Первоначальная настройка Ubuntu

- Первоначальная настройка сервера Ubuntu 24.04
- Установка и настройка Telegraf на Ubuntu
- Установка Docker и Docker Compose на Ubuntu
- Установка 3X-UI v2.8.11 на Ubuntu
- Установка Psiphon (psiphon-docker) на Ubuntu
- Установка Nebula 1.10.3 на Ubuntu

Первоначальная настройка сервера Ubuntu 24.04

Пошаговая настройка свежеустановленного сервера: обновление, рабочий пользователь, файрвол, защита SSH, Caddy как реверс-прокси.

> **Техника безопасности.** Шаги 5 и 8 могут отрезать вас от сервера при ошибке в конфиге. Всё время работы держите **вторую открытую SSH-сессию** и не закрывайте её, пока не убедитесь, что новая сессия открывается в третьем окне. Если доступа к консоли (IPMI, KVM, веб-консоль провайдера) нет — обеспечьте его до начала работы.

Порядок шагов важен: сначала создаём пользователя и убеждаемся, что он заходит по ключу, и только потом закрываем root.

Каждый блок кода ниже рассчитан на копирование целиком: это либо одна команда, либо цепочка через `&&`, которая отработает и одной строкой.

---

1. Обновление системы

```
sudo apt update && sudo apt upgrade -y
```

Если обновилось ядро, перезагружаемся, чтобы дальше работать на актуальной системе:

```
[ -f /var/run/reboot-required ] && sudo reboot
```

Опционально — автоматические обновления безопасности. Установка:

```
sudo apt install -y unattended-upgrades
```

Интерактивная настройка (задаст один вопрос, ответить «Да»):

```
sudo dpkg-reconfigure -plow unattended-upgrades
```

---

2. Пользователь mrleo1nid с sudo без пароля

Создаём пользователя. Команда интерактивная: спросит пароль и несколько необязательных полей, которые можно пропустить через Enter.

```
sudo adduser mrleo1nid
```

Добавляем в группу `sudo`:

```
sudo usermod -aG sudo mrleo1nid
```

Sudo без пароля

Правило пишем отдельным файлом в `/etc/sudoers.d/`, а не правкой `/etc/sudoers` — так изменение легко откатить и оно переживёт обновление пакета:

```
echo 'mrleo1nid ALL=(ALL) NOPASSWD:ALL' | sudo tee /etc/sudoers.d/mrleo1nid && sudo chmod 440 /etc/sudoers.d/mrleo1nid
```

Проверка синтаксиса — **обязательна**. Ошибка в sudoers ломает sudo целиком, и без root-доступа это уже не чинится:

```
sudo visudo -c -f /etc/sudoers.d/mrleo1nid
```

Ожидаемый вывод: `parsed OK`.

SSH-ключ

Если ключа ещё нет, создаём его **на локальной машине**:

```
ssh-keygen -t ed25519
```

Копируем на сервер, тоже с локальной машины:

```
ssh-copy-id mrleo1nid@CEPBEP
```

Проверяем в отдельном окне, что вход работает:

```
ssh mrleo1nid@CEPBEP
```

И что sudo не спрашивает пароль (должно вывести `root`):

```
sudo whoami
```

Не переходите к шагу 8, пока это не заработало.

---

3. Переключение на iptables legacy

Ubuntu 24.04 по умолчанию использует backend `nftables`. Скрипт бана из шага 7 работает через классический `iptables` + `ipset`, поэтому переключаем систему в legacy-режим — тогда UFW, Docker и скрипт будут писать правила в одну таблицу, а не в две параллельные.

```
sudo apt install -y iptables ipset
```

```
sudo update-alternatives --set iptables /usr/sbin/iptables-legacy && sudo update-alternatives --set ip6tables /usr/sbin/ip6tables-legacy && sudo update-alternatives --set arptables /usr/sbin/arptables-legacy && sudo update-alternatives --set ebtables /usr/sbin/ebtables-legacy
```

Проверка — в выводе должно быть `(legacy)`:

```
sudo iptables --version
```

Чтобы старые nft-правила не висели параллельно, перезагружаемся:

```
sudo reboot
```

> Если на сервере установлен Docker, после переключения его нужно перезапустить — backend определяется при старте демона. Перезагрузка делает это сама.

---

4. Fail2ban

Банит IP после нескольких неудачных попыток входа.

```
sudo apt install -y fail2ban
```

Свои настройки пишем в `jail.local` — `jail.conf` перезаписывается при обновлении пакета:

```
sudo nano /etc/fail2ban/jail.local
```

Содержимое:

```
[DEFAULT]
bantime  = 1h
findtime = 10m
maxretry = 5
# свои адреса, чтобы не забанить себя
ignoreip = 127.0.0.1/8 ::1 192.168.0.0/16 10.0.0.0/8

[sshd]
enabled = true
backend = systemd
```

`backend = systemd` заставляет читать журнал systemd вместо файла `/var/log/auth.log` — надёжнее, так как наличие этого файла зависит от того, установлен ли rsyslog.

