который работает у нас на сервере HQ-SRV.



Это нужно сделать обязательно, т.к. при установке **Samba** как доменного контроллера, будет ещё использоваться DNS-сервер. а у нас уже есть DNS-сервер, и это – **HQ-SRV**, вот он и будет использоваться для разрешения доменных имен.

Теперь удалим конфиг smb.conf, чтобы он не мешал при настройке службы:

## rm -rf /etc/samba/smb.conf



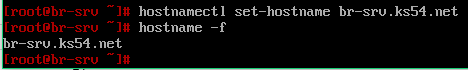
Проверяем, что установлено полное доменное имя у **BR-SRV**:

## hostname -f



Запись не соответствует таблице настройки доменных имен устройств, поэтому нужно его настроить в соответствии с заданием (исправить):

## hostnamectl set-hostname br-srv.ks54.net; exec bash



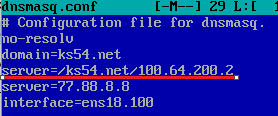
В противном случае идем дальше.

Настроим **hosts**, добавив новую запись в конец файла:

## mcedit /etc/hosts 100.64.200.2 br-srv.ks54.net

Теперь в конфигурацию нашего DNS-сервера на **HQ-SRV** добавим следующую строку:

## server=/ks54.net/100.64.200.2



Перезапускаем **dnsmasq** как службу:

## systemctl restart dnsmasq



А теперь запускаем автонастройку доменного контроллера на **BR-SRV**. Если предложенные значения верны, те, что находятся в [], то нажимаем Enter, если нет, то нужно проверять предыдущие настройки.

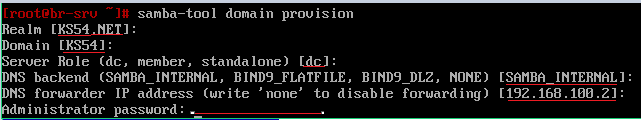
**samba-tool domain provision**



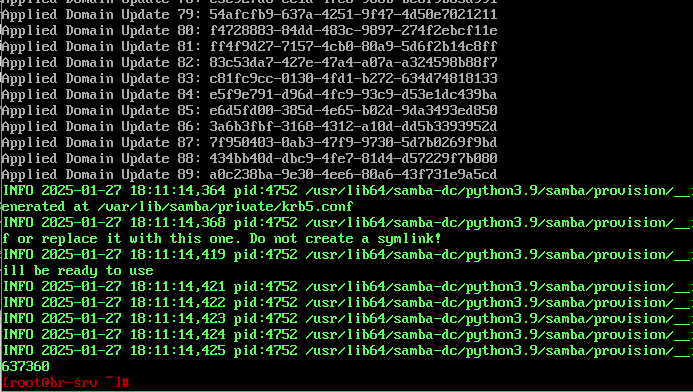
**KS54.NET KS54**

**dc SAMBA\_INTERNAL 192.168.100.2**

**DCP@ssw0rd** (Здесь вводим значение вручную)



После настройки должно появится такое в терминале, это значит, что всё настроено верно:



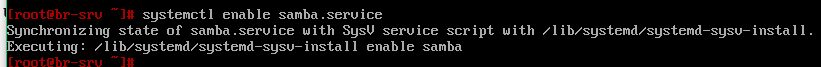
Перемещаем сгенерированный конфиг krb5.conf

## mv -f /var/lib/samba/private/krb5.conf /etc/krb5.conf



и включаем службу samba:

## systemctl enable samba.service



Из-за того, что на **Alt Linux** могут пропадать IP-адреса после перезагрузки системы, добавим запись о перезапуске службы **network** и **samba** в **crontab** (именно в таком порядке), пишем в консоль:

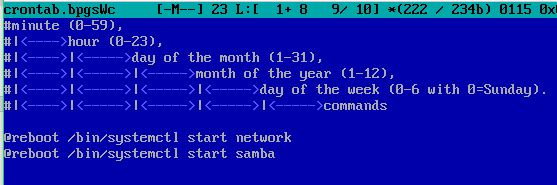
**export EDITOR=mcedit** (создаем локальную переменную запуска текстового редактора)

**сrontab -e** (запускаем crontab с помощью нашей переменной и редактора в нем) И вносим в конец файла следующие строки:

## @reboot /bin/systemctl start network

**@reboot /bin/systemctl start samba**

**!ВАЖНО!** Оставляем ещё одну пустую строку снизу, иначе файл не станет сохраняться!

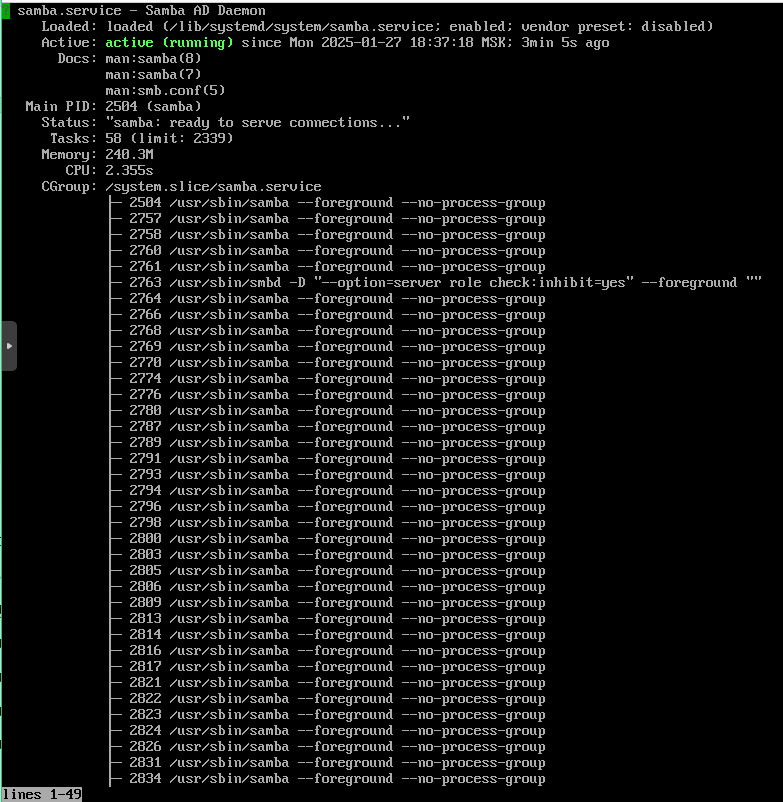


Теперь **ПЕРЕЗАПУСКАЕМ** машину **BR-SRV**:

## reboot



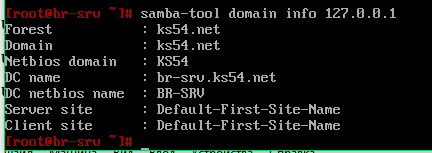
Логинимся и проверяем работу сервиса samba командой systemctl status samba



Если все выдает без ошибок, то тогда уже проверяем работу домена:

## samba-tool domain info 127.0.0.1





Домен работает, у вас должно всё соответствовать картинке выше.

**Теперь создадим 5 пользователей:**

**samba-tool user add user1.hq P@ssw0rd**



**samba-tool user add user2.hq P@ssw0rd**



**samba-tool user add user3.hq P@ssw0rd**



**samba-tool user add user4.hq P@ssw0rd**



**samba-tool user add user5.hq P@ssw0rd**



Теперь создадим группу и поместим туда созданных пользователей:

**samba-tool group add hq**

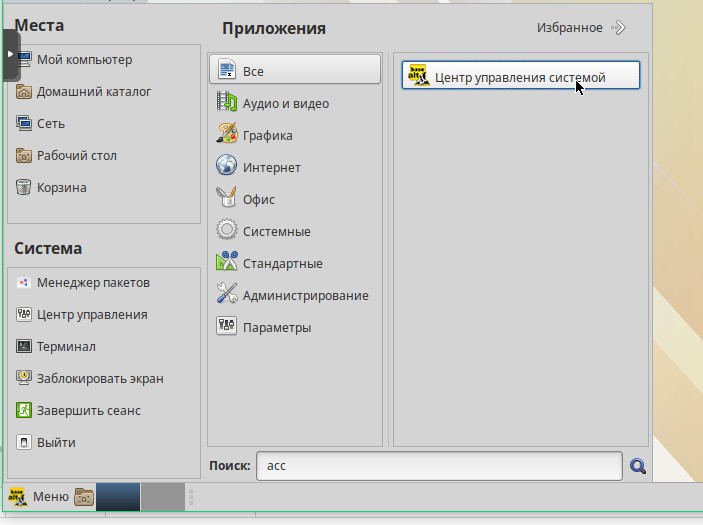


**samba-tool group addmembers hq user1.hq,user2.hq,user3.hq,user4.hq,user5.hq**

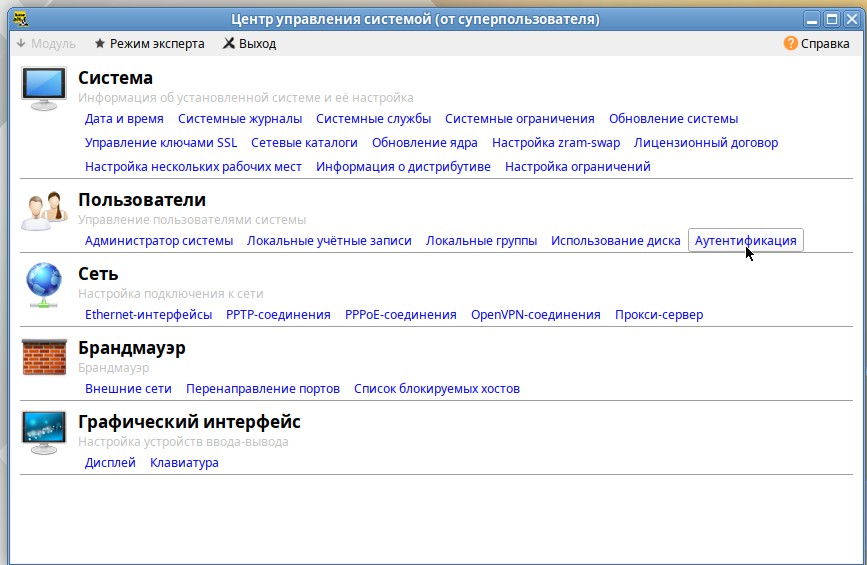


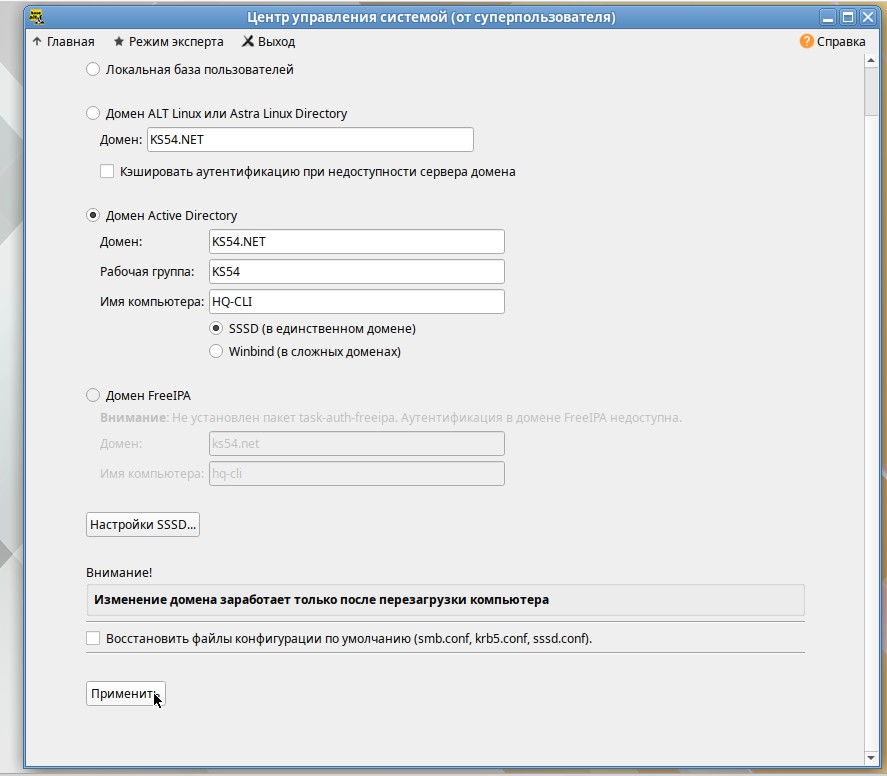
**Теперь введём клиентскую машину HQ-CLI в домен:**

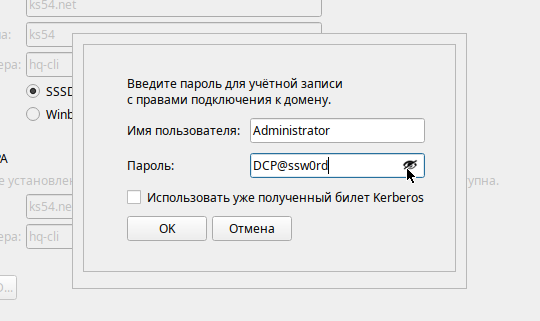
Найдем в меню Центр управления системой или в строке поиска введем аббревиатуру на английском **acc –** она нам выдаст центр управления быстро. Вводим пароль от суперпользователя: **toor**



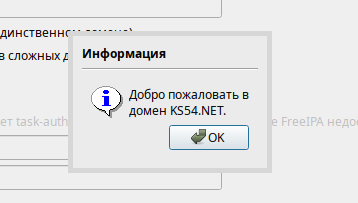
Находим в разделе пользователей строку Аутентификация и далее по скриншотам приведенным ниже добавляем машину в домен.

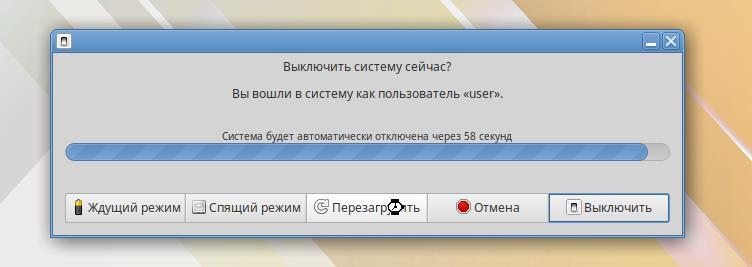




Вписываем все как на скриншоте выше/ Далее нажимаем Применить. И подтверждаем Вводим пароль, который вводили при настройке домена через samba-tool: **DCP@ssw0rd**

После ввода в домен должно появиться следующее сообщение на экране:



Перезагружаем машину **HQ-CLI**:

Далее можем залогиниться под одним из пяти доменных пользователя для теста и проверить, что все работает и машина в домене. Но не забудьте выйти и перезайти под локальным юзером (user / resu), нам эта учетка понадобиться для работы в дальнейшем.

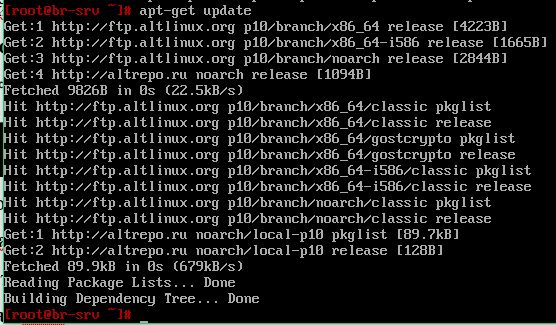
Чтобы настроить права созданных нами пользователей, нужно установить ещё один пакет на **BR- SRV**, но перед этим нужно подключить нужный репозиторий следующей командой:

**apt-repo add rpm** [**http://altrepo.ru/local-p10 noarch local-p10**](http://altrepo.ru/local-p10%20noarch%20local-p10)

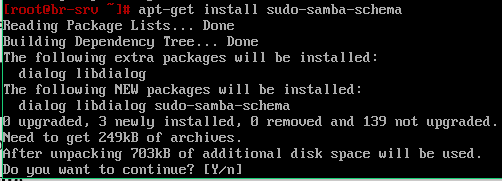
так как в базовых репозиториях данного пакета нет.



