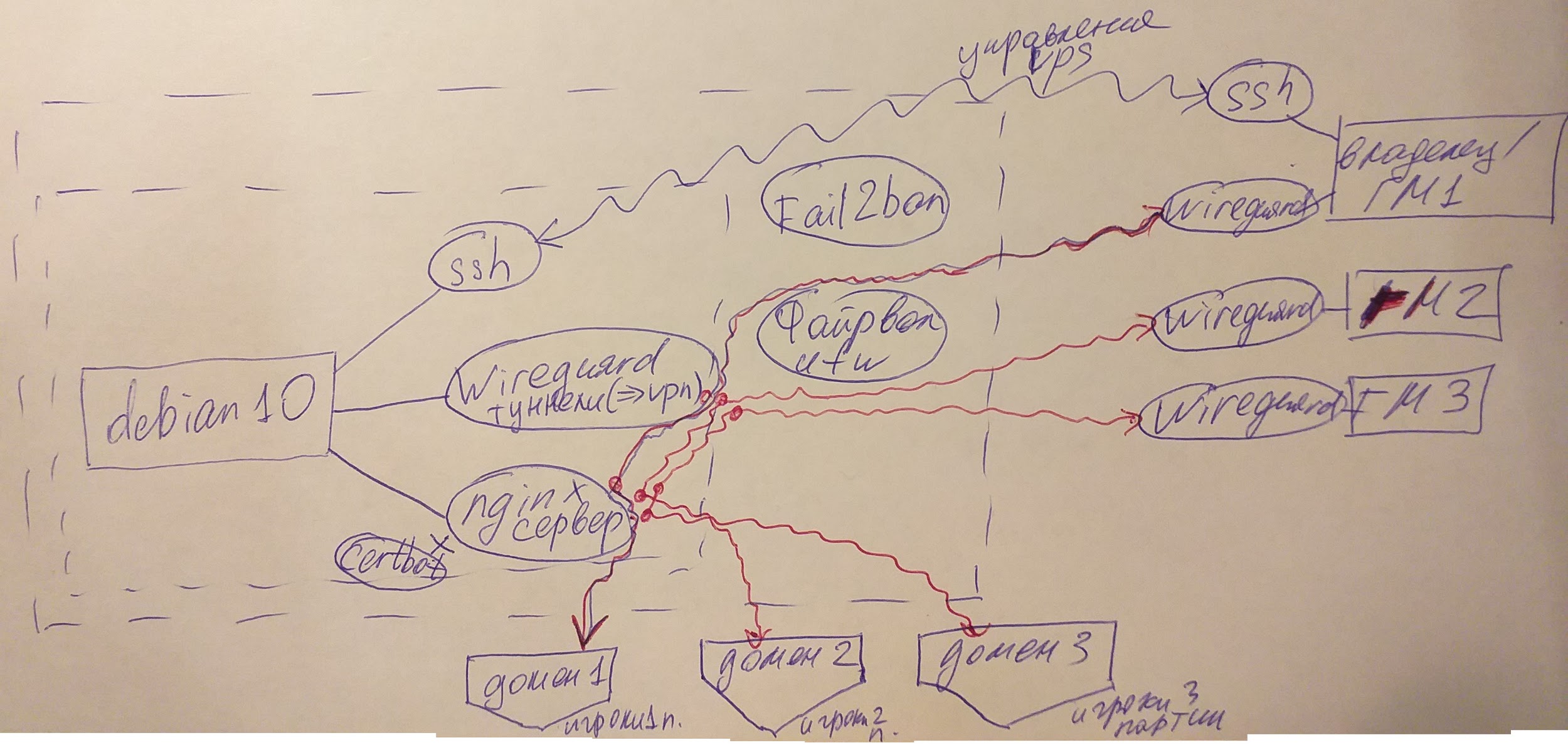
Настройка debian 10 для доступа игроков к Foundry через Wireguard или домен по https

Ниже один из вариантов действий для настройки сервера с debian 10 для доступа через ssh по публичному ключу, настройки файрвола ufw и fail2ban, настройки Wireguard.