Запускаем:

```
sudo systemctl enable --now fail2ban && sudo systemctl restart fail2ban
```

Проверяем, что jail поднялся:

```
sudo fail2ban-client status sshd
```

Разбанить адрес, если забанили себя:

```
sudo fail2ban-client set sshd unbanip 1.2.3.4
```

---

5. Файрвол

```
sudo apt install -y ufw
```

Сначала разрешаем SSH, потом включаем. Обратный порядок отрежет вас от сервера. Правила задаются одной цепочкой:

```
sudo ufw default deny incoming && sudo ufw default allow outgoing && sudo ufw allow OpenSSH &&
sudo ufw allow 80/tcp && sudo ufw allow 443/tcp
```

Только теперь включаем (команда спросит подтверждение — ответить `y`):

```
sudo ufw enable
```

Проверка:

```
sudo ufw status verbose
```

Порты 80 и 443 нужны Caddy: 80 — для ACME-проверки Let's Encrypt, без него сертификат не выпустится.

> **Docker и UFW.** Если на сервере есть контейнеры с проброшенными портами (`ports:` в compose), Docker пишет правила напрямую в iptables в обход UFW. Запрет в `ufw` такие порты не закроет. Ограничивать их нужно правилами в цепочке `DOCKER-USER` либо биндить публикацию на конкретный интерфейс: `'127.0.0.1:8080:80'` вместо `'8080:80'`.

---

6. Caddy

Реверс-прокси с автоматическим получением и продлением сертификатов Let's Encrypt.

Зависимости:

```
sudo apt install -y debian-keyring debian-archive-keyring apt-transport-https curl
```

Ключ репозитория:

```
curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/gpg.key' | sudo gpg --dearmor -o /usr/share/keyrings/caddy-stable-archive-keyring.gpg
```

Сам репозиторий:

```
curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/debian.deb.txt' | sudo tee /etc/apt/sources.list.d/caddy-stable.list
```

Установка:

```
sudo apt update && sudo apt install -y caddy
```

Перед настройкой убеждаемся, что A-запись домена указывает на этот сервер:

```
dig +short example.com
```

Конфиг:

```
sudo nano /etc/caddy/Caddyfile
```

```
example.com {
    reverse_proxy 127.0.0.1:8080
}
```

Проверяем синтаксис и применяем:

```
sudo caddy validate --config /etc/caddy/Caddyfile && sudo systemctl reload caddy
```

Смотрим логи — должна появиться запись об успешно полученном сертификате:

```
sudo journalctl -u caddy -f
```

Если Caddy стоит вторым в цепочке прокси

Когда перед ним есть ещё один Caddy или nginx, во внутреннем блоке достаточно передать протокол:

```
example.com {
    reverse_proxy http://10.10.1.129:8080 {
        header_up X-Forwarded-Proto https
    }
}
```

Дублировать `header_up` для `Host`, `X-Real-IP` и `X-Forwarded-For` не нужно: Caddy проставляет их сам, причём `X-Forwarded-For` дописывает к цепочке, а ручное `header_up` его затирает — приложение начнёт видеть IP прокси вместо реального клиента.

---

7. Блокировка IP по внешнему списку (ban.sh)

Скрипт из somebody-knocked-my-ssh скачивает список IP/CIDR, заливает его в `ipset` и вешает одно правило `DROP` в `INPUT` — вместо сотен отдельных правил на каждый адрес.

Скачиваем и делаем исполняемым:

```
sudo wget "https://raw.githubusercontent.com/jahlib/somebody-knocked-my-ssh/refs/heads/main/ban.sh" -O /usr/local/sbin/ban.sh && sudo chmod +x /usr/local/sbin/ban.sh
```

Перед первым запуском задаём переменную `URL` вверху скрипта — адрес текстового файла со списком (одна запись на строку, комментарии через `#`). Остальные параметры (`SET_NAME`, `REMOVE_ORPHANS`, `LOCK_FILE`) можно оставить по умолчанию:

```
sudo nano /usr/local/sbin/ban.sh
```

Первый запуск:

```
sudo bash /usr/local/sbin/ban.sh
```

Скрипт сам поставит `ipset`, если его нет, создаст набор, добавит правило и выведет строку-итог вида `added / skipped / invalid / removed / total`.

Автообновление по cron

```
sudo crontab -e
```

Добавить в конец:

```
@reboot      sleep 60 && bash /usr/local/sbin/ban.sh
0 */6 * * *  bash /usr/local/sbin/ban.sh
```

Строка `@reboot` важна: `ipset` живёт в памяти, и после перезагрузки набор пуст, а правило `iptables` без него ничего не блокирует. Задержка даёт подняться сети.