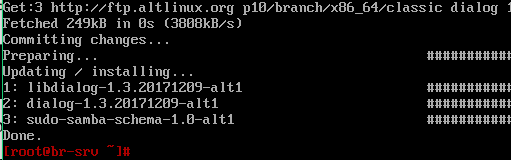
Далее обновление списка пакетов доступных репозиториев командной **apt-get update**



Устанавливаем пакет командой **apt-get install sudo-samba-schema** и подтверждаем установку



Получаем



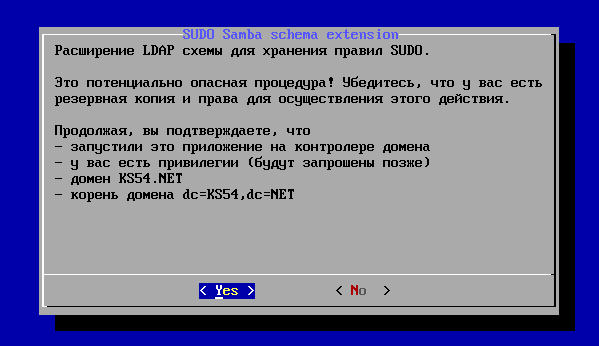
## ПРЕДВАРИТЕЛЬНО ПРОВЕРЬТЕ, ЧТО ПОСЛЕ ПЕРЕЗАГРУЗКИ СЕРВЕРА В ФАЙЛЕ

**/etc/resolv.conf ОСТАЛСЯ nameserver 192.168.100.2, если там внешний, то его нужно закомментировать и прописать указанный выше.**

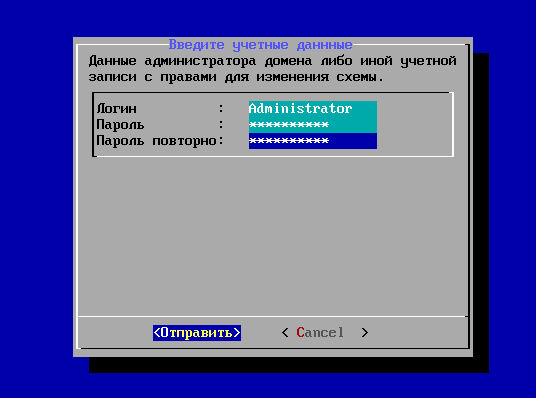
Добавим с помощью данной утилиты новую схему, используя команду **sudo-schema-apply**



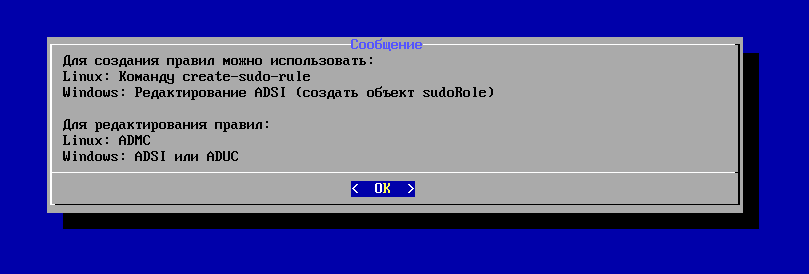
Далее откроется следующее диалоговое окно, в котором будем вводить необходимые параметры для создания. Сперва выбираем **Yes**



Затем происходит запрос пароля доменного администратора Вводим его **DCP@ssw0rd**



После должно появиться следующее окно:



Подтверждаем и он нас переводит обратно в консольную среду

 (у вас путь может быть другим, в зависимости от того, из какого каталога

вы начинали настройку схемы)

Далее мы создаём новое правило следующей командой (которую он сам предлагает в этом окне):

## create-sudo-rule

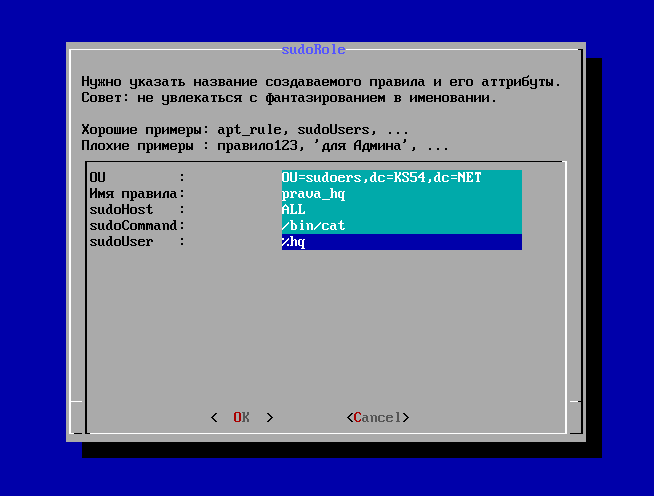


И вносим следующие изменения (имя правила можно любое):

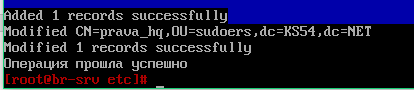
Имя правила : **prava\_hq**

sudoCommand : **/bin/cat**

sudoUser : **%hq**

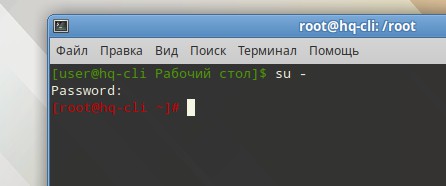


При успешном добавлении выведет следующие строки:



Заходим под локальным пользователем на клиентской машине **HQ-CLI** и получаем права root:

## su -



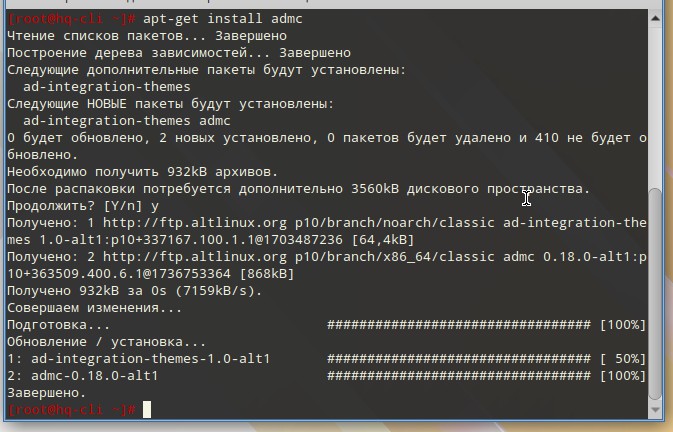
Обновляем список пакетов:

## apt-get update



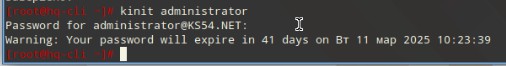
Далее ставим пакет admc:

## apt-get install admc



Затем создаём тикет доменного администратора, чтобы получить права на редактирование правил на сервере:

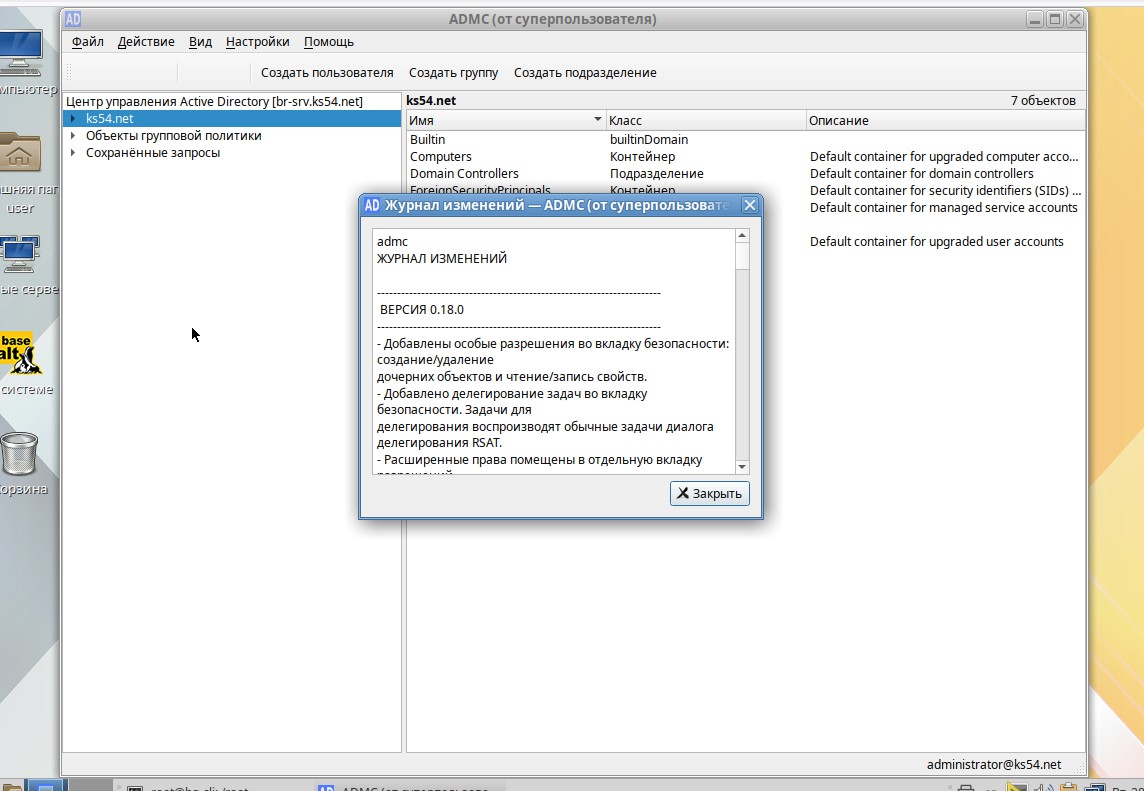
## kinit administrator DCP@ssw0rd



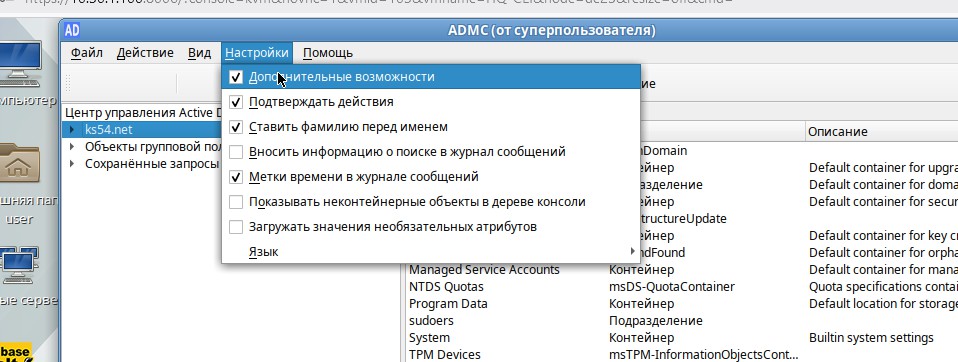
И запускаем admc:

## admc

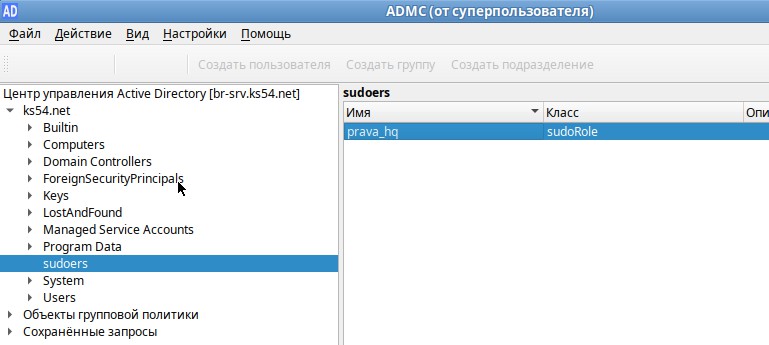
 и получаем:



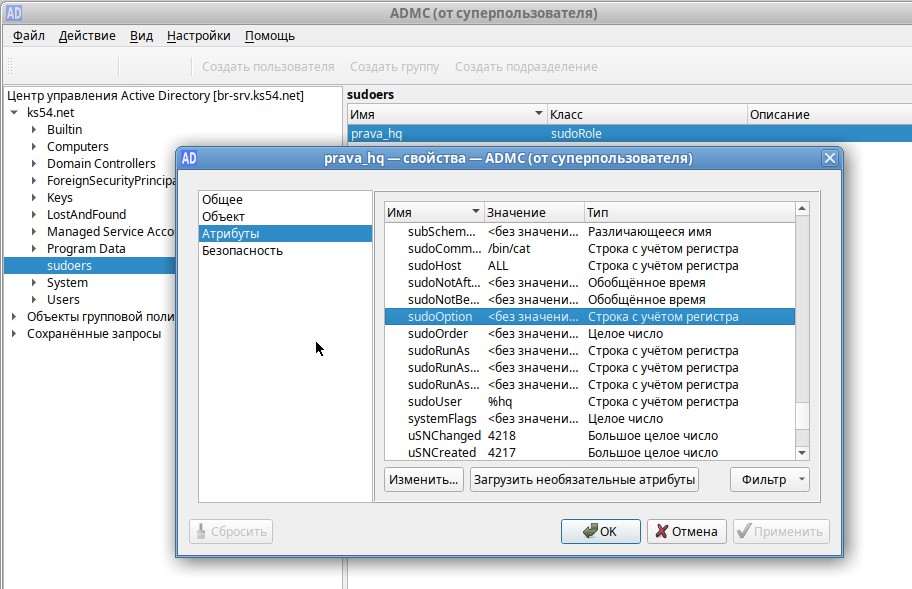
Включим дополнительные возможности через настройки:



Поменяем опцию **sudoOption** в созданном нами ранее правиле **prava\_hq** (правило всегда будет находиться в OU с названием **sudoers)**:

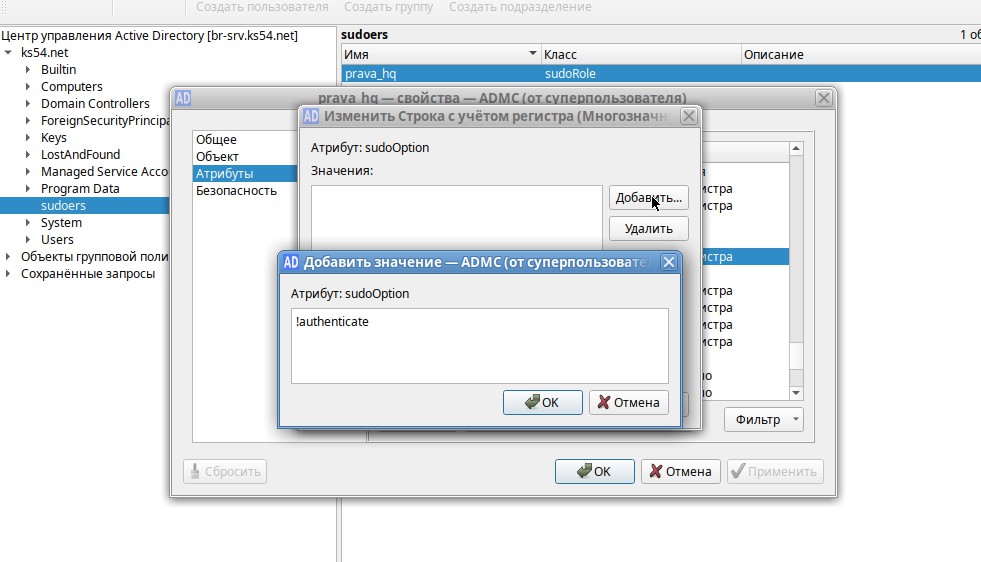


Двойной щелчок мыши и заходим в свойства нашего правила и находим sudoOption

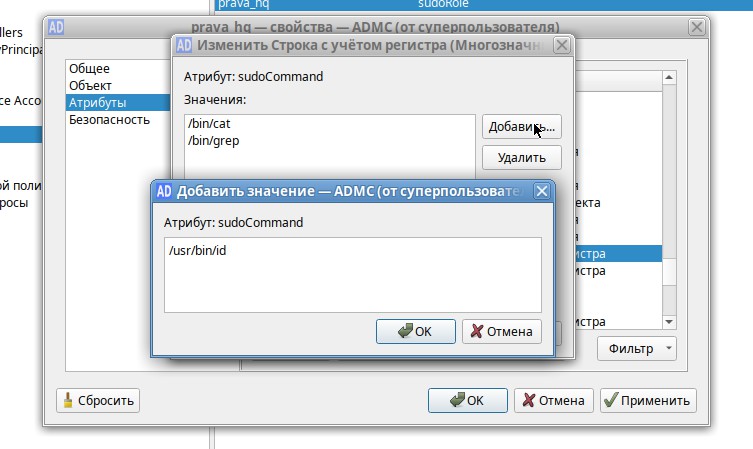


Вбиваем новое значение

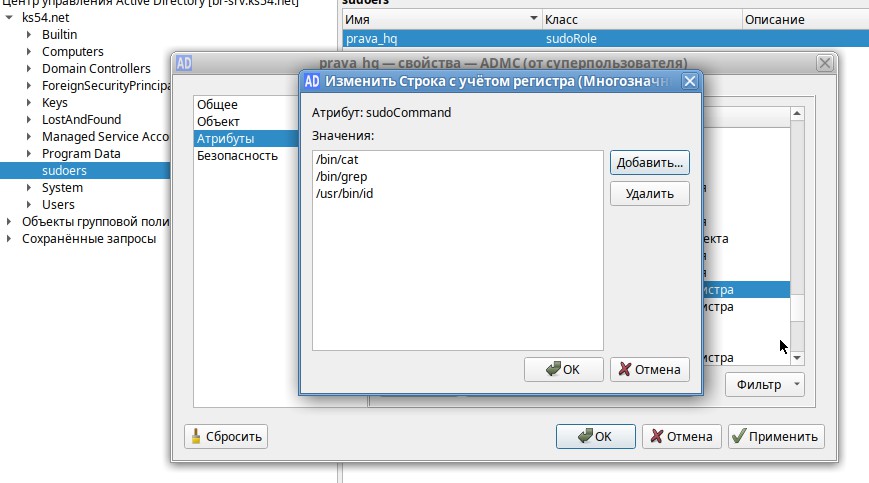
## !authenticate



И добавим ещё две команды в опцию **sudoCommand** (**grep** и **id**):



Получаем в итоге 3 команды



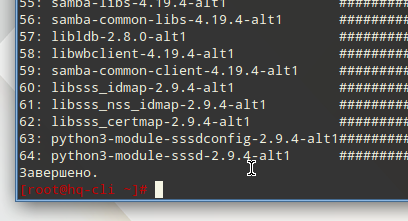
Обратите внимание, что путь до **id** отличается от других команд! Нажимаем Ок или Применить во всех диалоговых окнах и идем дальше.

Теперь, чтобы работали все созданные нами правила, нужно на **HQ-CLI** установить дополнительные пакеты:

## apt-get update

**apt-get install sudo libsss\_sudo**

Получаем установку и обновление нужных нам пакетов



Разрешаем использование sudo:

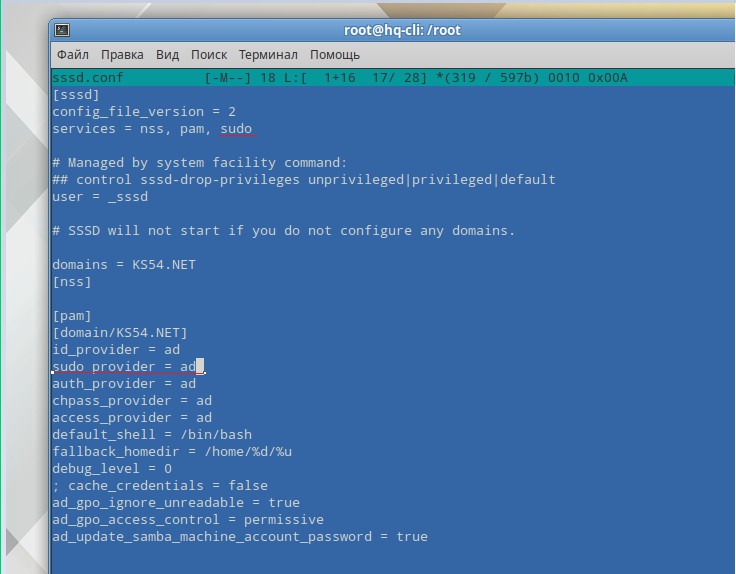
## control sudo public



Настроим конфиг **sssd.conf**: **mcedit /etc/sssd/sssd.conf**



В нем прописываем: **services = nss, pam, sudo sudo\_provider = ad**

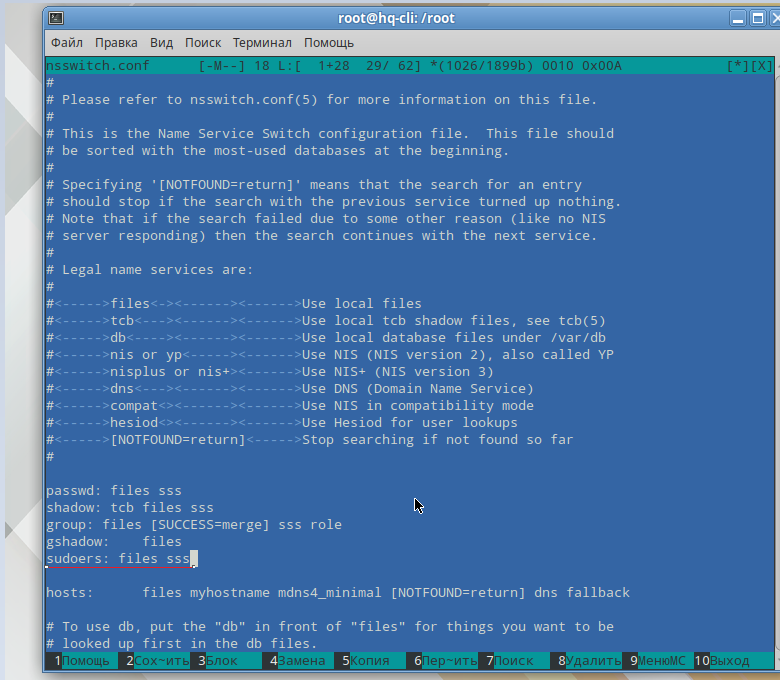


Теперь отредактируем **nsswitch.conf**: **mcedit /etc/nsswitch.conf**



Вносим следующее:

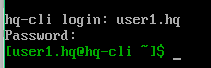
## sudoers: files sss



И перезагружаем службу **sssd**:

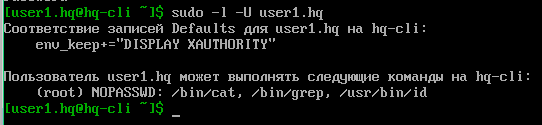
## systemctl restart sssd



Залогинимся под нашим доменным пользователем в терминальной линии tty2 (**сочетание клавиш Ctrl+Alt+F2**) командой **su user1.hq**, вбиваем пароль **P@ssw0rd** и получили

Теперь проверим, какие правила для sudoers получил наш доменный пользователь:

## sudo -l -U user1.hq



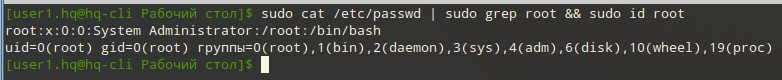
Вернёмся в графическую сессию (**сочетание клавиш Ctrl+Alt+F1**) и залогинимся под нашем доменным пользователем **user1.hq** и проверить настроенные права наглядно:

## Ctrl+Alt+F1

**sudo cat /etc/passwd | sudo grep root && sudo id root**



Получаем результат, где используются все три команды из политики разрешений, а значит права для нашего доменного пользователя настроены правильно



Настройка прав для группы **hq** завершена!