Данные команды приведены для сервера на debian >=10, скорее всего, они также без изменений корректны для Ubuntu >=20 (>=18?), для другого дистрибутива нужно внести соответствующие корректировки. На стороне клиента использовалась Ubuntu 20, все соответствующие действия выделены синим цветом (ubuntu была развернута внутри виртуальной машины virtualbox, сам факт виртуализации на вид команд не влияет). Для другой системы нужно будет сделать соответствующие изменения. Коричневым выделил ссылки на источники. Жёлтым отметил то, что сделано непрямыми руками, нужно потом будет поправить, или же то, в чём я пока не разбирался, а поверил “на слово” (настройка /etc/sysctl.conf под Wireguard, ниже), позже проведу ревизию.

Кратко о том, что будет сделано ниже:



## 1. vps/vds-провайдер

ИМХО, но тем не менее:

а) Сервер (по опыту) имеет смысл искать в пределах того же города, где мастер выходит в интернет. Например, если мастер в СПб, а сервер в Москве, уже будут заметные задержки.

б) Также имеет смысл подбирать провайдера и тариф, чтобы иметь доступ через vnc, чтобы иметь больше контроля над машиной. С другой стороны, стабильность соединения и малые задержки важнее, в этом плане, хоть у Beget такой возможности нет, но соединение на тестах у них было стабильнее, чем у Sprinthost при прочих равных.

Провайдер предоставит пароль для пользователя **root** - **<root\_password>** (допустим, по почте).

Пользователь: root

Пароль: <root\_password>

Статья на хабре: <https://habr.com/ru/company/vdsina/blog/521388/>

Статья 2: <https://timeweb.com/ru/help/pages/viewpage.action?pageId=9241442>

Если нужно сразу сменить пароль для root:

**passwd root**

## 2. Подготовка сервера

Обновляем список пакетов из репозиториев, обновляем все установленные пакеты до актуальной версии.

**apt-get update**

**apt-get upgrade**

//Параллельно в /home/ должна была появиться папка <username>

## 3. Нерутовый юзер, на сервере

Поскольку у пользователя root абсолютные привилегии в системе, то, чтобы запретить ему удалённое администрирование, создадим нерутового пользователя, а для root отключим удалённое администрирование по ssh. Придумываем нерутового пользователя **<username>** и пароль для него **<user\_pass>**:

**adduser <username>**

При этом сразу попросит ввести **<user\_pass>** //позже можно сменить с помощью passwd <username>

Дальше попросит указать некоторую информацию для нового пользователя, можно просто несколько раз нажать Enter и затем Y.

Если sudo ещё не установлен, установим его:

**apt install sudo**

Теперь пользователя нужно добавить в группу, которая имеет право выполнять команды с повышением привилегий sudo:

**usermod -aG sudo <username>**

//Можно проверить, что пользователь был добавлен в группу sudo: "vi /etc/group" //закрыть Esc и ":q"

//Параллельно в /home/ должна была появиться папка <username>

## 4. Ключи вместо паролей SSH

//авторизация по ssh: <https://losst.ru/avtorizatsiya-po-klyuchu-ssh>

//памятка пользователю ssh: <https://habr.com/ru/post/122445/>

Брутфорс или утечка паролей — стандартный вектор атаки, так что аутентификацию по паролям в SSH (Secure Shell) лучше отключить, а вместо неё использовать аутентификацию по ключам. Используем клиент openssh. В качестве альтернативы есть также lsh и Dropbear.

Установка клиента OpenSSH на Ubuntu:

sudo apt install openssh-client

Установим на сервере:

**sudo apt install openssh-server**

Запустим демона SSH на сервере

**sudo systemctl start ssh**

Автоматический запуск демона при каждой загрузке:

**sudo systemctl enable ssh**

Создадим папку /home/<username>/.ssh/

**mkdir -p /home/<username>/.ssh/**

Создадим в ней файл authorized\_keys

**touch /home/<username>/.ssh/authorized\_keys**

Проверка существования файла:

**ls -lh /home/<username>/.ssh/authorized\_keys**

установим корректные разрешения для папки и файла:

**chmod 700 /home/<username>/.ssh && chmod 600 /home/<username>/.ssh/authorized\_keys**

Изменим владельца и группу для каталога /home/<username>/.ssh на нерутового пользователя:

**chown -R <username>:<username> /home/<username>/.ssh**

Теперь на стороне клиента,

**> Ubuntu 20:**

создадим файл с ключом <custom\_server\_key\_file> и отправим его на сервер, чтобы его содержимое было записано в файл .../authorized\_keys:

**sudo mkdir -p /root/.ssh/**

Генерирую ssh-ключ для сервера, пароль для ssh-ключа: <ssh\_key\_pass>

**sudo ssh-keygen**

В процессе попросит ввести путь до создаваемых приватного и публичного ключей:

>sudo ssh-keygen

Generating public/private rsa key pair.