Проверка

Сколько адресов в наборе:

```
sudo ipset list blacklist | grep "Number of entries"
```

Стоит ли правило в цепочке:

```
sudo iptables -L INPUT -n --line-numbers | grep blacklist
```

Замечания

- Только IPv4. Для IPv6 скрипт ничего не делает.
- Правило добавляется в конец `INPUT`, а UFW ставит свои переходы в начало цепочки. Проверьте по `--line-numbers`, что порядок вас устраивает: если UFW разрешает порт раньше, чем сработает DROP, блокировка для этого порта не применится.
- В режиме зеркала (`REMOVE_ORPHANS=1`) пустой или битый удалённый список на очередной синхронизации разбанит всех. Следите за значением `removed=` в логе.
- Список по URL стоит держать неугадываемым: скрипт не умеет авторизацию, и любой, кто найдёт адрес, увидит его содержимое.

Как удалить

```
sudo iptables -D INPUT -m set --match-set blacklist src -j DROP -m comment --comment "ipset-blacklist-drop" && sudo ipset destroy blacklist
```

---

8. Закрытие root-доступа

> Выполнять **только после** того, как вход пользователем `mrleo1nid` по ключу и `sudo` без пароля проверены в отдельной сессии.

Отключаем вход root по SSH

В Ubuntu 24.04 в `/etc/ssh/sshd_config` есть директива `Include /etc/ssh/sshd_config.d/*.conf`, поэтому правки удобнее держать отдельным файлом:

```
sudo nano /etc/ssh/sshd_config.d/99-hardening.conf
```

Содержимое:

```
PermitRootLogin no
PasswordAuthentication no
KbdInteractiveAuthentication no
PubkeyAuthentication yes
```

Проверяем синтаксис и применяем одной командой — при ошибке в конфиге `sshd -t` выведет её и не даст перезапустить сломанный сервис:

```
sudo sshd -t && sudo systemctl restart ssh
```

В **новом окне** проверяем, что вход по-прежнему работает:

```
ssh mrleo1nid@CEPBEP
```

И что root теперь получает отказ:

```
ssh root@CEPBEP
```

> **Про смену порта SSH.** В 24.04 sshd запускается через сокет-активацию (`ssh.socket`), и директива `Port` в `sshd_config` игнорируется. Порт меняется через `sudo systemctl edit ssh.socket` с переопределением `ListenStream=`.

Убираем пароль root

```
sudo passwd -l root
```

`-l` блокирует пароль: локальный вход под root по паролю становится невозможен, `sudo` при этом работает как обычно.

> Команда `passwd -d root` **удаляет** пароль, а не блокирует. При некоторых конфигурациях PAM пустой пароль означает вход вообще без пароля — на консоли или при физическом доступе это дыра. Используйте `-l`, если нет конкретной причины поступить иначе.

Проверка — во втором поле должно быть `L`:

```
sudo passwd -S root
```

Итоговая проверка

Backend iptables (ожидается `legacy`):

```
sudo iptables --version
```

Файрвол активен, открыты 22/80/443:

```
sudo ufw status verbose
```

Fail2ban работает:

```
sudo fail2ban-client status sshd
```

Чёрный список загружен:

```
sudo ipset list blacklist | grep "Number of entries"
```

Caddy запущен:

```
sudo systemctl status caddy
```

Root заблокирован:

```
sudo passwd -S root
```

Фактически применённая конфигурация SSH — полезно, когда настройки разбросаны по нескольким файлам и непонятно, какая победила:

```
sudo sshd -T | grep -E 'permitrootlogin|passwordauthentication'
```

Что стоит сделать дополнительно

- Настроить бэкапы (`restic`, `borg`) с выгрузкой на другой хост.
- Поставить мониторинг доступности, чтобы узнавать о падении не от пользователей.
- Через неделю работы просмотреть `sudo journalctl -p err -b` — там обычно всплывает то, что настроено криво, но пока не мешает.

Установка и настройка Telegraf на Ubuntu

Пошаговая настройка Telegraf с отправкой метрик в InfluxDB: подключение репозитория, установка, конфиг, автозапуск, проверка и известный фикс ошибки с GPG-ключом.

Каждый блок кода ниже рассчитан на копирование целиком.

1. Подключение репозитория InfluxData

Ключ репозитория (актуальный, .gpg-формат вместо устаревшего apt-key):

```
sudo mkdir -p /etc/apt/keyrings && curl -fsSL https://repos.influxdata.com/influxdata-archive.key | sudo gpg --dearmor -o /etc/apt/keyrings/influxdata.gpg
```

Проверка, что ключ действительно скачался и не битый (необязательно, но полезно):

```
gpg --show-keys /etc/apt/keyrings/influxdata.gpg
```

В выводе должен быть отпечаток `DA61 C26A 0585 BD3B`.