Приступаем к следующему этапу – **импортируем** пользователей из таблицы **Users.csv**. Файл Users.csv должен лежать в каталоге /opt..



Создаём файл **import**



и пишем туда следующий код:

**mcedit import**

**#!/bin/bash csv\_file=”/opt/Users.csv”**

**while IFS=”;” read -r firstName lastName role phone ou street zip city country password; do if [ “$firstName” == “First Name” ]; then**

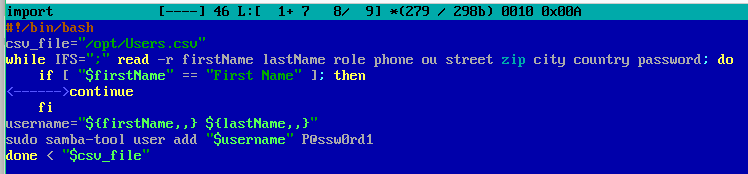
**continue**

**fi**

**username=”${firstName,,} ${lastName,,}”**

**sudo samba-tool user add “$username” P@ssw0rd1 done < “$csv\_file”**

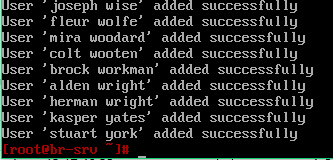
**Когда будете писать скрипт, он не будет у вас подсвечиваться разными цветами, все будет серым шрифтом. Вы сохраняете файл, потом просматриваете опять его и вуаля получаете красивый скрипт**



Выдаём файлу скрипта право на выполнение и запускаем его:

## chmod +x /root/import bash /root/import

Получаем:

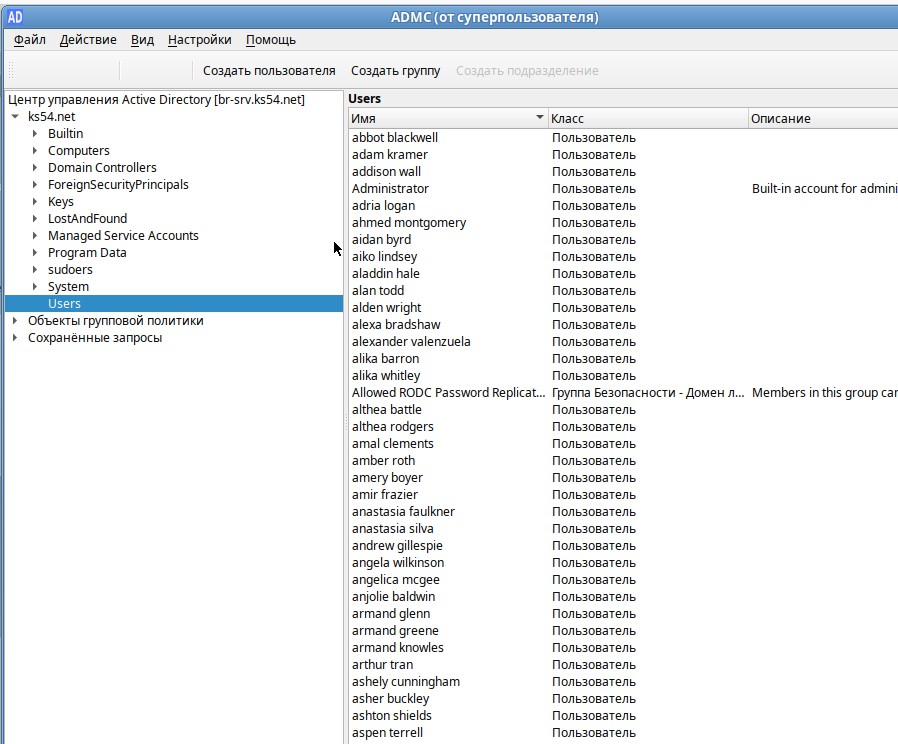


Добавляться они будут долго, но мы это сделали автоматизировано! Можно проверить добавленных пользователей. Есть несколько вариантов:

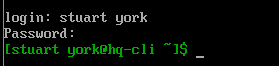
1. Проверить через **admc**. Для этого в терминале от пользователя root подтвердить право на редактирование параметров. **kinit administrator** и вводим пароль от нашего домена **DCP@ssw0rd**



Далее вбиваем admc и смотрим в домене ks54.net наших пользователей

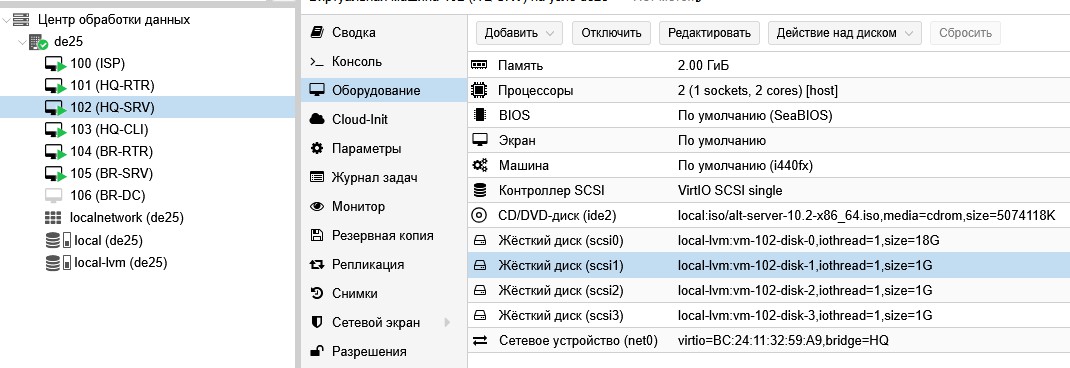


Или можем еще зайти под одним из этих пользователей через клиентскую машину, например, в терминальной линии CTRL+ALT+F2 (пароль мы по скрипту делали P@ssw0rd1)



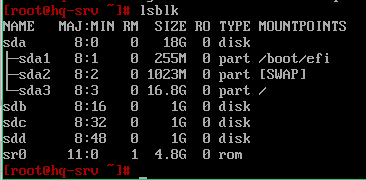
# КОНФИГУРАЦИЯ ФАЙЛОВОГО ХРАНИЛИЩА НА HQ-SRV

Сконфигурируем рейд-массив на HQ-SRV. Для начала проверяем, что у нас в свойствах виртуальной машины присутствует 3 неразмеченных диска по 1 Гб. Если это не так, нужно их добавить.



Теперь заходим в виртуальную машину и просматриваем все диски, которые мы подключили, следующей командой:

## lsblk

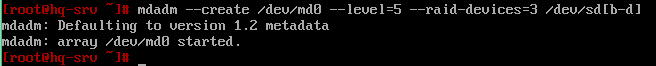


Теперь создадим дисковый массив уровня 5 из трёх дополнительных дисков следующей командой:

## mdadm - -create /dev/md0 - -level=5 - -raid-devices=3 /dev/sd[b-d]

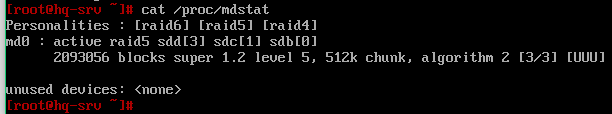


Получаем на выходе работы команды



Просмотрим статус созданного нами raid-массива:

## cat /proc/mdsat



Сохраним конфигурацию массива в файл **/etc/mdadm.conf** следующей командой:

## mdadm --detail -scan --verbose > /etc/mdadm.conf



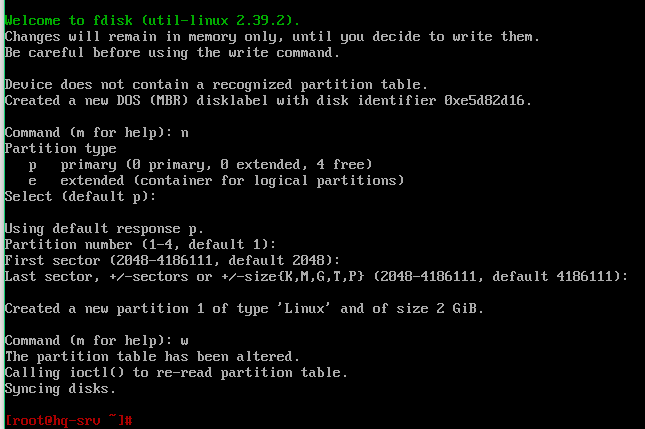
Теперь создаём раздел через **fdisk**.

Для этого пишем следующую команду:

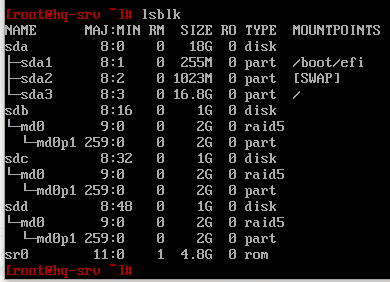
## fdisk /dev/md0



Затем пишем **n**, чтобы создать раздел, прокликиваем **Enter**, потому что он по дефолту предлагает то, что нам нужно, а в конце, когда предложит нам строчку Command снова - пишем **w**, чтобы записать изменения:

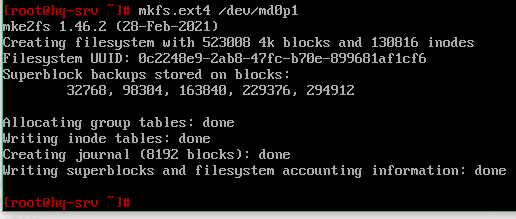


Проверим что нам покажет lsblk



Теперь создадим файловую систему, по заданию требуется **ext4**, создаём её следующей командой:

## mkfs.ext4 /dev/md0p1



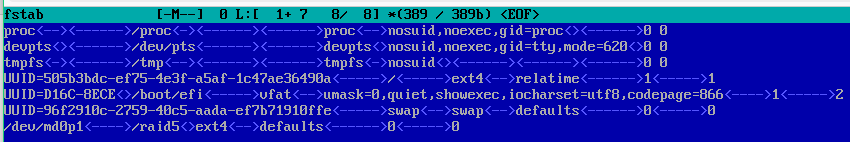
Настроим автоматическое монтирование в каталог **/raid5**.



Добавляем следующую строку в конец файла **/etc/fstab**:

## /dev/md0p1 /raid5 ext4 defaults 0 0

**(пробелы между значениями ставим через Tab)**



Затем создаём каталог **/raid5** и монтируем ФС из **/etc/fstab**:

## mkdir /raid5 mount –a

Заметьте, что команда не должна ничего выводить!

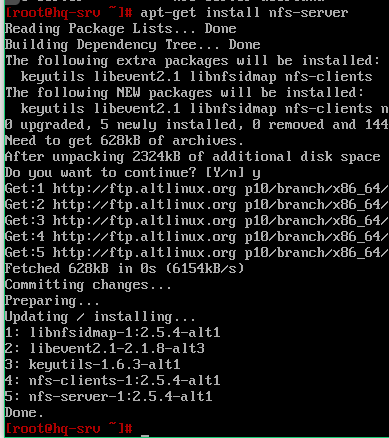
Теперь настроим сервер файловой системы **NFS**, для этого обновляем список пакетов и устанавливаем службу **nfs-server** следующей командой:

## apt-get update

**apt-get install nfs-server**



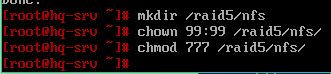
Подтверждаем желание установки и ждем окончания операции



Приступаем к самой настройке и создадим каталог, назначим нового владельца и группу ему и выдадим новые права:

## mkdir /raid5/nfs chown 99:99 /raid5/nfs

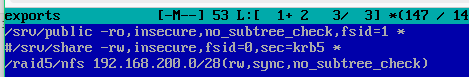
**chmod 777 /raid5/nfs**



Откроем каталог для общего доступа в сторону подсети, где находится **HQ-CLI**, для этого заходим в **/etc/exports**

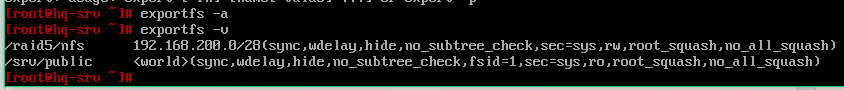
И пишем следующую строку в конец файла:

## /raid5/nfs 192.168.2.0/28(rw,sync,no\_subtree\_check)



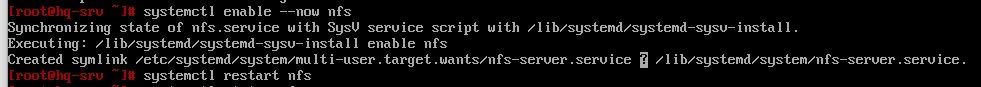
После редактирования файла применяем изменения и смотрим, что она экспортировалась:

## exportfs -a exportfs -v

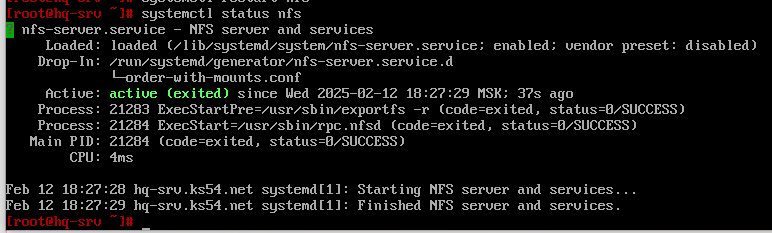


Включаем и перезапускаем службу NFS:

## systemctl enable - -now nfs systemctl restart nfs



Проверяем статус службы nfs



Теперь необходимо настроить систему на стороне клиента. **Переходим к монтированию на HQ-**

**CLI.** Для этого установим пакет nfs-clients

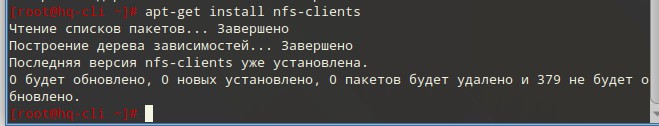
## apt-get update



**apt-get install nfs-clients**



Если она у вас уже установлена, то выдаст об этом информацию, в противном случае установит. В нашем – ничего не должен установить



Теперь настроим автоматическое монтирование в каталог **/mnt/nfs**, но для начала создадим его:

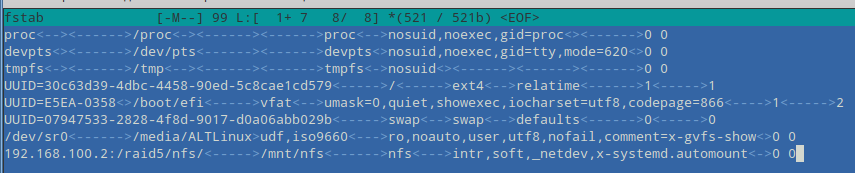
## mkdir -p /mnt/nfs



Добавляем следующую строку в конец файла **/etc/fstab**:



## 192.168.1.2:/raid5/nfs /mnt/nfs nfs intr,soft,\_netdev,x-systemd.automount 0 0



Монтируем ФС из файла **/etc/fstab** и проверяем, что она появилась в списке:

## mount –a mount –v

**Внизу видим ссылку на удаленный ресурс сервера HQ-SRV - 192.168.100.2**

Попробуем создать в нашем подключенном каталоге файл для проверки и посмотрим, появиться ли он на сервере

## touch /mnt/nfs/proverka\_nfs



Заходим на сервер и проверяем содержимое папки



Система полностью функционирует.

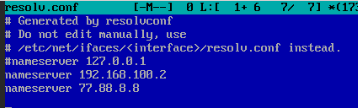
# НАСТРОЙКА СЛУЖБЫ СЕТЕВОГО ВРЕМЕНИ НА БАЗЕ СЕРВИСА CHRONY

Для начала проверим файл /etc/resolv.conf

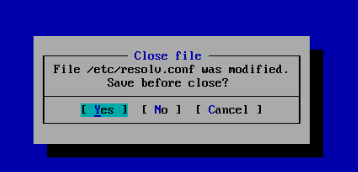
Если в нем отсутствуют внешние и локальные dns-сервера, то добавим строчки nameserver 192.168.100.2

nameserver 77.88.8.8

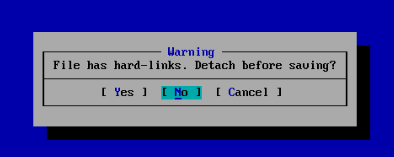
И приведем содержимое файла к следующему виду:



Далее выходим из редактирования с сохранением



Но не редактируем жесткие ссылки и выбираем No

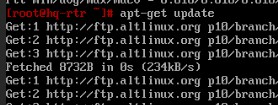


Если же в файле все хорошо и у вас идет ping ya.ru то тогда просто идете к следующему шагу.