Enter file in which to save the key (/root/.ssh/id\_rsa): **/root/.ssh/<custom\_server\_key\_file>**

Enter passphrase (empty for no passphrase): **<ssh\_key\_pass>**

Enter same passphrase again: **<ssh\_key\_pass>**

Your identification has been saved in /root/.ssh/id\_rsa/<custom\_server\_key\_file>

Your public key has been saved in /root/.ssh/id\_rsa/<custom\_server\_key\_file>.pub

<...>

По умолчанию ssh-keygen любит кидать файлы в папку /home/ (?), поэтому нужно забивать полный путь, и все поддиректории должны существовать.

Теперь перешлём публичный ключ серверу, чтобы он записал содержимое в файл /home/<username>/.ssh/authorized\_keys:

**sudo ssh-copy-id -i /root/.ssh/<custom\_server\_key\_file>.pub <username>@<server\_ip\_address>**

Можно проверить, что ключ “дошёл”, для этого на стороне сервера введём команду

**vi /home/<username>/.ssh/authorized\_keys**

В данный момент можно подключиться к серверу со стороны клиента по паролю, ключ не задействован (но лучше этого не делать):

**ssh <username>@<server\_ip\_address>**

Поэтому далее отключаем удаленный заход с помощью ssh из-под рута и использование паролей, для этого правим:

**vi /etc/ssh/sshd\_config**

Итого раскомментировать, где нужно, и установить (в VIM переход в режим редактирования нажатием кнопки Insert, вставка текста из буфера обмена Shift+Insert, удаление всего текста с данной позиции Esc + :.,$d + Enter, выход без записи :q!):

**PermitRootLogin no**

**PubkeyAuthentication yes**

**PasswordAuthentication no**

**ChallengeResponseAuthentication no**

**UsePAM no**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Перезапустим демон ssh, чтобы изменения вступили в силу:

**sudo systemctl restart ssh**

В данный момент со стороны клиента подключение к серверу возможно командой (при входе попросит ввести пароль **<ssh\_key\_pass>**):

**sudo ssh <username>@<server\_ip\_address> -i /root/.ssh/<custom\_server\_key\_file>**

## 5. Файрвол

Файрвол гарантирует, что на сервер пойдёт только тот трафик по тем портам, которые вы напрямую разрешили. Это защищает от эксплуатации портов, которые случайно включились с другими сервисами, то есть сильно уменьшает поверхность атаки.

В качестве файрвола воспользуемся ufw. Если он ещё не установлен, установим его:

**sudo apt install ufw**

Внесём SSH в список исключений файрвола (иначе после запуска файрвола мы не сможем подключиться к серверу):

**sudo ufw allow ssh**

Теперь запустим файрвол:

**sudo ufw enable**

Теперь можно проверить статус файрвола, введя:

**sudo ufw status**

ufw отобразит, что TCP-подключение по порту 22 разрешено (для ssh стандартный порт 22): 22/tcp - ALLOW - Anywhere

В случае, если понадобится перезапуск файрвола:

**sudo systemctl restart ufw**

Вероятно, теперь станет видно, как активно внешний мир пытается общаться с сервером, и начнут появляться сообщения вида “[UFW BLOCK]”.

Если возникнет потребность в более детальной настройке ufw, может пригодиться эта статья: <https://1linux.ru/old/fajrvoll-primery-s-iptables-ufw.html>.

## 6. Fail2ban

Сервис Fail2Ban анализирует логи на сервере и подсчитывает количество попыток доступа с каждого IP-адреса. В настройках указаны правила, сколько попыток доступа разрешено за определённый интервал — после чего данный IP-адрес блокируется на заданный отрезок времени. Например, разрешаем 5 неудачных попыток аутентификации по SSH в промежуток 2 часа, после чего блокируем данный IP-адрес на 12 часов.