Сам источник APT:

```
echo "deb [signed-by=/etc/apt/keyrings/influxdata.gpg] https://repos.influxdata.com/ubuntu stable main" | sudo tee /etc/apt/sources.list.d/influxdata.list
```

2. Установка

```
sudo apt update && sudo apt install telegraf influxdata-archive-keyring -y
```

Если при обновлении пакет спросит про изменённый конфиг — выбрать **install the package maintainer's version** (первый пункт). Свои настройки всё равно пишем заново на шаге 3, так что перезаписывать конфиг безопасно.

3. Конфигурация

Открываем конфиг:

```
sudo nano /etc/telegraf/telegraf.conf
```

Содержимое (заменить `hostname`, `urls`, `database`, `username`, `password` на свои):

```
[global_tags]

# Configuration for telegraf agent
[agent]
    interval = "10s"
    debug = false
    hostname = "server1.domain.sas"
    round_interval = true
    flush_interval = "10s"
    flush_jitter = "0s"
    collection_jitter = "0s"
    metric_batch_size = 1000
    metric_buffer_limit = 40000
    quiet = false
    logfile = ""
    omit_hostname = false

###############################################################################
#                                  OUTPUTS                                    #
###############################################################################

[[outputs.influxdb]]
    urls = [ "https://influx.domain.sas" ]
    database = "db"
    username = "user"
    password = "password"
```

```
###############################################################################
#                                  INPUTS                                     #
###############################################################################

[[inputs.cpu]]
[[inputs.disk]]
[[inputs.diskio]]
[[inputs.mem]]
[[inputs.system]]
[[inputs.swap]]
[[inputs.net]]
[[inputs.nstat]]
[[inputs.netstat]]
[[inputs.processes]]
[[inputs.kernel]]

###############################################################################
#                                  PROCESSORS                                 #
###############################################################################



###############################################################################
#                                  AGGREGATORS                                #
###############################################################################
```

Проверка синтаксиса перед перезапуском — обязательна, чтобы не запускать сервис на битом конфиге:

```
telegraf --config /etc/telegraf/telegraf.conf --test
```

4. Права sudo для сбора метрик Fail2ban

Нужно, только если в конфиге есть `[[inputs.exec]]`, дёргающий `fail2ban-client status` — по умолчанию у пользователя `telegraf` прав на это нет. Как и в примере с sudoers для рабочего пользователя, правило пишем отдельным файлом в `/etc/sudoers.d/`, а не правкой `/etc/sudoers`:

```
sudo tee /etc/sudoers.d/telegraf >/dev/null <<'EOF'
telegraf ALL=(root) NOPASSWD: /usr/bin/fail2ban-client status
telegraf ALL=(root) NOPASSWD: /usr/bin/fail2ban-client status *
Defaults!/usr/bin/fail2ban-client !logfile, !syslog, !pam_session
EOF

sudo chmod 440 /etc/sudoers.d/telegraf
sudo visudo -c -f /etc/sudoers.d/telegraf
```

Проверка синтаксиса обязательна — ошибка в sudoers ломает sudo целиком. Ожидаемый вывод: `parsed OK`.

5. Запуск и автозапуск

```
sudo systemctl restart telegraf && sudo systemctl enable telegraf
```

6. Проверка

Статус сервиса:

```
sudo systemctl status telegraf --no-pager
```

Логи в реальном времени — там же будет видно, если InfluxDB отвергает метрики (неверный пароль, недоступный хост и т. п.):

```
sudo journalctl -u telegraf -f
```

7. Фикс ошибки с GPG-ключом

Если `apt update` выдаёт:

```
The following signatures couldn't be verified because the public key is not available:
NO_PUBKEY DA61C26A0585BD3B
```

— значит InfluxData сменили ключ подписи, а на сервере лежит старый. Решение:

```
sudo rm -f /etc/apt/keyrings/influxdata-archive_compat.key && sudo mkdir -p /etc/apt/keyrings && curl -fsSL https://repos.influxdata.com/influxdata-archive.key | sudo gpg --dearmor -o /etc/apt/keyrings/influxdata.gpg
```

Источник APT (перезаписываем на всякий случай, вдруг там осталась старая ссылка на `influxdata-archive_compat.key`):

```
echo "deb [signed-by=/etc/apt/keyrings/influxdata.gpg] https://repos.influxdata.com/ubuntu stable main" | sudo tee /etc/apt/sources.list.d/influxdata.list
```

Установка:

```
sudo apt update && sudo apt install telegraf influxdata-archive-keyring -y
```

Что стоит сделать дополнительно

- Проверить, что порт InfluxDB (обычно 8086 для v1 API или 443, если он за Caddy/nginx) открыт в файрволе только там, где нужно.
- Ограничить набор `[[inputs.*]]` под задачу — включённые по умолчанию `nstat`/`netstat`/`processes` не всегда нужны и добавляют лишние метрики.
- Через день работы проверить объём отправляемых данных (`metric_batch_size` / `flush_interval`), чтобы не упереться в лимиты InfluxDB.