пакет:

Для настройки сервера chrony на роутере **HQ-RTR**, обновим список пакетов и установим сам

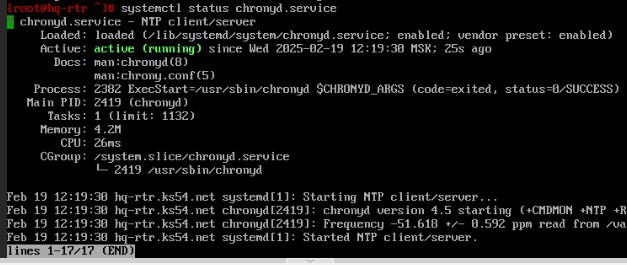
## apt-get update



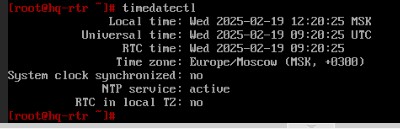
**apt-get install chrony**



Проверим работу службы **chrony** и **timedatectl**: **systemctl status chronyd.service**



## timedatectl



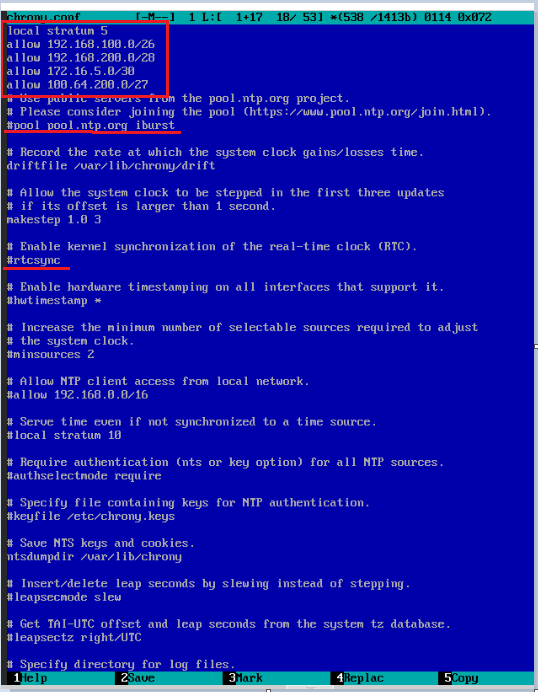
Видим, что и chronyd и NTP-сервер в tamedatectl активны.

Теперь приступим к настройке, редактируем файл **/etc/chrony.conf**,



введем в начало файла следующие строки и в целом приводим файл к такому виду (закомментировать в файле, что подчеркнуто на скриншоте):

## local stratum 5 allow 192.168.1.0/26 allow 192.168.2.0/28 allow 172.16.5.0/28 allow 192.168.4.0/27



Включаем и перезапускаем службу **chrony**: Запускаем ее снова

Добавляем ее в автозагрузку



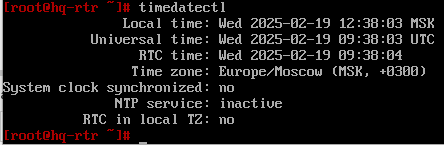
получаем на выводе:

Выключаем теперь синхронизацию, оставляя, по сути, главным сервером NTP – наш роутер **HQ- RTR,** и проверяем ещё раз статус timedatectl:

## timedatectl set-ntp 0



**timedatectl**



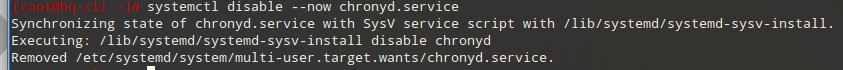
Видим, что ntp-сервер стал неактивным (он брался из сети интернет)

Теперь переходим к настройке клиента chrony, в качестве примера возьмём **HQ-CLI**. А далее сделаем остальные машины, которые указаны в задании.

Перед установкой новой службы выключим chrony на **HQ-CLI**: **systemctl disable --now chronyd.service**

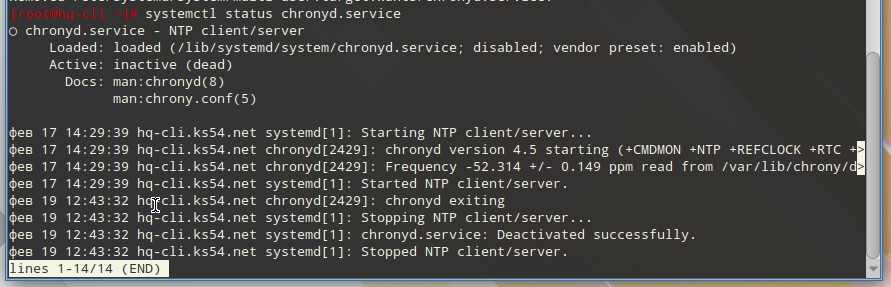


Он удалится из автозагрузки, о чем пишет в результате срабатывания команды

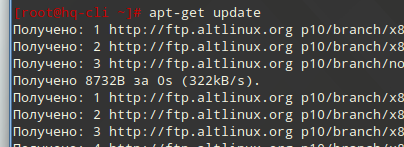


Проверяем статус

## systemctl status chronyd.service



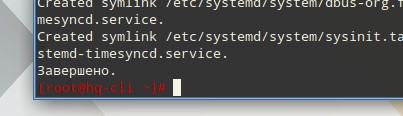
Обновляем список пакетов на **HQ-CLI** и скачиваем службу **systemd-timesyncd**: **apt-get update**



## apt-get install systemd-timesyncd



Подтверждаем установку и ожидаем ее окончания.



Теперь заходим в конфиг **/etc/systemd/timesyncd.conf**



и отредактируем только одну строку:

## NTP=192.168.100.1



Теперь включим службу **systemd-timesyncd**

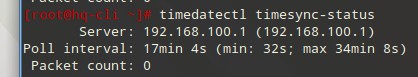


Добавим в автозагрузку и посмотрим её статус работы:

## systemctl enable --now systemd-timesyncd.service timedatectl timesync-status

Если вывод соответствует, то все хорошо.

Если вывод не такой как на скришноте выше, а 2-3 строчки всего:

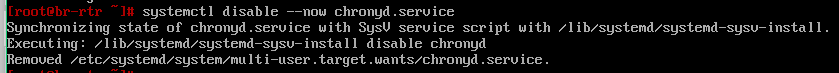


То тогда стоит проверить работает ли сервис chronyd на HQ-RTR. В случае необходимости перезапустите сервис **systemctl restart chronyd.service**

Проделаем тоже самое и для остальных машин, которые будут являться клиентами локального сервера времени.

Для начала начнем со второго роутера **BR-RTR**

Остановим chronyd и ntp, чтобы не мешал работе system-timesyncd



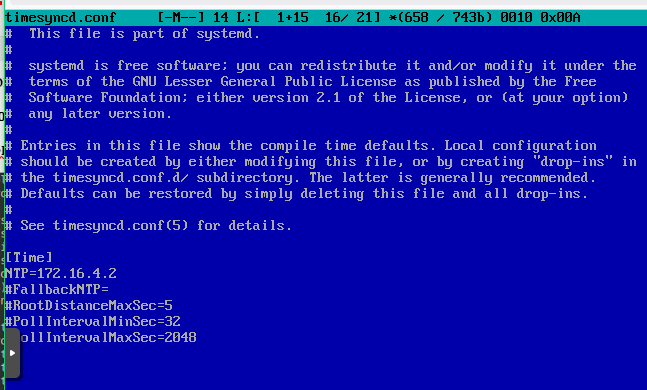


Ставим пакет службы system-timesyncd



Зайдём в конфиг **/etc/systemd/timesyncd.conf** и отредактируем только одну строку (учитываем, что для BR-RTR айпи нужен внешнего интерфейса, так как связь идет с HQ-RTR через него):

**NTP=172.16.4.2**

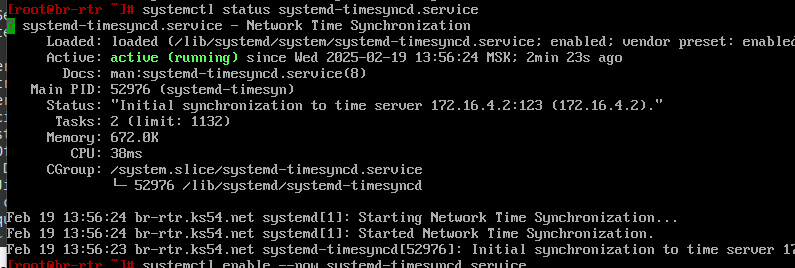


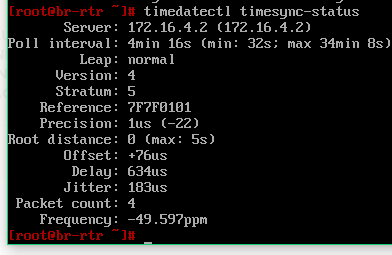
Теперь включим службу **systemd-timesyncd** и посмотрим её статус работы:





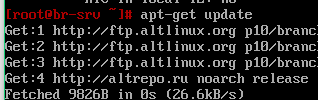
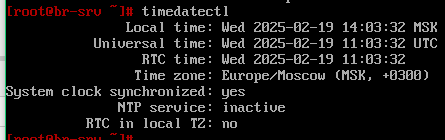
Далее проверяем статус



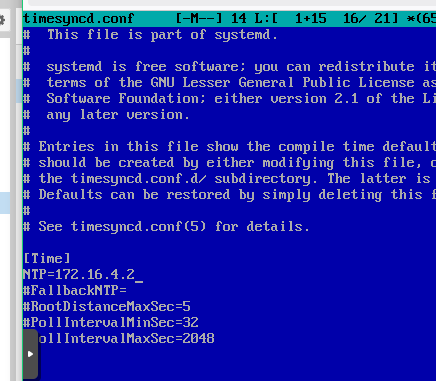
И проверим работу и связь с локальным сервером времени

Настроим теперь тоже самое на **BR-SRV**. Настройки аналогичные BR-RTR Кратко повторим:











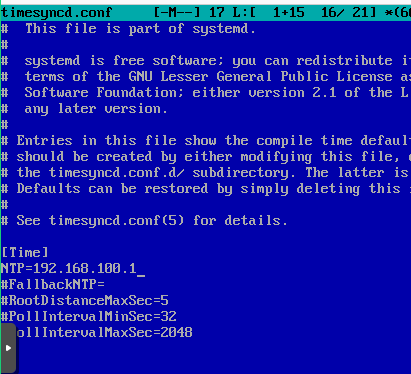
Все работает. Осталось настроить на сервере **HQ-SRV**

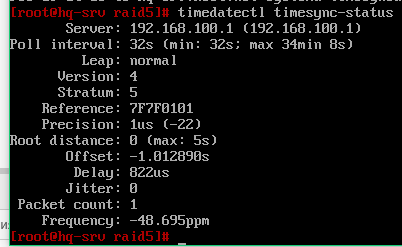
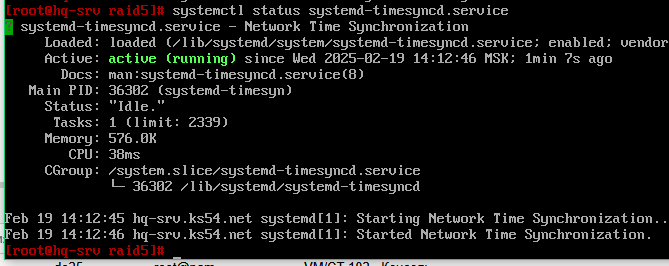
Настройки аналогичные, то адрес ntp будет 192.168.100.1

Пусть из которого происходят все дальнейшие манипуляции не играют роль, у вас он может быть другим, главное результат







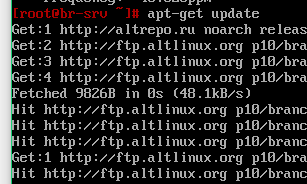


**На выходе получаем полностью выполненное задание по настройке сетевого времени на базе сервера chrony.**

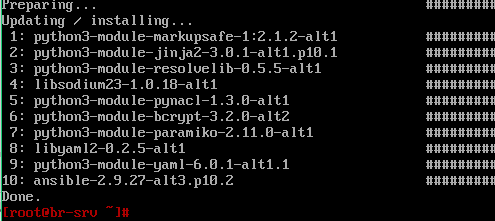
# СКОНФИГУРИРУЙТЕ ANSIBLE НА СЕРВЕРЕ BR-SRV

## BR-SRV

Для начала проверим, обновлены ли у нас списки пакетов и затем попробуем установить **ansible**: **apt-get update**



## apt-get install ansible



Далее нам нужен рабочий каталог для нашего ansible, который может быть уже создан, НО если нет, то создадим его следующей командой: mkdir /etc/ansible

У нас он уже создан, и там же находится нужный нам файл **hosts**, тоже уже созданный, НО в случае его отсутствия, нужно также его создать: mcedit /etc/ansible/hosts

Проверим содержимое папки, чтобы понять, что нам необходимо сделать из вышеуказанного.

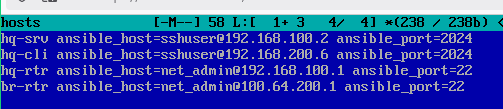


Теперь нам нужно написать следующие строки в файл **hosts**:

## hq-srv [ansible\_host=sshuser@192.168.100.2](mailto:ansible_host%3Dsshuser@192.168.100.2) ansible\_port=2024

**hq-cli** [**ansible\_host=sshuser@192.168.200.6**](mailto:ansible_host%3Dsshuser@192.168.200.6) **ansible\_port=2024** (у вас может быть другой адрес, проверьте)

## hq-rtr [ansible\_host=net\_admin@192.168.100.1](mailto:ansible_host%3Dnet_admin@192.168.100.1) ansible\_port=22 br-rtr [ansible\_host=net\_admin@100.64.200.1](mailto:ansible_host%3Dnet_admin@100.64.200.1) ansible\_port=22



И настроим в каталоге **/etc/ansible** файл **ansible.cfg**: **mcedit /etc/ansible/ansible.cfg**



Добавим под строку **[defaults]** ещё одну:

## ansible\_python\_interpreter=/usr/bin/python3



Но, возникает другая проблема, в первом модуле мы настраивали SSH на серверах, однако маршрутизаторы **HQ-RTR**, **BR-RTR** и клиент **HQ-CLI** не входили в пункт по настройке. Теперь его нужно настроить сейчас, на этих устройствах, поэтому приступаем к их настройке, чтобы мы могли выполнить задание по **ansible**.

Первым делом настроим ssh на **HQ-RTR**

Обновим пакеты и установим ssh-сервер

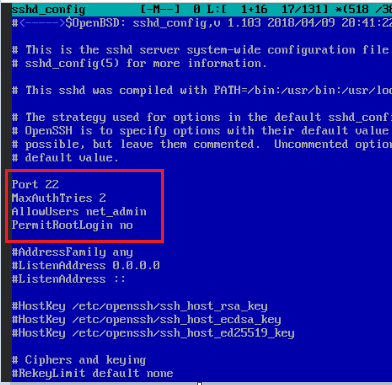




После установки заходим в конфигурационный файл **mcedit /etc/openssh/sshd\_config** и вносим в него следующие строки:

## Port 22

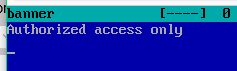
**MaxAuthTries 2 AllowUsers net\_admin PermitRootLogin no**



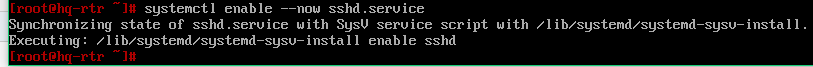
Создаем баннер по аналогии с заданием 1 модуля и добавляем ссылку на его в конфигурационный файл sshd\_config.



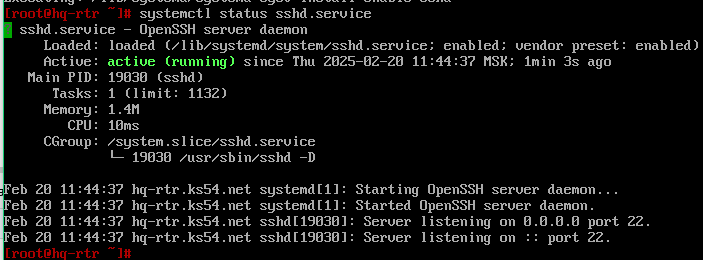
(не забываем про один Enter после фразы, он обязателен)



прописываем ссылку на баннер в конфигурационном файле Запускаем службу ssh и ставим в автозагрузку системы



Проверяем статут нашего ssh-сервера



Проделываем все то же самое и с роутером **BR-RTR**, что делали с HQ-RTR



обновляем пакеты

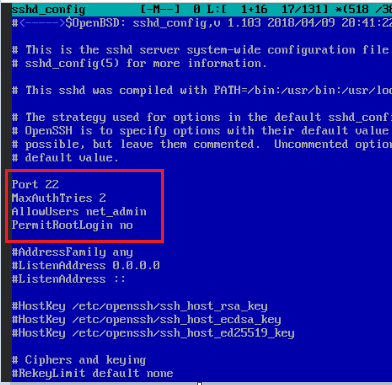
устанавливаем ssh-сервер

редактируем конфигурационный файл и

вписываем туда параметры

## Port 22

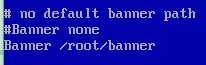
**MaxAuthTries 2 AllowUsers net\_admin PermitRootLogin no**



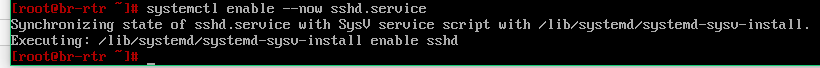
Создаем баннер по аналогии с заданием 1 модуля и добавляем ссылку на его в конфигурационный файл sshd\_config.



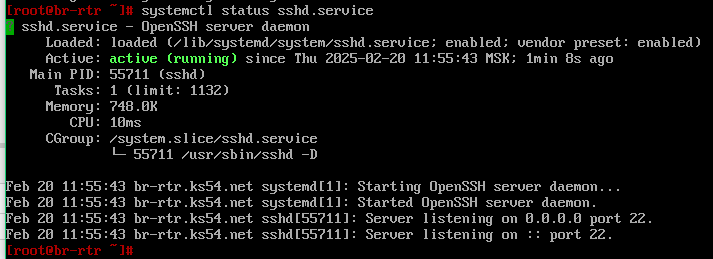
Вносим ссылку на баннер в конфигурационный файл sshd\_conf



Запускаем службу ssh и ставим в автозагрузку системы



Проверяем статут нашего ssh-сервера



Осталось настроить ssh-сервер на **HQ-CLI**

Здесь есть особенности, так как по 1 модулю у нас нет пользователя под ssh, нам нужно будет его создать, и затем поднять сервер ssh.