Установим fail2ban:

**sudo apt install fail2ban**

Запустим и установим запуск при старте системы:

**sudo systemctl start fail2ban**

**sudo systemctl enable fail2ban**

В программе два конфигурационных файла: /etc/fail2ban/fail2ban.conf и /etc/fail2ban/jail.conf. Ограничения для бана указываются во втором файле.

Джейл для SSH включён по умолчанию с дефолтными настройками (5 попыток, интервал 10 минут, бан на 10 минут).

[DEFAULT]

ignorecommand =

bantime = 10m

findtime = 10m

maxretry = 5

Кроме SSH, Fail2Ban может защищать и другие сервисы на веб-сервере nginx или Apache.

## 7. Смена портов по умолчанию

Для ssh номер порта 22 задан по умолчанию. Чтобы уменьшить поверхность атаки (вопросы к <https://habr.com/ru/company/vdsina/blog/521388/>), изменим номер порта.

Номер порта можно настроить, изменив директиву Port 22 в файле конфигурации

**sudo vi /etc/ssh/sshd\_config**

Поставим

**Port <custom\_ssh\_port>**

Ещё раз перезапустим демон ssh, чтобы изменения вступили в силу:

**sudo systemctl restart ssh**

Теперь также нужно внести соответствующее изменение для ufw:

**sudo ufw allow <custom\_ssh\_port>/tcp**

Чтобы откатить обратно: sudo ufw delete allow <custom\_ssh\_port>/tcp

Теперь удалим правило на разрешение общения через TCP по 22 порту:

**sudo ufw delete allow 22/tcp**

Проверить, какие подключения в данный момент разрешены:

**sudo ufw status**

<https://www.cyberciti.biz/faq/howto-change-ssh-port-on-linux-or-unix-server/>

Теперь, чтобы удалённо подключиться по ssh, нужно ввести команду, с учётом нестандартного порта (при входе попросит ввести пароль <ssh\_key\_pass>):

**sudo ssh <username>@<server\_ip\_address> -i /root/.ssh/<custom\_server\_key\_file> -p <custom\_ssh\_port>**

-------Автоматические обновления безопасности

Нужно ли это вообще? Пока пропустил (<https://linux-audit.com/automatic-security-updates-with-dnf/>).

## 8. Wireguard

<https://www.wireguard.com/quickstart/>

<https://habr.com/ru/post/432686/>

<https://www.linux.org.ru/forum/admin/14321745>

### 8.1. Установка

Установим Wireguard на стороне сервера и на стороне клиента. В рамках сети, которую мы развернём, оба будут формально являться пирами, но, чтобы сеть “работала”, как минимум у одного из них (в данной случае у “сервера”) должен быть “белый” ip. Итак, на стороне сервера и на стороне клиента, добавим соответствующий репозиторий:

а) если debian > 10 или ubuntu >=20 (>=18?)

Прямыми руками - например, если добавить соответствующий репозиторий через add-apt-repository, но эта возможность до debian 11 работать не будет:

**apt-get install software-properties-common** //теперь доступна команда add-apt-repository

**sudo add-apt-repository ppa:wireguard/wireguard**

б) если debian ~10 (с оговорками 9).

Поэтому до debian 11.0 нужно использовать бэкпорты, в оф. репозиториях wireguard нет. Эта строка говорит пакет-менеджеру, что нужно использовать также репозиторий buster-backports для поиска и установки пакетов (а contrib / non-free - разделы в которых apt будет искать главный contributed и несвободный софт):

**sudo sh -c "echo 'deb http://deb.debian.org/debian buster-backports main contrib non-free' > /etc/apt/sources.list.d/buster-backports.list"**

Подключение репозитория для debian / ubuntu сделано. Теперь обновим пакеты:

**sudo apt-get update**

**sudo apt-get upgrade**

Установим Wireguard:

**sudo apt install wireguard**

Сделаем доступной wg-quick (понадобится дальше, <https://3dnews.ru/1002719/wireguard-vpn-setup>):

**sudo apt-get install linux-headers-$(uname -r)**

//если пакет сходу не находит (например, у меня так случилось в случае debian 9), то нужно найти и установить ближайший: sudo apt-cache search linux-headers (<https://stackoverflow.com/questions/22165929/install-linux-headers-on-debian-unable-to-locate-package>)

### 8.2. Настройка - перенаправление сетевых пакетов на стороне сервера

Теперь приступим к настройке Wireguard.