Установка Docker и Docker Compose на Ubuntu

Установка через официальный репозиторий Docker: движок, CLI и плагин `docker compose` (отдельный `docker-compose` v1 не нужен — compose теперь встроен как подкоманда).

Каждый блок кода ниже рассчитан на копирование целиком.

1. Удаление старых версий

Если Docker когда-либо ставился из репозитория Ubuntu, старые пакеты конфликтуют с официальными — удаляем:

```
sudo apt-get remove docker.io docker-doc docker-compose docker-compose-v2 podman-docker containerd runc -y
```

Если Docker раньше не ставился, команда просто ничего не найдёт и завершится без ошибок.

2. Подключение репозитория Docker

Зависимости и ключ репозитория:

```
sudo apt-get update && sudo apt-get install ca-certificates curl -y && sudo install -m 0755 -d /etc/apt/keyrings && sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc && sudo chmod a+r /etc/apt/keyrings/docker.asc
```

Сам источник APT:

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

3. Установка

```
sudo apt-get update && sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin -y
```

4. Автозапуск при загрузке

```
sudo systemctl enable docker.service && sudo systemctl enable containerd.service
```

5. Проверка

Тестовый контейнер — должен отработать без sudo-ошибок и вывести приветственное сообщение:

```
sudo docker run hello-world
```

Версия движка и compose:

```
docker --version && docker compose version
```

6. Запуск docker без sudo (опционально)

Добавляем текущего пользователя в группу `docker`:

```
sudo usermod -aG docker $USER
```

Группа применится после повторного входа. Проверить в текущей сессии, не перелогиниваясь:

```
newgrp docker
```

Членство в группе `docker` равносильно root-доступу на хосте: контейнер, запущенный с `-v /:/host`, даёт полный доступ к файловой системе. Добавлять пользователя в группу стоит только тем, кому реально нужен `docker` без `sudo`.

7. Docker и UFW

Если на сервере уже настроен файрвол, учтите: Docker пишет свои правила напрямую в iptables в обход UFW. Любой проброшенный порт (`ports:` в compose) окажется доступен извне, даже если `sudo ufw status` его не показывает как открытый.

Ограничить такие порты можно двумя способами:

- правилами в цепочке `DOCKER-USER`;
- либо биндингом публикации на конкретный интерфейс — `'127.0.0.1:8080:80'` вместо `'8080:80'`, если порт нужен только локально (например, за реверс-прокси).

Что стоит сделать дополнительно

- Настроить ротацию логов контейнеров (`log-driver` / `max-size` в `/etc/docker/daemon.json`) — по умолчанию логи ничем не ограничены и могут забить диск.
- Если контейнеры публикуют порты, сразу решить вопрос из шага 7 — до того, как сервис окажется доступен снаружи.
- Проверить `docker system df` через какое-то время работы — неиспользуемые образы и volume'ы накапливаются.

Установка 3X-UI v2.8.11 на Ubuntu

Панель управления Xray (VLESS/VMess/Trojan/Shadowsocks и др.) — MHSanaei/3x-ui. Установка через официальный скрипт с жёсткой фиксацией версии, чтобы скрипт не подтянул `latest`.

Каждый блок кода ниже рассчитан на копирование целиком.

1. Зависимости

Скрипту нужен `curl`:

```
sudo apt update && sudo apt install curl -y
```

2. Установка панели (фиксированная версия v2.8.11)

Версию нужно передать дважды: один раз в переменную `VERSION` (подставляется в URL самого скрипта), второй раз — аргументом скрипту (иначе он всё равно спросит версию или возьмёт последнюю):

```
VERSION=v2.8.11 && bash <(curl -Ls "https://raw.githubusercontent.com/mhsanaei/3x-ui/$VERSION/install.sh") $VERSION
```

Скрипт интерактивный и спросит:

- имя пользователя и пароль панели (не оставлять по умолчанию `admin`/`admin`);
- порт панели (лучше нестандартный, не 80/443 — их обычно занимает реверс-прокси);
- webBasePath — случайный путь вида `/a1b2c3d4/`, без него панель на голом порту легко просканировать.

Установщик сам поставит `x-ui` как systemd-сервис и включит автозапуск.

3. Файрвол

Открываем порт панели, который указали на шаге 2 (пример для порта `2053`; если ставили из-под UFW из предыдущего гайда по серверу — SSH/80/443 уже открыты):

```
sudo ufw allow 2053/tcp
```

Порты самих inbound'ов (VLESS/Trojan и т. п.) открываются отдельно, когда добавляете их в панели — там же видно, какой порт и протокол занят.

4. Проверка

Статус сервиса:

```
sudo systemctl status x-ui --no-pager
```

Установленная версия и путь к панели — команда `x-ui` открывает интерактивное меню:

```
x-ui
```

В меню пункт со статусом покажет версию, порт и webBasePath — свериться, что совпадает с тем, что вводили на шаге 2.