Создадим пользователя sshuser на HQ-CLI



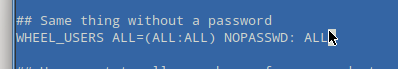
Зададим пользователю пароль P@ssw0rd



Заходим в файл /etc/sudoers



и раскомментируем строчку **WHEEL\_USERS ALL=(ALL:ALL) NOPASSWD: ALL**



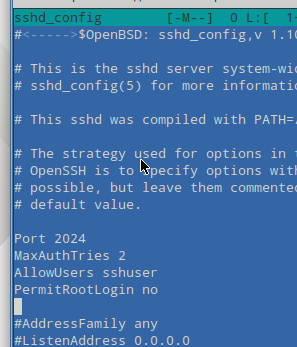
Добавляем нашего пользователя sshuser в группу wheel. Для этого используем команду

## usermod –aG wheel sshuser

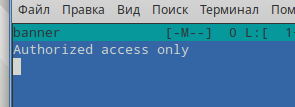


Теперь установим ssh-сервер и настроим его.

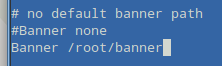
прописываем 4 параметра.



Добавим баннер



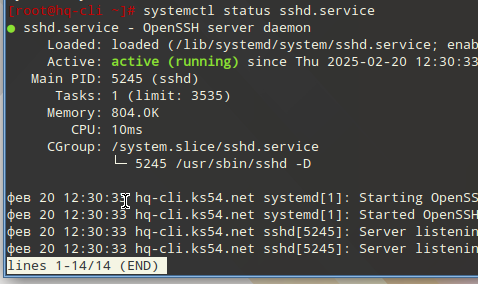
Добавляем ссылку на наш баннер в конфигурационный файл



Перезапускаем сервер и добавляем его в автозагрузку



Проверяем работу

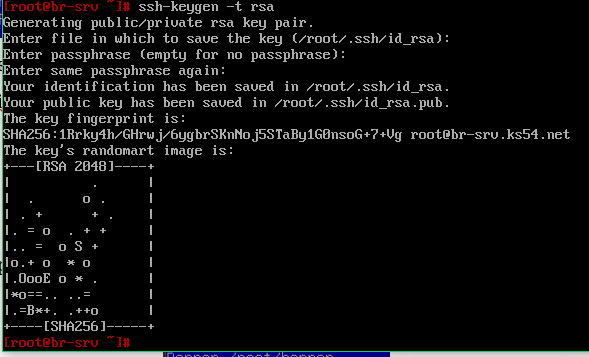


Теперь все готово к дальнейшей настройке ansible.

Теперь на **BR-SRV** генерируем ключи **RSA**, чтобы экспортировать их на машины клиенты, строку с путём и **passphrase** оставляем пустой:

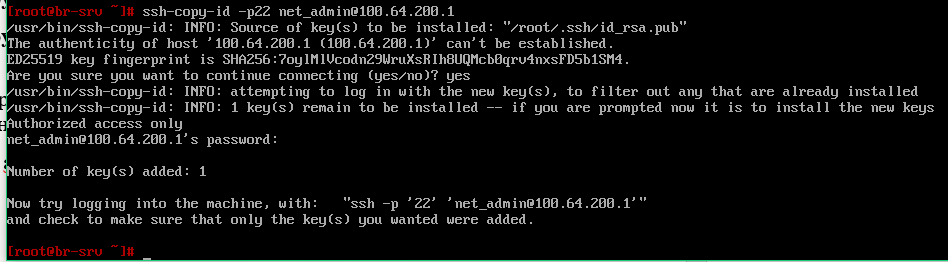
## ssh-keygen -t rsa



Прищёлкиваем везде Enter не вводя ничего и получаем на выходе:

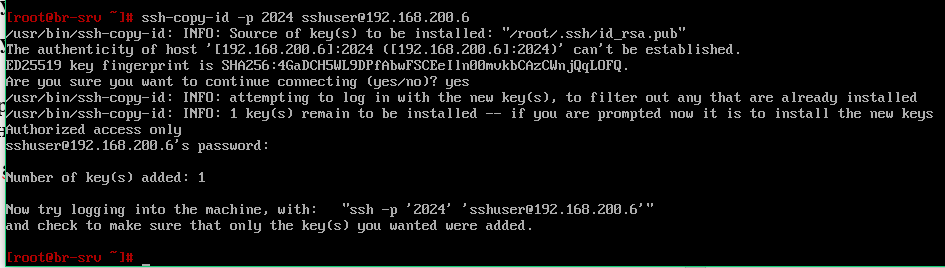
Копируем публичный ключ на клиентские машины, первая из них будет **BR-RTR**: **ssh-copy-id -p 22** [**net\_admin@100.64.200.1**](mailto:net_admin@100.64.200.1)

Потребуется при подтверждении написать **yes** (сокрашенно y он не примет), а также ввести пароль от пользователя net\_admin. Это пароль P@ssw0rd



Аналогично скопируем все на остальные клиенты ansible наш ключ.

**ssh-copy-id -p 2024** [**sshuser@192.168.200.6**](mailto:sshuser@192.168.200.6)(Но у вас может быть другой IP, т.к. адрес он получает по DHCP)



## ssh-copy-id -p 2024 [sshuser@192.168.100.2](mailto:sshuser@192.168.100.2)



**ssh-copy-id -p 22** [**net\_admin@192.168.100.1**](mailto:net_admin@192.168.100.1)

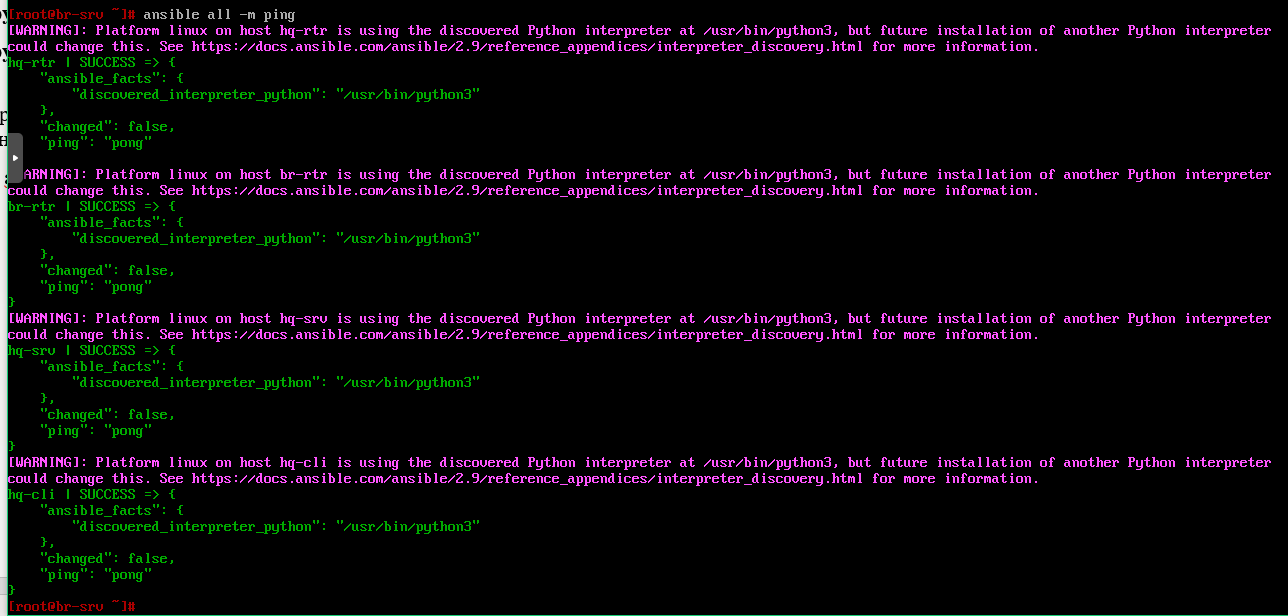


После этого мы можем проверить связь. Машины должны без предупреждений и ошибок отвечать

**pong** на команду **ping** в **ansible** посланную с **BR-SRV**: **ansible all -m ping**



Получаем на выходе:



Ansible работает, задание выполнено.

# РАЗВЕРТЫВАНИЕ ПРИЛОЖЕНИЙ В DOCKER НА СЕРВЕРЕ

**BR-SRV**

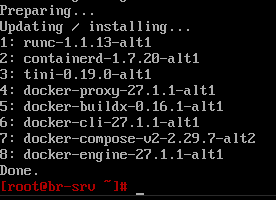
Перед настройкой нам необходимо обновить список пакетов и установить **docker-engine** и **docker- compose**:

## apt-get update



**apt-get install docker-engine docker-compose**

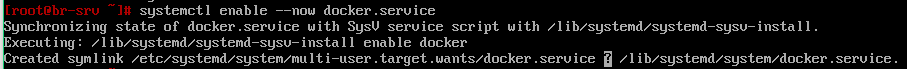


Подтверждаем установку и ждем окончания операции.

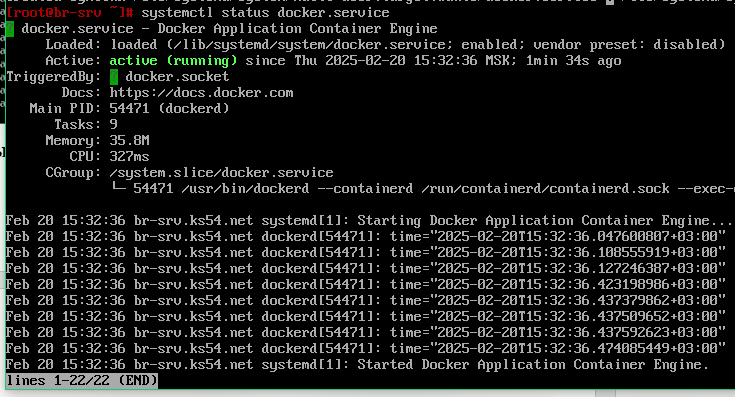
Запускаем службу **docker**:



Добавляем ее в автозагрузку с помощью команды **systemctl enable --now docker.service**



Проверяем статус службы **systemctl status docker.service**

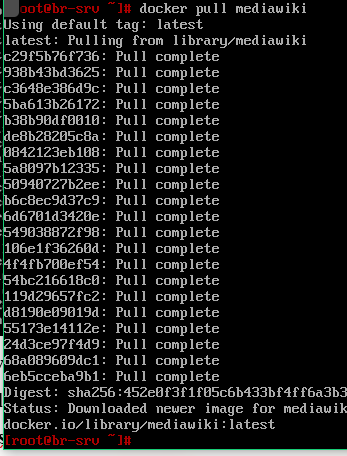


Загружаем нужные образы по заданию следующими командами:

## docker pull mediawiki



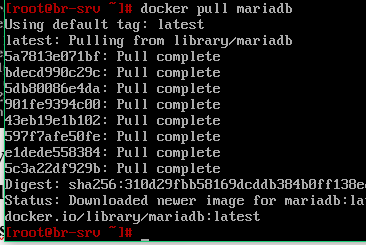
Ждем загрузки полной и получаем следующий вывод команды



## docker pull mariadb



Получаем после скачивания образа



Создаем в домашней директории пользователя файл, в качестве пользователя, которого мы создавали при установке ОС, у нас – **user**, а его домашний каталог – **/home/user**, файл называется – **wiki.yml,** для приложения MediaWiki:

## mcedit /home/user/wiki.yml



И заполняем его следующими строками, обратите внимание, что в строках ПРОБЕЛЫ, А НЕ ТАБУЛЯЦИЯ (клавиша Tab):

**services:**

**mariadb:**

**image: mariadb container\_name: mariadb restart: always environment:**

**MYSQL\_ROOT\_PASSWORD: rootP@ssw0rd MYSQL\_DATABASE: mediawiki**

**MYSQL\_USER: wiki MYSQL\_PASSWORD: WikiP@ssw0rd volumes: [ mariadb\_data:/var/lib/mysql ] wiki:**

**image: mediawiki container\_name: wiki restart: always environment:**

**MEDIAWIKI\_DB\_HOST: mariadb MEDIAWIKI\_DB\_USER: wiki**

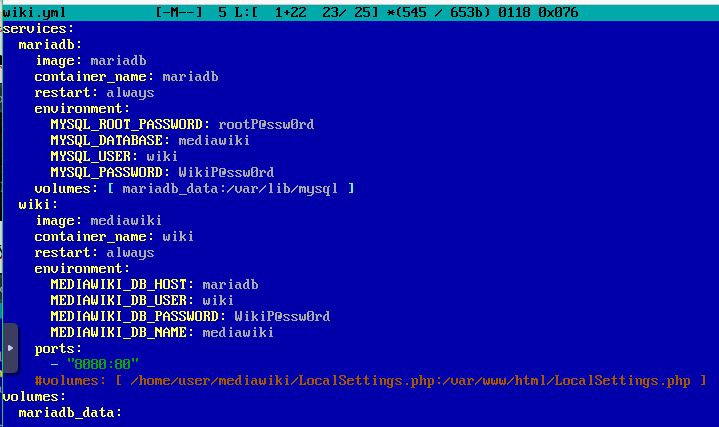
**MEDIAWIKI\_DB\_PASSWORD: WikiP@ssw0rd MEDIAWIKI\_DB\_NAME: mediawiki**

**ports:**

**- "8080:80"**

**#volumes: [ /home/user/mediawiki/LocalSettings.php:/var/www/html/LocalSettings.php ] volumes:**

**mariadb\_data:**

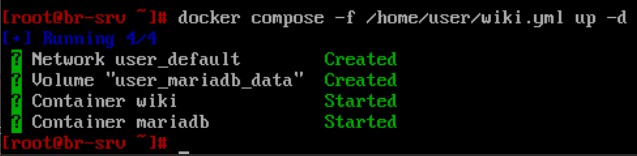
Количество пробелов сравнивайте по расположению букв, но для ориентира разница в отступах вперед и назад в 2 пробела

После всех настроек строку **volumes..** мы обратно раскомментируем, убрав символ **#** Приступаем к запуску контейнера **wiki.yml** с помощью команды:

## docker compose -f /home/user/wiki.yml up -d



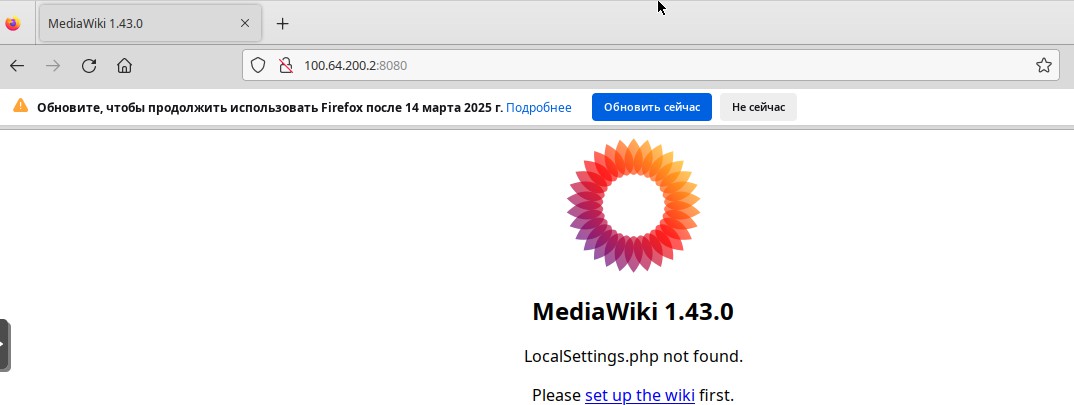
Должны получить на выходе старт сервисов:



Теперь для теста заходим с клиента **HQ-CLI** на сайт после запуска контейнера:

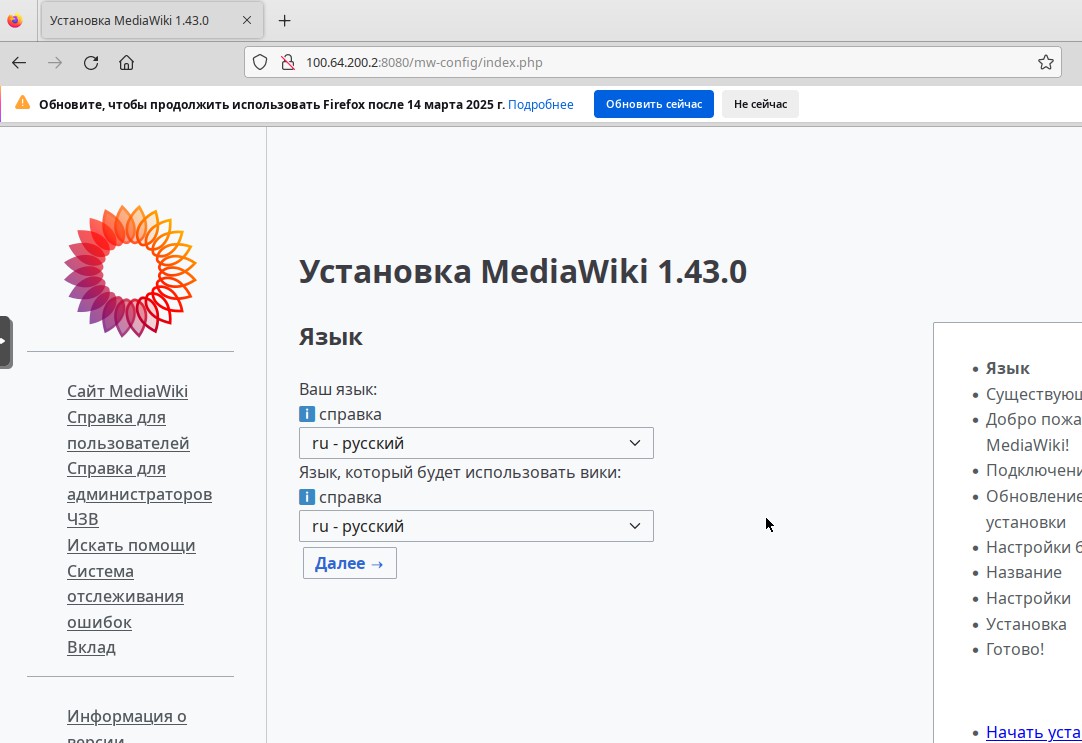
В браузере вбиваем ip-адрес BR-SRV с номером порта 8080

Получаем:

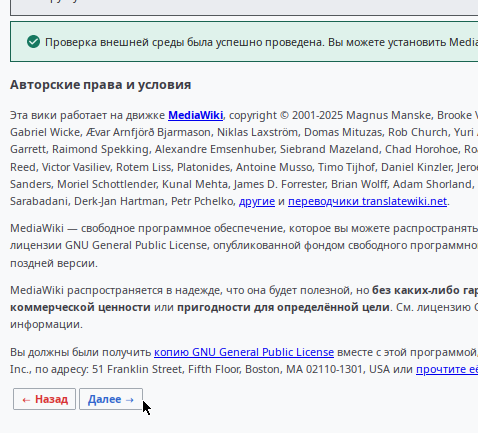
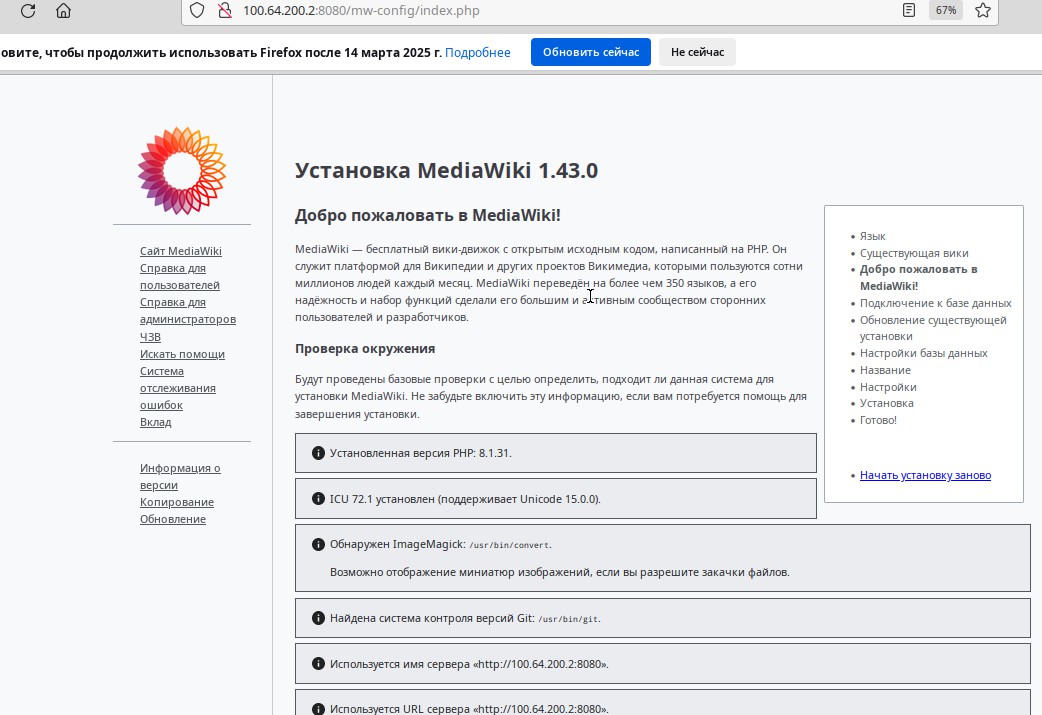


Видим, что файл **LocalSettings.php** не найден, и нажимаем на **complete the installation** или **set up the wiki**.

Выбираем удобный для вас язык:



Здесь просто идём далее:



Видим строки, которые нужно заполнить: Хост базы данных:

## mariadb

Имя базы данных (без дефисов):

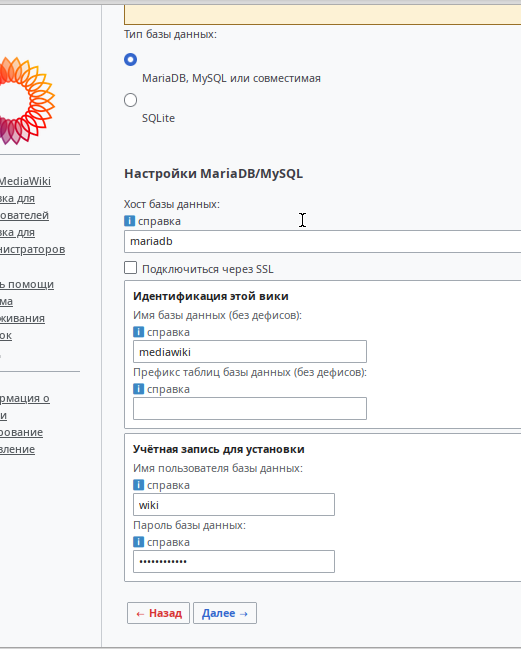
## mediawiki

Имя пользователя базы данных:

## wiki

Пароль базы данных:

## WikiP@ssw0rd



Затем Прожимаем **Далее**, оставляя всё как есть:



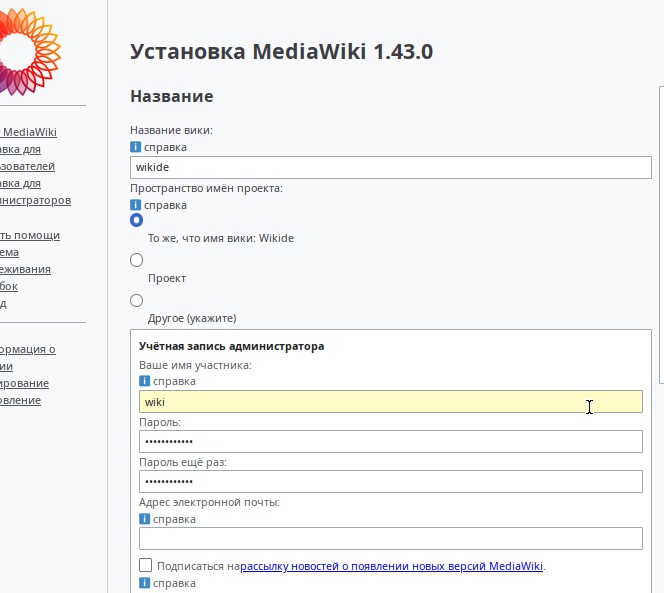
Пишем в строках следующее и выбираем пункты, как на скрине: Название вики:

**wikide** (можно своё название) Ваше имя участника:

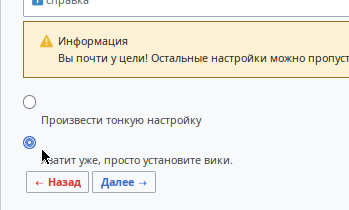
## wiki

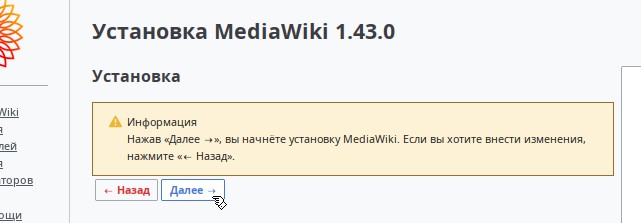
Пароль:

## WikiP@ssw0rd

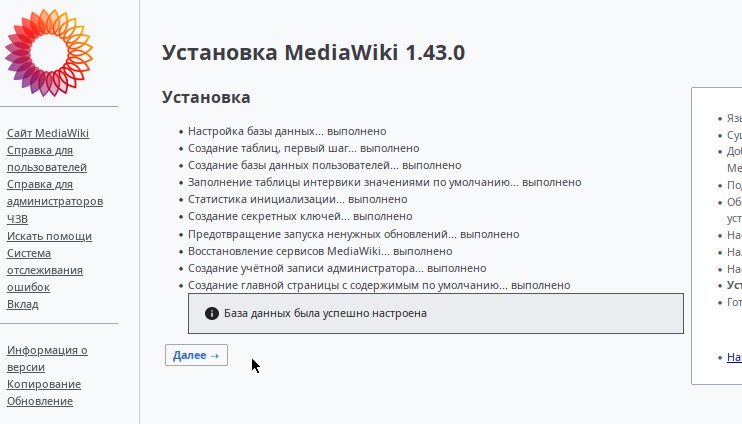


Внизу прожимаем «Хватит уже, просто установите вики»

 и жмем дадее. Затем опять далее:



И вот мы успешно создали базу данных:



Далее автоматически скачивается файл **LocalSettings.php**, который нужно переместить теперь на сервер с **mediawiki**, а именно на **BR-SRV** c **HQ-CLI**:

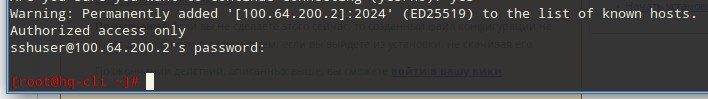
## scp -P 2024 /home/user/Загрузки/LocalSettings.php sshuser@100.64.200.2:/home/sshuser/



Далее пишем **yes**



Вводим пароль P@ssw0rd и вуаля, файл там, где нам нужно.



Теперь заходим на сервер **BR-SRV** и перемещаем скачанный файл в **/root**, но перед этим удаляем то, что создалось в **/root** (могло и не создаваться, так даже лучше):

## rm -rf /home/user/LocalSettings.php (в нашем случае этот файл не создался) mkdir /home/user/mediawiki



**mv /home/sshuser/LocalSettings.php /home/mediawiki/ ls /home/user/mediawiki/**

Раскомментируем, как и говорили ранее, строку **volumes…**:

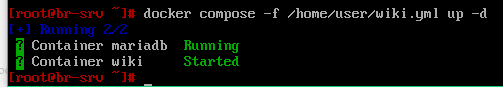


Удаляем # перед строчкой volumes

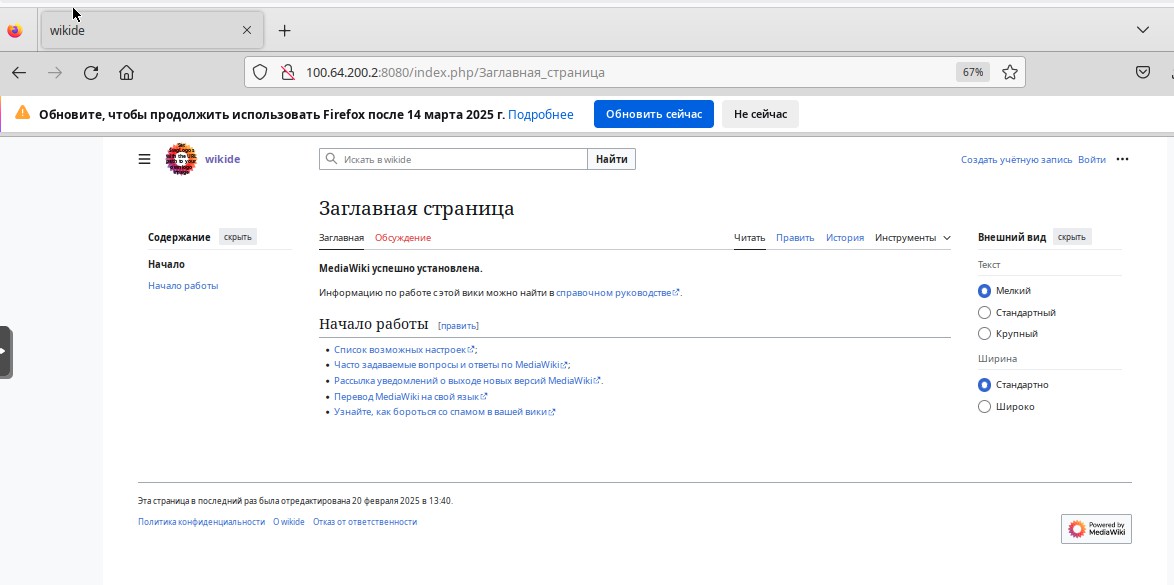


Теперь перезапускаем контейнеры путём запуска контейнера ещё раз:

## docker compose -f wiki.yml up -d



Проверим работу сайта, зайдем вновь через клиента **HQ-CLI** и увидим домашнюю страницу сайта:



Задание выполнено.

# НА МАРШРУТИЗАТОРАХ СКОНФИГУРИРУЙТЕ СТАТИЧЕСКУЮ ТРАНСЛЯЦИЮ ПОРТОВ

Пробросим порт **80** в порт **8080** и порт **2024** в порт **2024** на **BR-SRV** на маршрутизаторе **BR-RTR**, для обеспечения работы сервиса **mediawiki** и **ssh**, правила прописываем через консоль:

**iptables -t nat -A PREROUTING -p tcp -d 100.64.200.1 --dport 80 -j DNAT --to-destination 100.64.200.2:8080 iptables -t nat -A PREROUTING -p tcp -d 100.64.200.1 --dport 2024 -j DNAT --to-destination 100.64.200.2:2024**



Сохраняем правила, не забывайте, что у вас уже есть правила, которые мы писали ещё в первом модуле, проверьте, чтобы в этом файле сохранялись и прошлые, и новые (которые мы сейчас ввели):

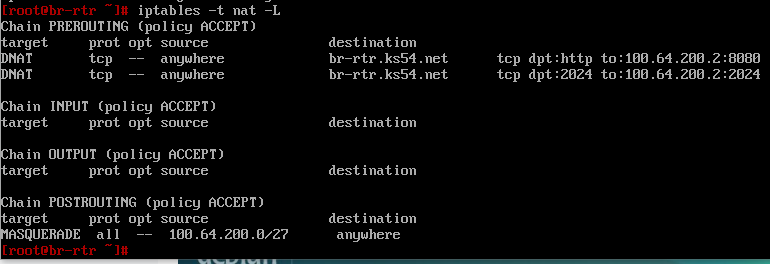
## iptables-save > /root/rules



Проверим наши правила, что они дописались в наш iptables командой:

## iptables –t nat -L

И убеждаемся, что наши правила запушены.



Пробросим порт **2024** в порт **2024** на **HQ-SRV** на маршрутизаторе **HQ-RTR**, для обеспечения работы сервиса ssh, правило прописываем через консоль:

**iptables -t nat -A PREROUTING -p tcp -d 192.168.100.1 --dport 2024 -j DNAT --to-destination 192.168.100.2:2024**

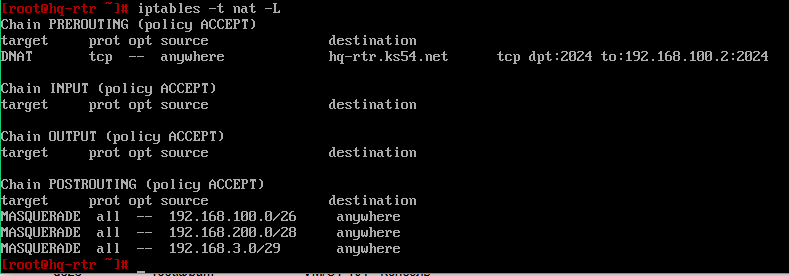


Сохраняем правила, не забывайте, что у вас уже есть правила, которые мы писали ещё в первом модуле, проверьте, чтобы в этом файле сохранялись и прошлые, и новое (которое мы сейчас ввели):

## iptables-save > /root/rules

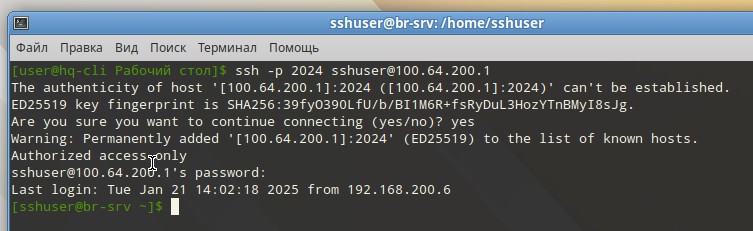


Проверим сохраненные правила



Проверим правила путём подключения с клиента **HQ-CLI** по **ssh** к серверу **BR-SRV** через IP-адрес роутера **BR-RTR**:

## ssh -p 2024 [sshuser@100.64.200.1](mailto:sshuser@100.64.200.1)



Как видим, стучались на адрес роутера BR-RTR через ssh, но правило сработало и перенаправило на нужный нам сервер BR-SRV.

Задание выполнено.

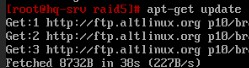
# ЗАПУСТИТЕ СЕРВИС MOODLE НА СЕРВЕРЕ HQ-SRV

Устанавливаем для этого ряд пакетов, которые будут нам нужны для работы **(увы их много и их придется запомнить)**:

* apache2
* php8.2
* apache2-mod\_php8.2
* mariadb-server
* php8.2-opcache
* php8.2-curl
* php8.2-gd
* php8.2-intl
* php8.2-mysqli
* php8.2-xml
* php8.2-xmlrpc
* php8.2-ldap
* php8.2-zip
* php8.2-soap
* php8.2-mbstring
* php8.2-json
* php8.2-xmlreader
* php8.2-fileinfo
* php8.2-sodium

Обновляем список репозитория

## apt-get update



Установить пакеты можно либо каждый отдельно и последовательно, как по списку, приведенному выше, либо в одну строку. Главное не запутаться и не потерять нужный пакет.

(**Подсказка:** если вписать название пакета, повторяющегося во многих нужных нам пакетах **php8.2** и нажать двойной **Tab**, то он выдаст список всех возможных пакетов с таким названием, и уже можно написать все нужные)



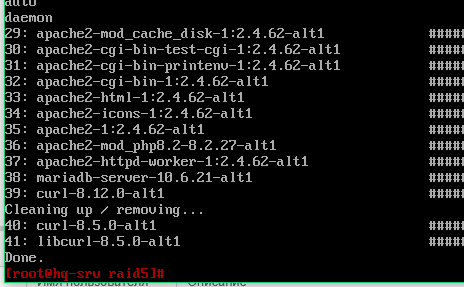
Поэтому можно разделить установку условно на 2 части.

1. Установка «уникальных пакетов»

## apt-get install apache2 php8.2 apache2-mod\_php8.2 mariadb-server



Подтверждаем установку и получаем

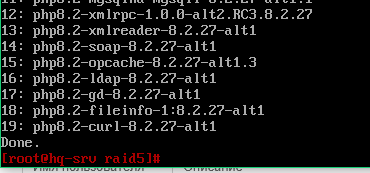


1. Установка оставшихся пакетов с повторяющимся началом.