Для того, чтобы пакеты перенаправлялись туда, куда надо, нужно разрешить перенаправление сетевых пакетов на уровне ядра. Для этого откроем файл /etc/sysctl.conf и добавим в конец такие строки (<https://losst.ru/ustanovka-wireguard-v-ubuntu>):

**sudo vi /etc/sysctl.conf**

**net.ipv4.ip\_forward = 1**

**net.ipv6.conf.default.forwarding = 1**

**net.ipv6.conf.all.forwarding = 1**

**net.ipv4.conf.all.rp\_filter = 1**

**net.ipv4.conf.default.proxy\_arp = 0**

**net.ipv4.conf.default.send\_redirects = 1**

**net.ipv4.conf.all.send\_redirects = 0**

Затем необходимо выполнить команду sysctl -p, чтобы система перечитала конфигурацию:

**sudo sysctl -p**

### 8.3. Генерация пар публичный - приватный ключ

Сгенерируем все ключи для конфигураций Wireguard на стороне клиента. Генерация ключей сервера:

**wg genkey | sudo tee server\_private.key | wg pubkey | sudo tee server\_public.key**

просмотреть ключи:

**cat server\_private.key**

//8JNvz….

**cat server\_public.key**

//+tYEi...

Я просто сохранял себе все ключи по очереди в текстовый редактор, а потом вставлял в нужные места в файлах конфигурации, правя конфигурацию сервера через ssh.

Генерация ключей клиента:

**wg genkey | sudo tee client\_private.key | wg pubkey | sudo tee client\_public.key**

**cat client\_private.key**

**cat client\_public.key**

Пусть нам нужно создать сеть на N клиентов (что не совсем корректно, в рамках Wireguard все считаются пирами), я просто взял и сгенерировал себе пары публичный и приватный ключи для клиента N раз и сохранил себе в текстовый редактор / “блокнот” что-то типа:

=2

private: UKBSB...

public: a3L4e...

### 8.4. Запись настроек в файлы конфигураций

Теперь создадим конфигурационный файл сервера. Выберем порт для UDP (Wireguard использует UDP) - **<custom\_wireguard\_port>.**

Наш конфигурационный файл сервера (назовём его <wg\_0>) будет находится по пути /etc/wireguard/<wg\_0>.conf и будет выглядеть следующим образом (открыл конфигурацию через ssh и внёс туда текст типа):

**sudo vi /etc/wireguard/<wg\_0>.conf**

Текст конфигурации для сервера:

[Interface]

Address = 10.10.0.1/24

ListenPort = **<custom\_wireguard\_port>**

PrivateKey = **<server\_private>**

PostUp = iptables -A FORWARD -i **<wg\_0>** -j ACCEPT; iptables -t nat -A POSTROUTING -o enp0s8 -j MASQUERADE; ip6tables -A FORWARD -i **<wg\_0>** -j ACCEPT; ip6tables -t nat -A POSTROUTING -o enp0s8 -j MASQUERADE

PostDown = iptables -D FORWARD -i **<wg\_0>** -j ACCEPT; iptables -t nat -D POSTROUTING -o enp0s8 -j MASQUERADE; ip6tables -D FORWARD -i **<wg\_0>** -j ACCEPT; ip6tables -t nat -D POSTROUTING -o enp0s8 -j MASQUERADE

[Peer]

PublicKey = **<client\_public\_N>**

AllowedIPs = 10.10.0.<N+1>/32

[Peer]

PublicKey = **<client\_public\_N+1>**

AllowedIPs = 10.10.0.<N+2>/32

//Сладко: <custom\_wireguard\_port> также является wg\_0 в PostUp и PostDown. AllowedIPs отвечает за таблицу роутинга и, используя там 32, мы говорим, что на той стороне только один адрес. Address - это настройки для сетевого интерфейса, тут должно быть 24, т.к. у всей сети 24 маска.