Логи, если панель не поднимается:

```
sudo journalctl -u x-ui -f
```

5. Управление сервисом

```
sudo systemctl restart x-ui   # перезапуск
sudo systemctl stop x-ui      # остановка
sudo systemctl enable x-ui    # автозапуск (обычно уже включён установщиком)
```

6. TLS для панели (опционально)

Если панель смотрит наружу напрямую (без Caddy/nginx перед ней), в меню `x-ui` есть пункт для выпуска сертификата Let's Encrypt на домен — доступен через тот же интерактивный `x-ui`, раздел SSL. Домен должен A-записью указывать на сервер до запроса сертификата, как и в шаге с Caddy из общего гайда по серверу.

7. Обновление и удаление

Обновление до другой версии — та же команда установки с новым тегом:

```
VERSION=vX.Y.Z && bash <(curl -Ls "https://raw.githubusercontent.com/mhsanaei/3x-ui/$VERSION/install.sh") $VERSION
```

Удаление — через меню:

```
x-ui
# выбрать пункт uninstall
```

Что стоит сделать дополнительно

- Сменить дефолтный порт панели и webBasePath сразу после установки, если пропустили на шаге 2 (доступно в меню `x-ui`).
- Не открывать порт панели на 80/443, если эти порты уже заняты Caddy/nginx — конфликт портов уронит один из сервисов.
- Настроить бэкап `/etc/x-ui/x-ui.db` — в нём вся конфигурация inbound'ов и клиентов.
- Официальный install-скрипт вызывает не флаг `-webBasePath` в кавычках без слешей, а напрямую бинарник, лежащий в `/usr/local/x-ui/`:

  bash

  ```
  /usr/local/x-ui/x-ui setting -webBasePath ваш-путь
  ```

Установка Psiphon (psiphon-docker) на Ubuntu

Прокси-туннель для обхода блокировок на базе psiphon-tunnel-core — swarupsengupta2007/psiphon-docker. Поднимает HTTP- и SOCKS5-прокси с выбором региона выхода.

Каждый блок кода ниже рассчитан на копирование целиком.

> Предполагается, что Docker и Docker Compose plugin уже установлены (см. страницу «Установка Docker и Docker Compose на Ubuntu»).

1. Подготовка каталога

```
sudo mkdir -p /opt/psiphon/config && sudo chown -R 1000:1000 /opt/psiphon && cd /opt/psiphon
```

`/opt/psiphon` выбран как каталог для стороннего софта по FHS — не привязан к домашней директории конкретного пользователя. Каталог `config` монтируется в контейнер и хранит `psiphon.config` — должен быть доступен на запись пользователю с указанными ниже `PUID`/`PGID` (в примере — `1000:1000`, при других значениях `chown` нужно поправить соответственно).

2. docker-compose.yml

```
nano docker-compose.yml
```

Содержимое:

```
services:
  psiphon:
    image: swarupsengupta2007/psiphon:latest
```

```
    container_name: psiphon
    environment:
      - PUID=1000
      - PGID=1000
      - HTTP_PORT=8080
      - SOCKS_PORT=1080
      - DEVICE_REGION=RU
      - EGRESS_REGION=SG
    volumes:
      - ./config:/config
    ports:
      - 127.0.0.1:1080:1080
      - 127.0.0.1:8080:8080
    restart: unless-stopped
```

> **Docker и UFW.** Как отмечено на странице первоначальной настройки сервера, Docker прописывает правила `ports:` напрямую в iptables в обход UFW. Публикация на `127.0.0.1:...` вместо `0.0.0.0` (голого порта) не даёт достучаться до прокси снаружи — открывайте наружу только если это осознанное решение.

Переменные окружения

Применяются **только при первом запуске**, пока в `/config` ещё нет `psiphon.config`:

Переменная	Описание	По умолчанию
`PUID`	UID процесса psiphon	1000
`PGID`	GID процесса psiphon	1000
`HTTP_PORT`	Порт HTTP-прокси	8080
`SOCKS_PORT`	Порт SOCKS-прокси	1080
`DEVICE_REGION`	Регион устройства	RU
`EGRESS_REGION`	Регион выхода трафика	SG

`PUID`/`PGID` текущего пользователя:

```
id -u && id -g
```

Какой EGRESS_REGION выбрать из RU

Psiphon не публикует официальный актуальный список стран, где у него сейчас подняты серверы выхода — набор доступных `EGRESS_REGION` **динамический** и различается от инстанса к инстансу. Проверить фактически доступный список для конкретного контейнера — по его логам, строка `AvailableEgressRegions`:

```
docker logs psiphon-lt | grep AvailableEgressRegions
```

На практике встречался, например, такой список: `AT, AU, BE, BR, CA, CH, DE, DK, ES, FR, GB, IE, IN, IT, JP, LT, NL, NO, PL, RS, SE, SG, US` — обратите внимание, что скандинавских/прибалтийских кодов вроде `FI`, `EE`, `LV` там могло не быть вовсе, хотя по географии они ближе всего к RU. Если в `EGRESS_REGION` указать код, которого нет в списке доступных, клиент не сможет установить туннель (в логе не будет `ConnectedServerRegion`).