## apt-get install php8.2-opcache php8.2-curl php8.2-gd php8.2-intl php8.2-mysqli php8.2-xml php8.2- xmlrpc php8.2-ldap php8.2-zip php8.2-soap php8.2-mbstring php8.2-json php8.2-xmlreader php8.2-fileinfo php8.2-sodium



Подтверждаем установку и ожидаем успешный вывод

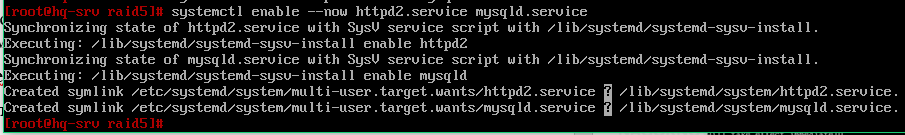


Включаем службы **httpd2** и **mysqld** для дальнейшей работы с ними следующей командой: Сперва запустим сервисы



Затем добавим их в атозагрузку (иногда достаточно просто поставить сервисы в автозагрузку, чтобы они и запустились, но иногда бывает, что запускается с ошибокой, поэтому сперва стартуем, потом добавляем)

## systemctl enable –now httpd2.service mysqld.service



Теперь настроим безопасный доступ к нашей будущей базе данных с помощью команды:

## mysql\_secure\_installation



Прожимаем просто **enter**, т.к. сейчас **root** без пароля:

## Enter

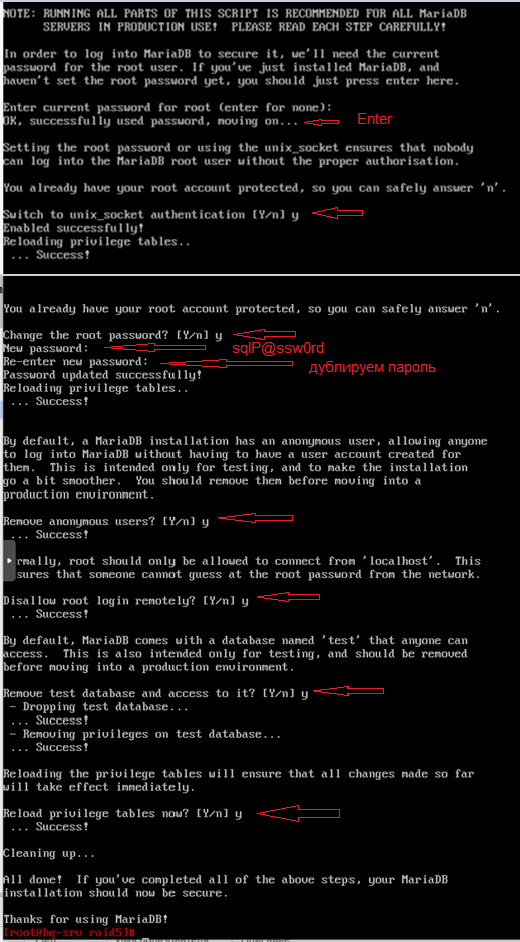
Прожимаем **y** для задания пароля:

## Y

Задаем пароль к нашему **root**: **sqlP@ssw0rd**

Далее нажимаем на всё **y**, как на скриншоте:

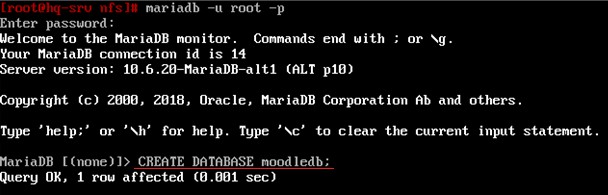
## Y



Теперь заходим в СУБД для создания и настройки базы данных:

## mariadb -u root –p



вводим пароль **sqlP@ssw0rd CREATE DATABASE moodledb;**

**CREATE USER moodle IDENTIFIED BY ‘P@ssw0rd’;**



**GRANT ALL PRIVILEGES ON moodledb.\* TO moodle;**



**FLUSH PRIVILEGES;**



**exit**

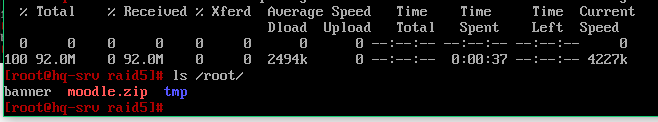


Теперь скачаем сам мудл стабильной версии:

**curl -L** [**https://github.com/moodle/moodle/archive/refs/tags/v4.5.0.zip**](https://github.com/moodle/moodle/archive/refs/tags/v4.5.0.zip) **> /root/moodle.zip**



Скачиваем его и проверяем, что он находится в указанной директории.



Разархивируем его в **/var/www/html/** для дальнейшей настройки:

**unzip /root/moodle.zip -d /var/www/html**



**mv /var/www/html/moodle-4.5.0/\* /var/www/html/**



**ls /var/www/html**



Создадим новый каталог **moodledata**, там будут храниться данные и изменим владельца на каталогах **html** и **moodledata**:

## mkdir /var/www/moodledata



**chown apache2:apache2 /var/www/html chown apache2:apache2 /var/www/moodledata**

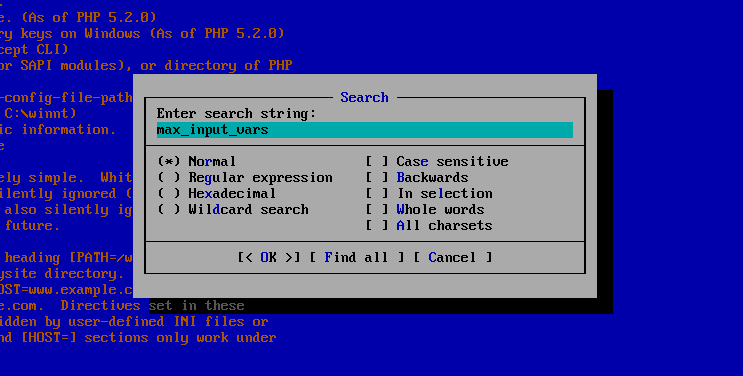
Поменяем значение параметра **max\_input\_vars** в файле **php.ini**:

## mcedit /etc/php/8.2/apache2-mod\_php/php.ini



Жмём **F7** для поиска нужной нам строки и пишем туда:

## max\_input\_vars



Наживаем Enter и записываем в найденной строке значение 5000, раскомментировав ее предварительно: **max\_input\_vars = 5000**

Удаляем стандартную страницу **apache**:

## cd /var/www/html ls

**rm index.html**



Перезапускаем службу **httpd2**:

## systemctl restart httpd2.service

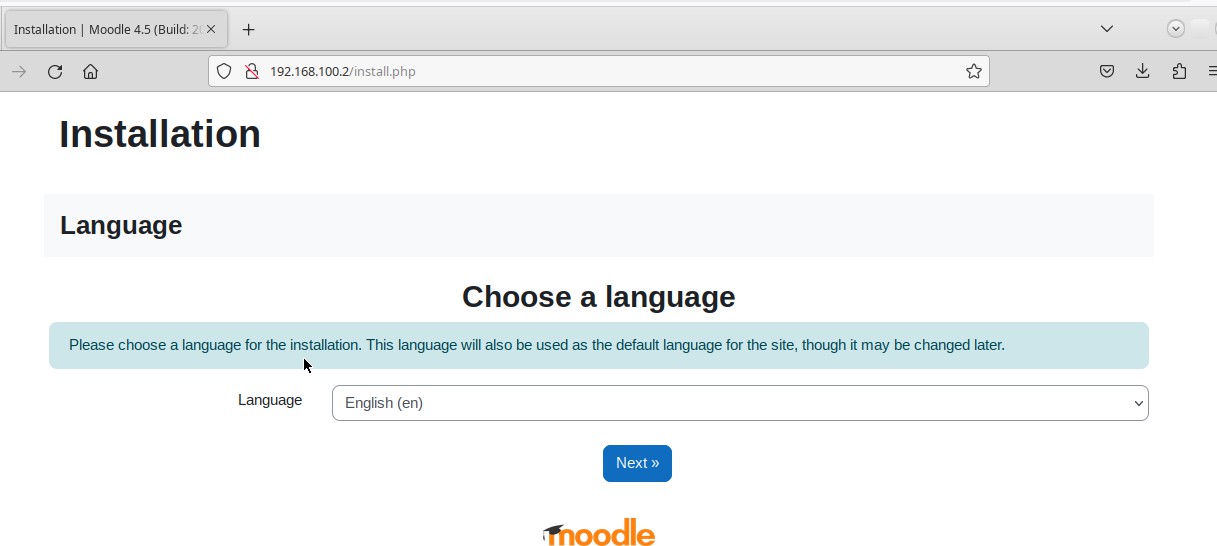


Теперь подключаемся с клиента HQ-CLI и начинаем настройку:

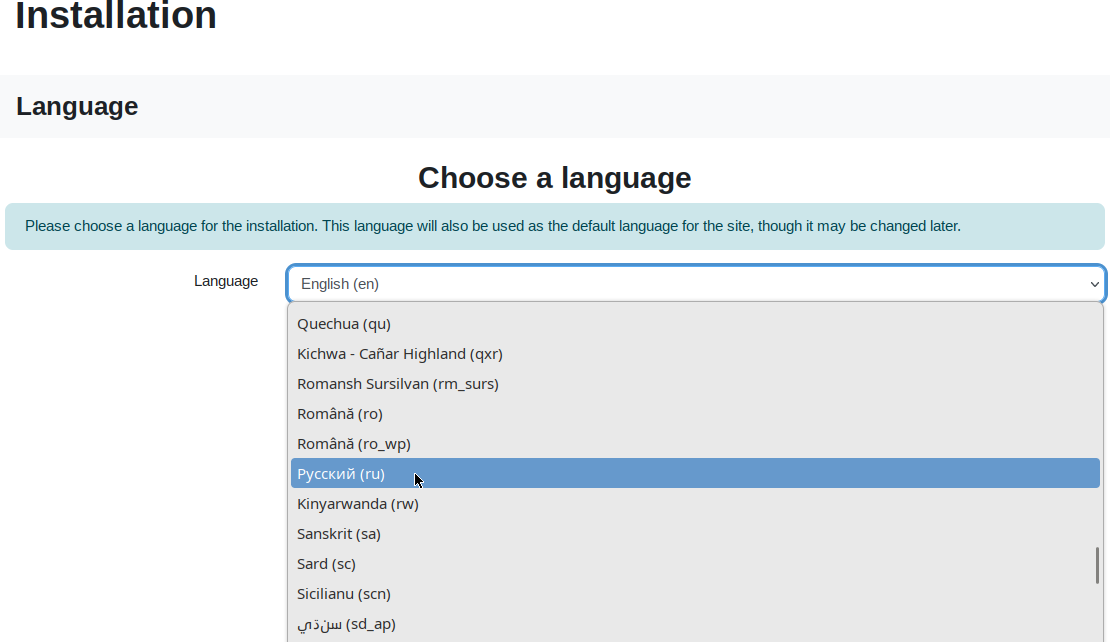
[**http://192.168.100.2/install.php**](http://192.168.100.2/install.php)



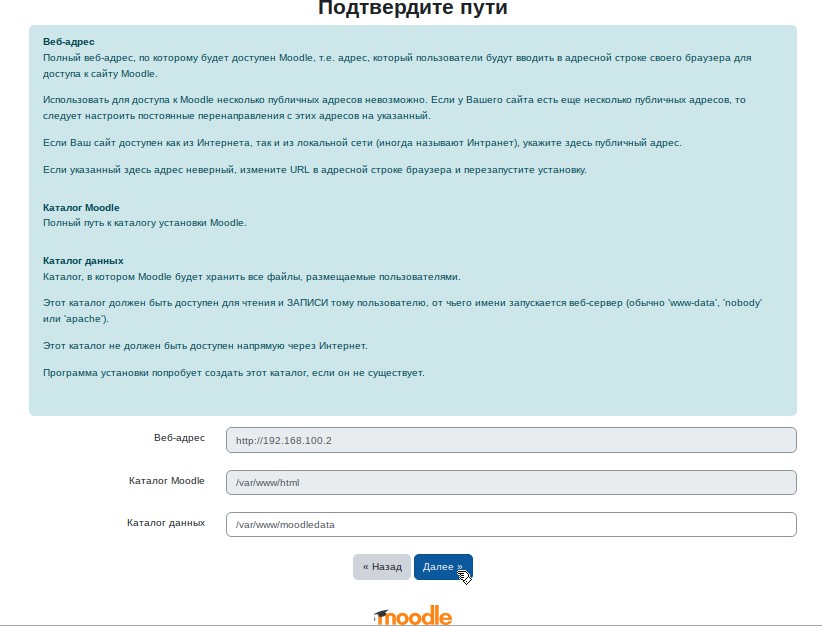
Получаем



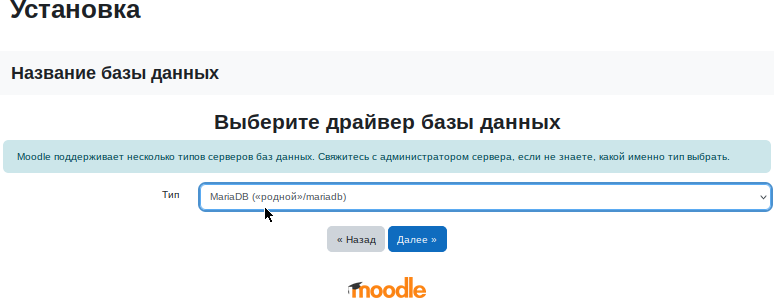
Выбираем язык – русский и жмем далее.



Жмём далее, т.к. каталог у нас уже создан:



Выбираем **MariaDB** в качестве драйвера базы данных:

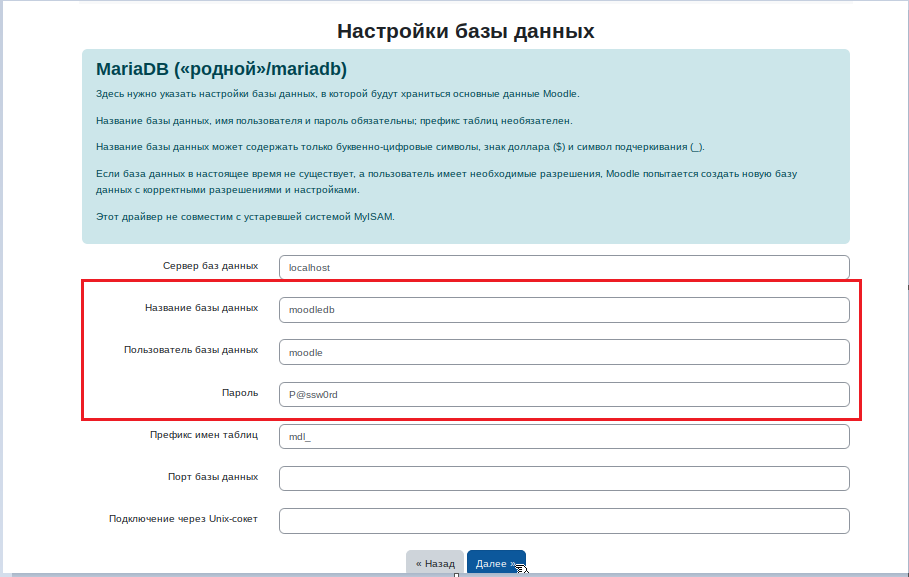


Введём нужные данные в следующие строки:

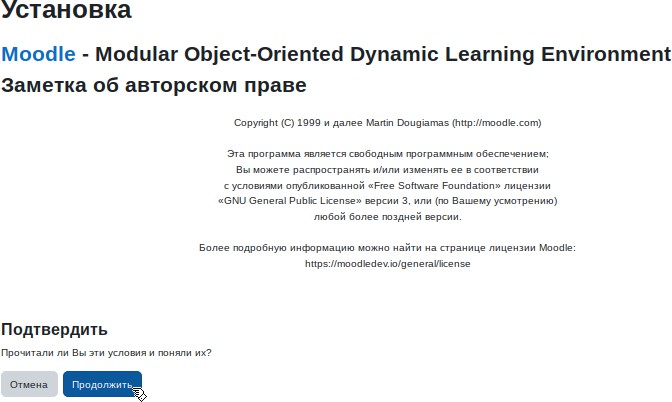
**Название базы данных: moodledb**

**Пользователь базы данных: moodle**

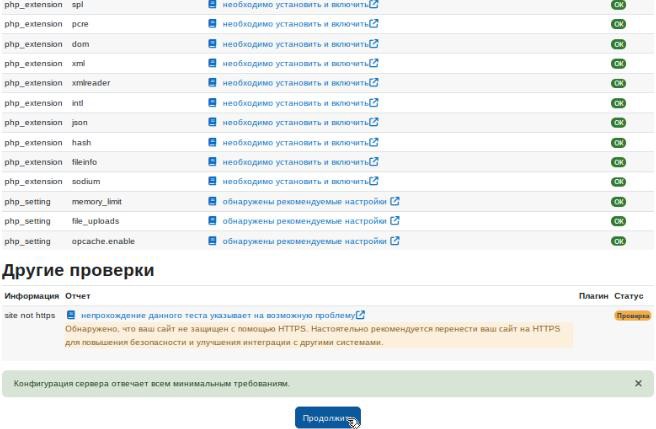
**Пароль: P@ssw0rd**



Нажимаем продолжить



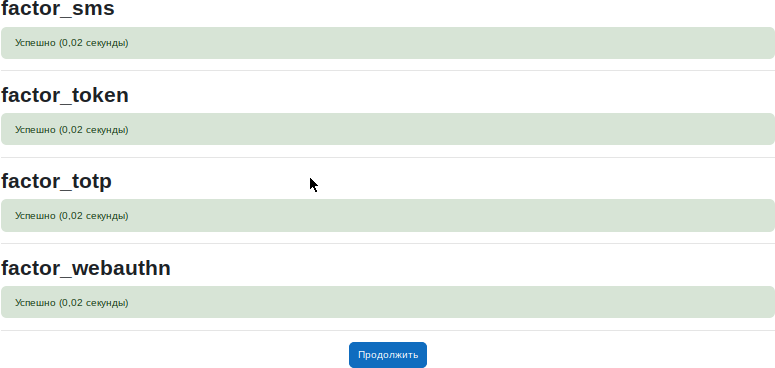
Просматриваем всё ли в статус “**OK**” или “**Проверка**” и прожимаем “**Продолжить**”:



Дальше пойдёт процесс установки в виде такого окна, процесс этот может быть долгим, не пугайтесь:



После установки видим, что всё прошло успешно и жмём “**Продолжить**”:



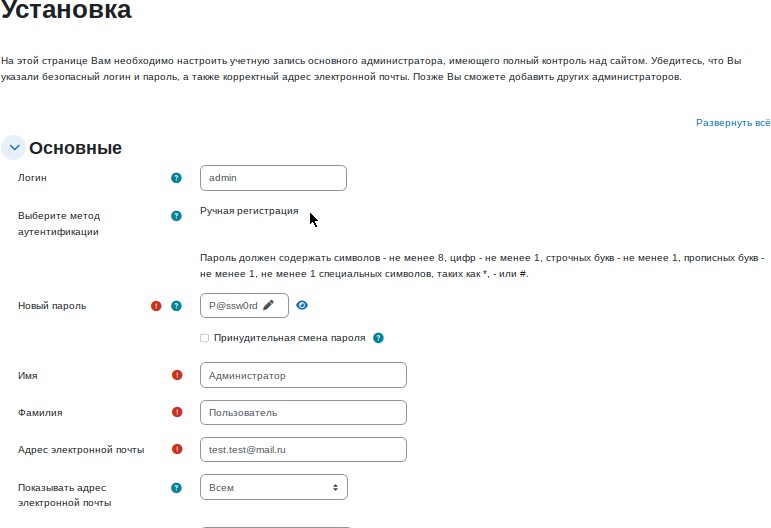
Далее заполняем обязательные поля для создания основного администратора:

## Логин: admin

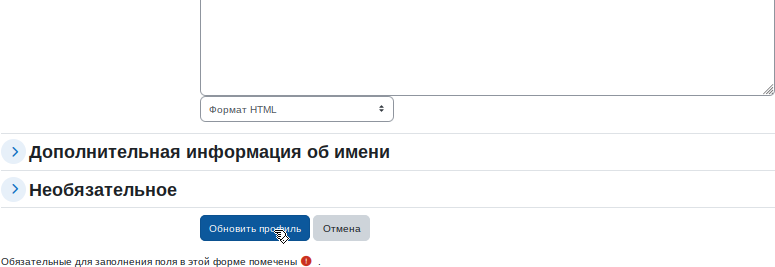
**Новый пароль: P@ssw0rd**

**Имя: Администратор (**можно любое**)**

**Фамилия: Пользователь (**можно любое**) Адрес электронной почты:** [**test.test@mail.ru**](mailto:test.test@mail.ru) **(**можно любое**)**

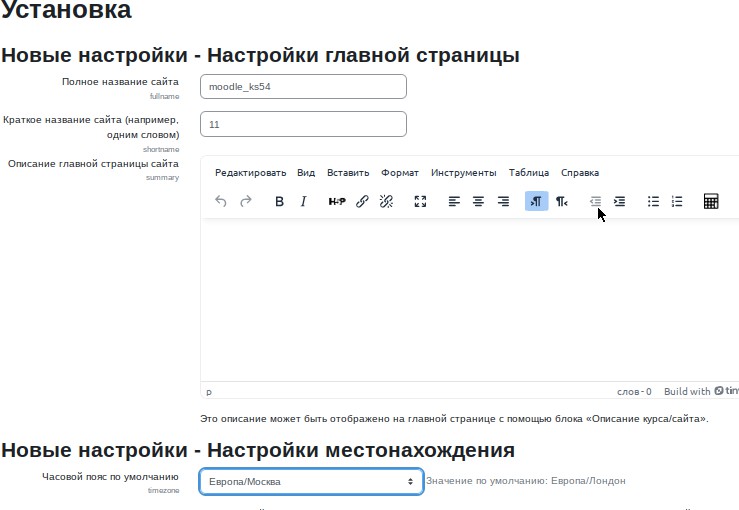


И нажимаем “**Обновить профиль**”:

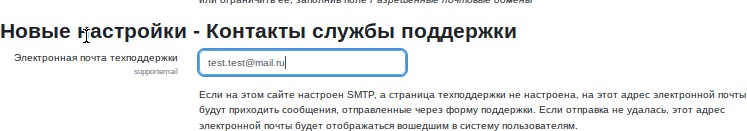


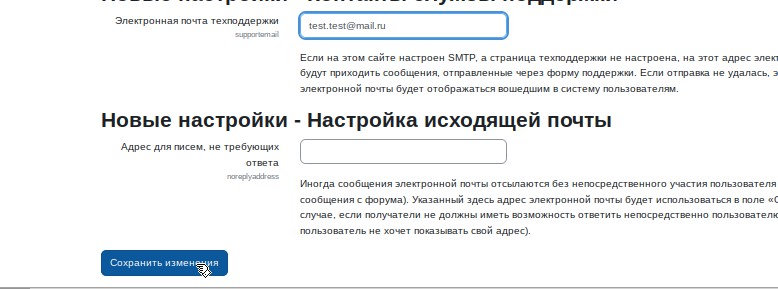
Теперь заполним ещё некоторые строки на следующем шаге:

**Полное название сайта: moodle\_ks54** (можно любое**) Краткое название сайта: 11** (согласно вашему рабочему месту)

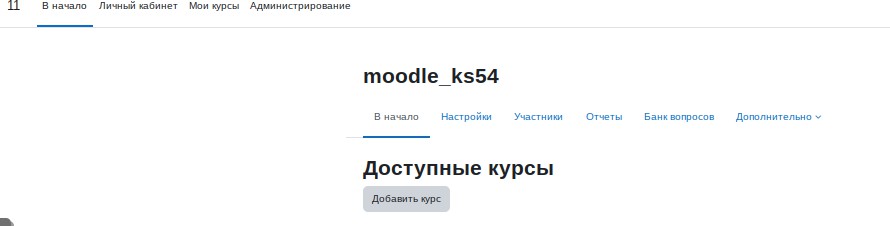
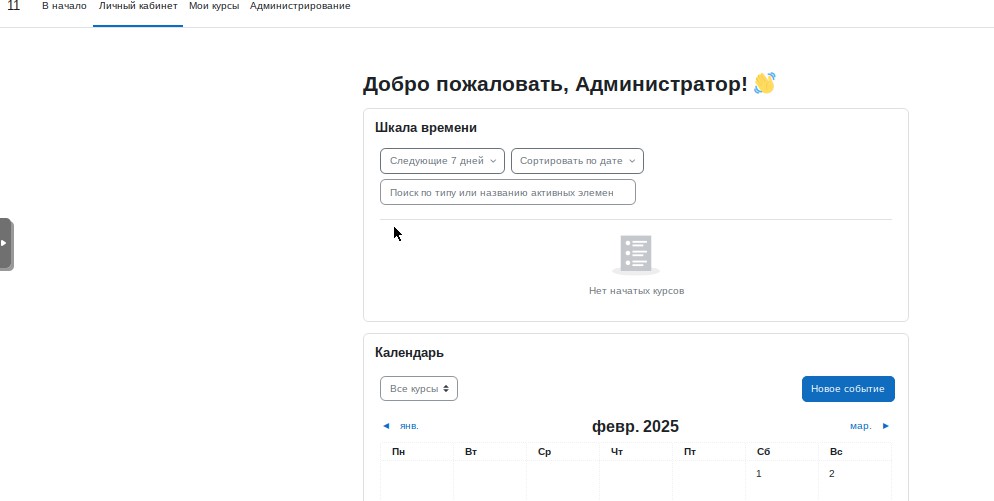
**Настройки местоположения: Европа/Москва** (согласно вашему региону)

**Контакты службы поддержки:** [**test.test@mail.ru**](mailto:test.test@mail.ru) **(**можно любое)



И жмём “**Сохранить изменения**” в конце страницы:

И после всего нас встречает рабочий сайт **moodle**, смотрим, что все наши указанные параметры отображаются:



На этом задании выполнено.

# НАСТРОЙТЕ ВЕБ-СЕРВЕР NGINX КАК ОБРАТНЫЙ ПРОКСИ-

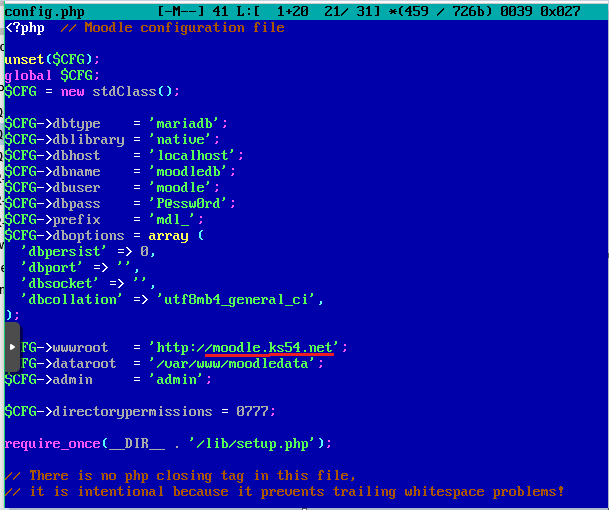
**СЕРВЕР НА HQ-RTR**

Поменяем значение **wwwroot** в конфигурации moodle на **HQ-SRV**: **mcedit /var/www/html/config.php**

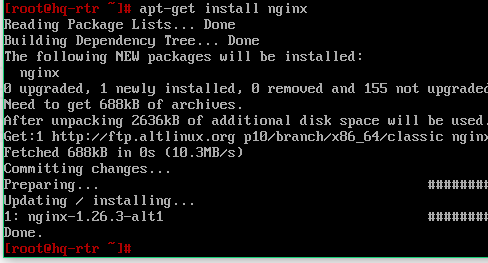


Заменим значение wwwroot с ip-адреса на доменное имя:

**$CFG->wwwroot = ‘**[**http://moodle.ks54.net**](http://moodle.ks54.net/)**’;**



Устанавливаем пакет **nginx** на **HQ-RTR** для дальнейшей настройки:



Создаём новый конфигурационный файл **proxy**: **mcedit /etc/nginx/sites-available.d/proxy.conf**



И заполняем его следующими строками:

**server {**

**listen 80;**

**server\_name moodle.au-team.irpo; location / {**

**proxy\_pass http://192.168.100.2:80; proxy\_set\_header Host $host; proxy\_set\_header X-Real-IP $remote\_addr;**

**proxy\_set\_header X-Forwarded-For $remote\_addr;**

**}**

**}**

**server {**

**listen 80;**

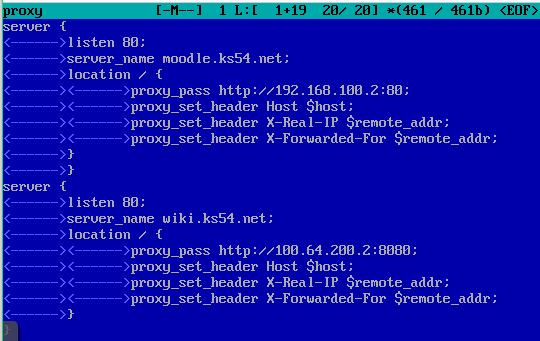
**server\_name wiki.au-team.irpo; location / {**

**proxy\_pass http://100.64.200.2:8080; proxy\_set\_header Host $host; proxy\_set\_header X-Real-IP $remote\_addr;**

**proxy\_set\_header X-Forwarded-For $remote\_addr;**

**}**

**}**



Удаляем конфигурацию (**default**), которую создал **nginx**, потом включаем созданную нами ранее (**proxy**), путём создания символической ссылки, а затем перезапускаем службу **nginx**:

## rm -rf /etc/nginx/sites-available.d/default.conf



На всякий случай проверяем файл default в папке sites-enabled.d. Если он имеется, то удаляем командной: **rm -rf /etc/nginx/sites-enabled.d/default.conf**

В противном случае идем дальше.

Создаем ссылку на наш файл proxy командой:

## ln -s /etc/nginx/sites-available.d/proxy /etc/nginx/sites-enabled.d/



Проверяем содержимое папки со ссылкой

## ls -la /etc/nginx/sites-enabled.d

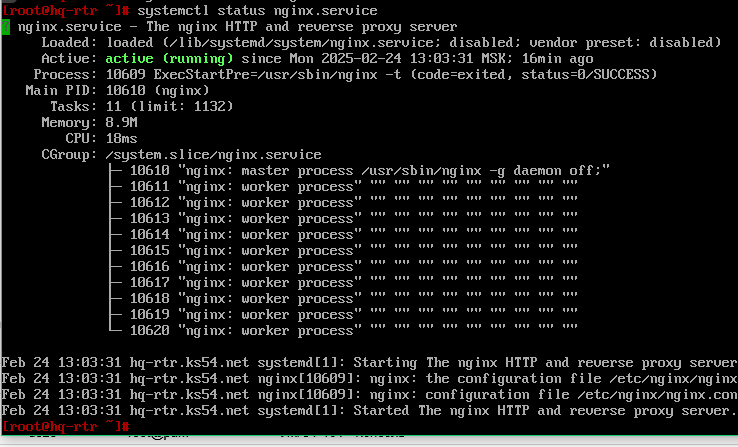


Перезапускаем сервис nginx

## systemctl restart nginx.service



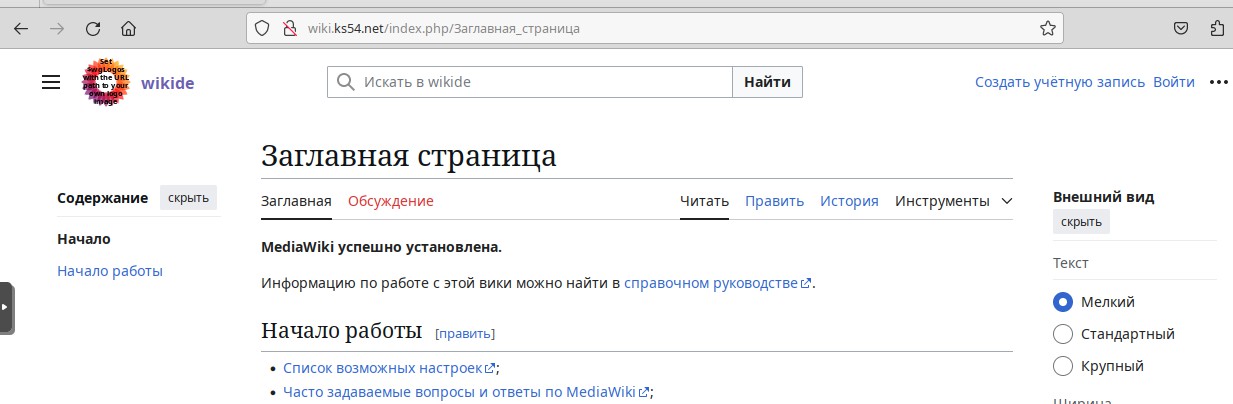
Для надежности проверим статус nginx командной: **systemctl status nginx.service**



Теперь протестируем работу обратного прокси через **HQ-CLI**

Зайдем на наши поднятые ранее сайты **moodle** и **wiki**

В адресной строке браузера впишем [**http://wiki.ks54.net**](http://wiki.ks54.net/)



Видим, что обратный прокси отрабатывает и открывает наш сайт без сертификата. (если браузер автоматически приписывает вместо http => http**s**, то тогда удалите s и нажмите Enter)

Аналогично проделываем с сайтом [**http://moodle.ks54.com**](http://moodle.ks54.com/)



Аналогично предыдущему можем исправить автоподписывание сертификата, удалив **s.**

Задание выполнено. Идем дальше.

# УДОБНЫМ СПОСОБОМ УСТАНОВИТЕ ПРИЛОЖЕНИЕ ЯНДЕКС БРАУЗЕРЕ ДЛЯ ОРГАНИЗАЦИЙ НА HQ-CLI

Установим Яндекс Браузер на **HQ-CLI** через терминал командами:

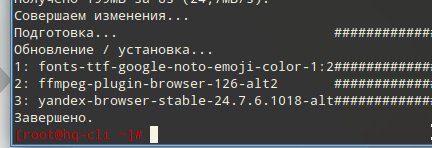
## apt-get update



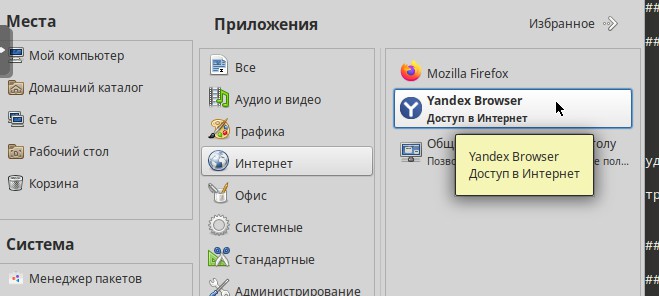
**apt-get install yandex-browser-stable**



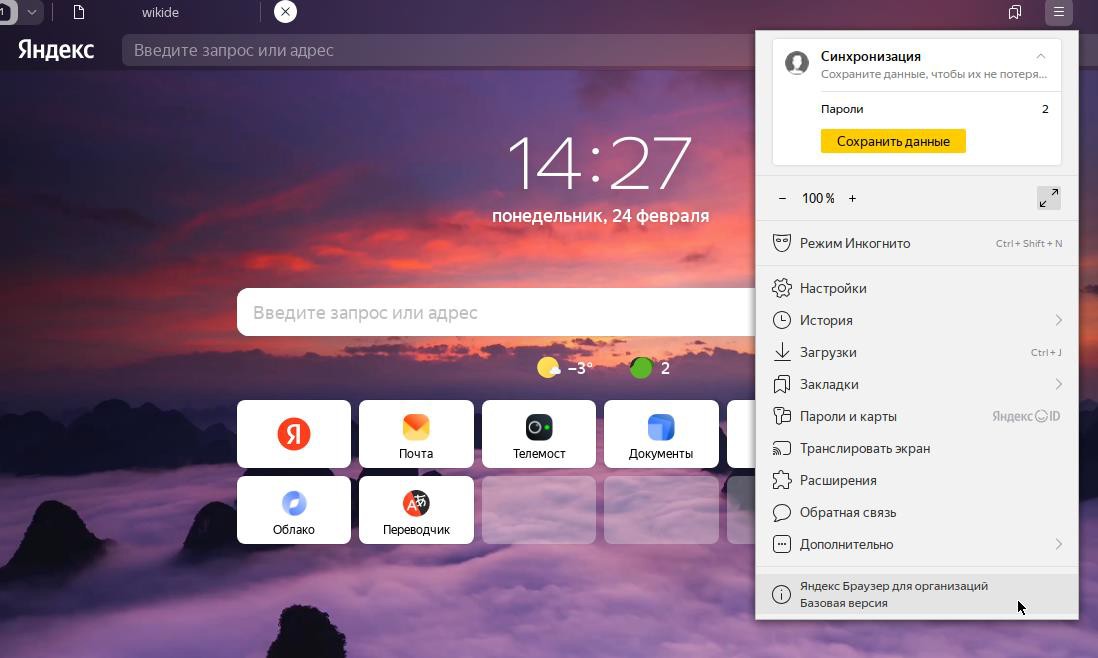
Подтверждаем установку и дожидаемся окончания



Проверяем появился ли в стартовом меню наш браузер



Запускаем и удостоверимся, что эта та версия, что нам необходима.

При нажатии “**☰**” мы увидим внизу надпись “**Яндекс.Браузер для организаций**”, это значит, что мы установили правильную версию браузера.

**ВСЕ ЗАДАНИЯ 1 МОДУЛЯ ДЭ ВЫПОЛНЕНЫ**