Теперь создадим конфигурационные файлы для клиентов:

**vi client\_<N>.conf**

Текст конфигурации для N-го клиента:

[Interface]

PrivateKey = **<client\_private\_N>**

Address = 10.10.0.<N+1>/24

[Peer]

PublicKey = **<server\_public>**

Endpoint = **<server\_ip\_address>**:**<custom\_wireguard\_port>**

AllowedIPs = 10.10.0.0/24

PersistentKeepalive = 25

Когда опция PersistentKeepAlive включена, пакет keepalive отправляется на конечную точку сервера один раз в некотором интервале секунд. Разумный интервал, который работает с широким спектром межсетевых экранов, составляет 25 секунд. Установка его равным 0 отключает эту функцию, что является значением по умолчанию, поскольку большинству пользователей это не нужно, и она делает WireGuard немного более разговорчивым.

После того, как вы внесли все изменения, скопируйте файл на компьютер каждого клиента под именем /etc/wireguard/<wg\_N>.conf (вместо <wg\_N> нужно подставить своё название, его нужно будет использовать ниже при запуске командах).

### 8.5. Запуск интерфейсов Wireguard с нужной конфигурацией

Запустим интерфейсы на стороне сервера и клиента, команды в данном блоке нужно будет запустить и на той, и на другой стороне. Для простоты я использую название <wg\_0>, для каждого клиента будет соответствующее выбранное название <wg\_N>. Если клиент на Windows 10, то нужно просто добавить файл конфигурации в программе Wireguard (“Добавить туннель”).

Для запуска интерфейса используем такую команду:

**sudo wg-quick up <wg\_0>**

//При этом wg-quick аналогично набору следующих команд (пишет в консоли):

~$ sudo wg-quick up <wg\_0>

[#] ip link add <wg\_0> type wireguard

[#] wg setconf <wg\_0> /dev/fd/63

[#] ip -4 address add 10.0.0.2/24 dev <wg\_0>

[#] ip link set mtu 1420 up dev <wg\_0>

Аналогично можно использовать systemd:

**sudo systemctl start wg-quick@<wg\_0>**

С помощью systemd можно настроить автозагрузку интерфейса Wireguard с нужной конфигурацией:

**sudo systemctl enable wg-quick@<wg\_0>**

Настройка файрвола:

**sudo ufw allow <custom\_wireguard\_port>/udp**

**sudo ufw status**

//перезапуск wireguard с нужной конфигурацией: **sudo systemctl restart wg-quick@<wg\_0>**

//убрать wireguard из запуска при старте локально на debian-подобной системе: **sudo systemctl disable wg-quick@<wg\_0>**

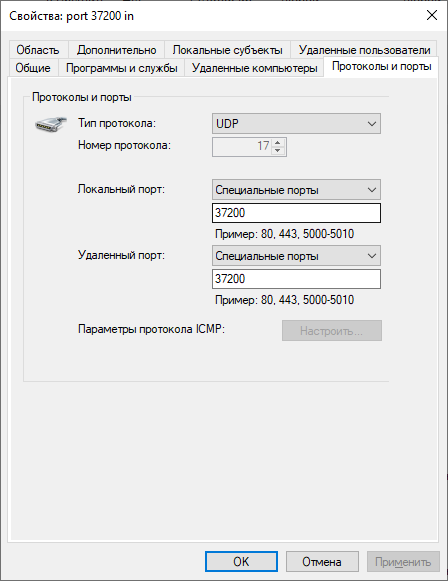
### 8.6. Открытие портов на машине с Foundryvtt

Здесь всё сильно зависит от используемой ОС, брандмауэров, файрволов, проч (как и вся клиентская часть, отмеченная синим). Wireguard работает с udp, поэтому нужно открыть соответствующие порты для получения и отправки трафика через UDP.

На windows 10 для встроенного брандмауэра (возможно, не достаточно):

Открыть UDP порт **<custom\_wireguard\_port>** на out и in в такой-то программе, в нулевом приближении просто открыть порт без привязки к программе.

Панель управления -> Брандмауэр Защитника Windows -> Дополнительные параметры -> Создать два правила для входящих и исходящих подключений.



В случае антивируса, нужно добавить виртуальную сеть, создаваемую Wireguard, в доверенные.