Кандидаты, отсортированные по географической близости к RU, которые стоит пробовать по очереди — но перед использованием сверяйтесь с фактическим списком выше:

Приоритет	Код	Регион
1	`LT`	Литва
2	`PL`	Польша
3	`SE`	Швеция
4	`DE`	Германия
5	`NO`	Норвегия
6	`DK`	Дания
7	`NL`	Нидерланды

Если ни один код не подходит (сервер в этом регионе недоступен) или нужен более общий обход — оставьте `EGRESS_REGION=""` (пустая строка), тогда клиент сам выберет наименее нагруженный доступный сервер, без привязки к региону.

Замерить фактическую задержку для конкретного региона после запуска контейнера:

```
curl -x socks5h://127.0.0.1:1080 -o /dev/null -s -w 'time_total: %{time_total}s\n'
https://ifconfig.me
```

Меняя `EGRESS_REGION` и пересоздавая контейнер (`docker compose up -d --force-recreate`), можно сравнить время ответа по разным кандидатам и оставить самый быстрый из реально работающих.

3. Запуск

```
docker compose up -d
```

4. Проверка

Статус контейнера и логи:

```
docker compose ps && docker logs -f psiphon
```

Healthcheck:

```
docker inspect --format='{{json .State.Health}}' psiphon
```

Ожидаемый статус — `"Status":"healthy"`.

Проверка прокси:

```
curl -x socks5h://127.0.0.1:1080 https://ifconfig.me
curl -x http://127.0.0.1:8080 https://ifconfig.me
```

Вернувшийся IP должен соответствовать выбранному `EGRESS_REGION`.

5. Управление сервисом

```
docker compose restart   # перезапуск
docker compose down      # остановка и удаление контейнера
docker compose pull && docker compose up -d   # обновление образа
```

6. Несколько регионов одновременно

`EGRESS_REGION` — это одно значение на один запущенный контейнер. Чтобы иметь под рукой сразу несколько регионов выхода — поднимают отдельный контейнер на каждый, с уникальными `container_name`, каталогом `config` и внешними портами:

```
services:
  psiphon-lt:
    image: swarupsengupta2007/psiphon:latest
    container_name: psiphon-lt
    environment:
      - PUID=1000
      - PGID=1000
      - HTTP_PORT=8080
      - SOCKS_PORT=1080
      - DEVICE_REGION=RU
      - EGRESS_REGION=LT
    volumes:
      - ./config-lt:/config
    ports:
      - 127.0.0.1:1080:1080
      - 127.0.0.1:8080:8080
    restart: unless-stopped

  psiphon-de:
    image: swarupsengupta2007/psiphon:latest
    container_name: psiphon-de
    environment:
      - PUID=1000
      - PGID=1000
      - HTTP_PORT=8080
      - SOCKS_PORT=1080
      - DEVICE_REGION=RU
      - EGRESS_REGION=DE
    volumes:
      - ./config-de:/config
    ports:
      - 127.0.0.1:1081:1080
      - 127.0.0.1:8081:8080
    restart: unless-stopped

  psiphon-nl:
```

```
    image: swarupsengupta2007/psiphon:latest
    container_name: psiphon-nl
    environment:
      - PUID=1000
      - PGID=1000
      - HTTP_PORT=8080
      - SOCKS_PORT=1080
      - DEVICE_REGION=RU
      - EGRESS_REGION=NL
    volumes:
      - ./config-nl:/config
    ports:
      - 127.0.0.1:1082:1080
      - 127.0.0.1:8082:8080
    restart: unless-stopped
```

Что важно не перепутать:

- **Разные** `container_name` — иначе Docker откажется поднимать второй сервис.
- **Разные каталоги в** `volumes` (`config-lt`, `config-de` и т. д.) — каждый регион хранит свой `psiphon.config`, смешивать их в одной папке нельзя.
- **Разные внешние порты** — внутри контейнера порты одинаковые (1080/8080), наружу пробрасываются на разные (1080/8080, 1081/8081, ...), иначе конфликт портов на хосте.
- Каталоги под каждый регион создаются заранее, как в шаге 1: `sudo mkdir -p /opt/psiphon/config-lt /opt/psiphon/config-de /opt/psiphon/config-nl && sudo chown -R 1000:1000 /opt/psiphon`.

Дальше в приложении просто выбираете локальный порт, соответствующий нужному региону выхода.

Что стоит сделать дополнительно

- Не менять переменные окружения после первого запуска без ручной правки/удаления `config/psiphon.config` — на уже созданный конфиг они не влияют.
- Если прокси должен быть доступен с других машин — не публиковать порт на `0.0.0.0` без дополнительного контроля доступа (файрвол/аутентификация на уровне прокси).
- Настроить бэкап каталога `config`.