---------Переключение между конфигурациями на debian / ubuntu

Чтобы свободно переключаться между конфигурациями (разные доступные серверы или разные IP виртуальной локальной сети), вначале отключим/уберём из автозагрузки все запущенные конфигурации:

**sudo systemctl disable wg-quick@<wg…>**

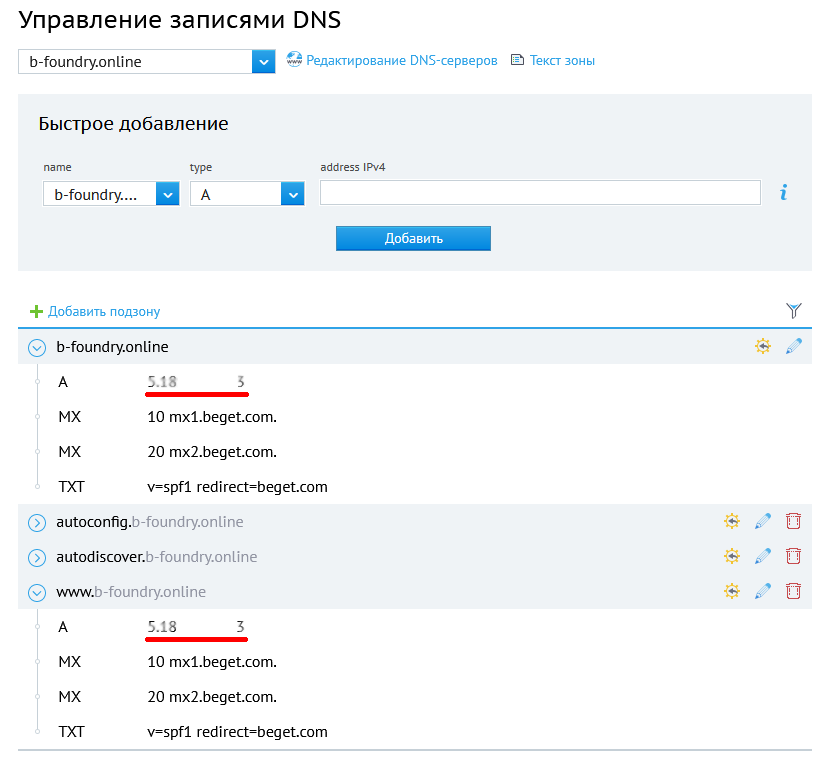
Затем просто поочередно запускать и останавливать нужные:

**sudo systemctl stop wg-quick@<wg\_K>**

**sudo systemctl start wg-quick@<wg\_K+1>**

## 9. Регистрация домена и привязка его к ip vps’а

Здесь ограничусь примером, после регистрации домена у того же провайдера, которому принадлежит vps, в случае beget нужно изменить A-запись для домена, в т.ч для подзоны www, по умолчанию у beget стоит заглушка, ведущая на IP страницы beget "Домен не прилинкован к директории на сервере!".



## 10. Развёртывание nginx сервера, как прокси, который предоставляет доступ к одному из пиров сети wireguard по порту 30000

### 10.1. Установка и настройка доступа через http

Ссылка: <https://foundryvtt.com/article/nginx/>

Установить nginx:

**sudo apt-get update**

**sudo apt-get install nginx**

В файле **/etc/nginx/sites-available/foundryvtt** прописал нужную конфигурацию, в разделе server\_name нужно указать зарегистрированный домен (выше), либо домен ~3 уровня, который выдан провайдером после начала аренды vps (типа “134-X-X-X.cloudvps.regruhosting.ru”). Далее - **<domain\_name>**.

**sudo vi /etc/nginx/sites-available/foundryvtt**

# Define Server

server {

# Enter your fully qualified domain name or leave blank

server\_name **<domain\_name> www.<domain\_name>**;

# Listen on port 80 without SSL certificates

listen 80;

# Sets the Max Upload size to 300 MB

client\_max\_body\_size 300M;

# Proxy Requests to Foundry VTT

location / {

# Set proxy headers

proxy\_set\_header Host $host;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header X-Forwarded-Proto $scheme;

# These are important to support WebSockets

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection "Upgrade";

# Make sure to set your Foundry VTT port number

# proxy\_pass <http://localhost:30000>;

# адрес Wireguard машины, на которой развёрнут и запущен Foundry

proxy\_pass **http://10.10.0.2:30000**;

}

}

//похоже, не обязательно (работает и без этого в данном контексте), на стороне клиента с Foundry:

В user data папке Foundryvtt в файле **{userData}/Config/options.json** изменил:

{

"port": 30000,

"upnp": true,

"fullscreen": false,

"hostname": "**<domain\_name>**",

"routePrefix": null,

"sslCert": null,

"sslKey": null,

"awsConfig": null,

"dataPath": "/home/hm/.local/share/FoundryVTT",

"proxySSL": false,

"proxyPort": 80,

"minifyStaticFiles": false,

"updateChannel": "release",

"language": "en.core",

"world": null

}

Дальше подключим новый сайт, создав символьную ссылку на конфигурацию в /etc/nginx/sites-enabled/

**sudo ln -s /etc/nginx/sites-available/foundryvtt /etc/nginx/sites-enabled/**

Проверка файла конфигурации

**sudo service nginx configtest**

Запуск nginx

**sudo service nginx start**

//sudo service nginx stop

//sudo service nginx restart

*(если удалять, то не так, нужно как-то аккуратнее!) В случае удаления конфигурации сервера вида* ***/etc/nginx/sites-available/<conf\_name>*** *нужно также удалить все ссылки на неё:*

***sudo rm* /etc/nginx/sites-available/foundryvtt**

***sudo rm /etc/nginx/sites-enabled/*foundryvtt**

***sudo systemctl restart nginx.service***

Также открыл порты у файрвола ufw:

Для http:

**sudo ufw allow 80/tcp**

Для https:

**sudo ufw allow 443/tcp**

**sudo ufw enable**

Теперь проверяем, что foundryvtt доступен через сеть интернет по протоколу http, если пользователь (ГМ) с запущенной foundry подключился к нужной конфигурации wireguard, а другой пользователь (игрок) зашёл по адресу "**<domain\_name>**" (при этом порт UDP **<custom\_wireguard\_port>** должен быть открыт на машине ГМа, - достаточно ли этого для работы?).

### 10.2. Установка бота Certbot для установки и автообновления ssl(tls?)-сертификатов от Let’s Encrypt

Далее нужно сделать доступ через https, для этого используем сертификат от Let’s Encrypt и его certbot: <https://certbot.eff.org/lets-encrypt/debianbuster-nginx>

Установим snapd:

**sudo apt update**

**sudo apt install snapd**

Убедимся, что версия snapd актуальна:

**sudo snap install core; sudo snap refresh core**

Удалим certbot-auto или все пакеты Certbot OS, если они были установлены менеджерами пакетов типо apt (т.к. для установки и обновления certbot рекомендуется использовать snapd):

**sudo apt-get remove certbot**

Установим Certbot:

**sudo snap install --classic certbot**

Убедимся, что команда certbot может быть запущена:

**sudo ln -s /snap/bin/certbot /usr/bin/certbot**

Установим сертификаты автоматически (попутно он изменит конфигурации тех “сайтов”, для которых мы выберем установку ssl-сертификатов):

**sudo certbot --nginx**

Описание “ручного” варианта установки по ссылке выше.

Проверим автоматическое обновление сертификатов:

**sudo certbot renew --dry-run**

### 10.3. Установка и настройка доступа через https

После автоматической установки Certbot он должен был изменить конфигурацию(и) типа **/etc/nginx/sites-available/foundryvtt**, привязав им сертификаты и сделав ряд других изменений, в том числе принудительное перенаправление http-запросов на https. Просмотреть совершенные изменения можно с помощью команды

**sudo vi /etc/nginx/sites-available/foundryvtt**

Если была выбрана ручная установка certbot, и эти изменения не были совершены (не проверял, может, и в этом случае правит конфигурации), то нужно сделать их вручную согласно <https://foundryvtt.com/article/nginx/>.

Теперь, если на машине ГМа запущен Foundryvtt, и сделано подключение к Wireguard, Foundry должен быть доступен в браузере по адресу **<domain\_name>** через протокол https.