Установка Nebula 1.10.3 на Ubuntu

Nebula — оверлейная mesh-VPN от Slack с собственным CA и P2P-соединениями через lighthouse-узлы.

> **Порядок важен.** Сертификаты выпускаются один раз на CA-хосте и раздаются вручную на каждый узел. Установка бинарника и systemd-юнита одинакова что для lighthouse, что для обычного узла — разница только в `config.yml`.

---

1. Каталоги

```
sudo mkdir -p /etc/nebula
```

Бинарник будет лежать в `/usr/local/bin`, конфиг и сертификаты — в `/etc/nebula`.

---

2. Скачивание бинарников (v1.10.3)

```
curl -LO https://github.com/slackhq/nebula/releases/download/v1.10.3/nebula-linux-amd64.tar.gz
```

Распаковка и установка:

```
sudo tar -xzf nebula-linux-amd64.tar.gz -C /usr/local/bin nebula nebula-cert && sudo chmod +x /usr/local/bin/nebula /usr/local/bin/nebula-cert && rm nebula-linux-amd64.tar.gz
```

Проверка версии:

```
nebula -version
```

Ожидается что-то вроде `Version: 1.10.3`.

> Для ARM-серверов (например, Oracle Ampere) вместо `linux-amd64` берите `linux-arm64` — архитектуру проверяйте через `uname -m`.

---

3. Сертификаты

На CA-хосте (выполняется один раз для всей сети)

Если CA ещё не создан:

```
sudo nebula-cert ca -name "Ваша Организация"
```

Создаст `ca.crt` и `ca.key` в текущей директории. `ca.key` **не копируется на узлы** — он остаётся только там, где будут выпускаться новые сертификаты.

Выпуск сертификата под конкретный узел (пример для lighthouse с IP `192.168.100.1` во внутренней nebula-сети `/24`):

```
sudo nebula-cert sign -name "lighthouse1" -ip "192.168.100.1/24"
```

Для обычного узла — аналогично, со своим именем и IP из той же подсети:

```
sudo nebula-cert sign -name "node1" -ip "192.168.100.2/24"
```

На каждом узле

Копируем на сервер (через `scp` с локальной машины или из CA-хоста) три файла в `/etc/nebula`:

- `ca.crt`
- `<имя_узла>.crt`

- `<имя_узла>.key`

```
sudo mv ca.crt node1.crt node1.key /etc/nebula/ && sudo chmod 600 /etc/nebula/*.key
```

---

4. Конфигурация

Берём образец конфига из репозитория как основу:

```
curl -Lo /tmp/config.yml
https://raw.githubusercontent.com/slackhq/nebula/v1.10.3/examples/config.yml
```

```
sudo cp /tmp/config.yml /etc/nebula/config.yml && sudo nano /etc/nebula/config.yml
```

Ключевые параметры, которые нужно поправить:

```
pki:
  ca: /etc/nebula/ca.crt
  cert: /etc/nebula/node1.crt
  key: /etc/nebula/node1.key

static_host_map:
  "192.168.100.1": ["ПУБЛИЧНЫЙ_IP_LIGHTHOUSE:4242"]

lighthouse:
  am_lighthouse: false      # true только на самом lighthouse
  hosts:
    - "192.168.100.1"

listen:
  host: 0.0.0.0
  port: 4242
```

На самом lighthouse-узле: `am_lighthouse: true`, а `lighthouse.hosts` оставляем пустым (он сам себе lighthouse).

---

5. Файрвол

Lighthouse должен принимать UDP на порту nebula (по умолчанию `4242`) из интернета:

```
sudo ufw allow 4242/udp
```

На обычных узлах отдельно открывать этот порт снаружи не обязательно, если исходящий трафик уже разрешён — соединение инициируется изнутри mesh-сети.

---

6. systemd-юнит

```
curl -Lo /tmp/nebula.service
https://raw.githubusercontent.com/slackhq/nebula/v1.10.3/examples/service_scripts/nebula.servi
ce
```

```
sudo cp /tmp/nebula.service /etc/systemd/system/nebula.service
```

Проверяем, что пути в юните совпадают с реальными (`ExecStart=/usr/local/bin/nebula -config /etc/nebula/config.yml`):

```
cat /etc/systemd/system/nebula.service
```

> В типовом юните нет ожидания сети при загрузке — при использовании FQDN в `static_host_map` возможны обрывы после ребута, если DNS ещё не поднялся. Если lighthouse указан по IP (как в примере выше), проблема не актуальна.

Запуск:

```
sudo systemctl daemon-reload && sudo systemctl enable --now nebula
```

---

7. Проверка

Статус сервиса:

```
sudo systemctl status nebula
```

Появился ли туннельный интерфейс (обычно `nebula1`):

```
ip addr show nebula1
```

Пинг lighthouse по внутреннему nebula-адресу:

```
ping -c 3 192.168.100.1
```

Логи в реальном времени, если что-то не так:

```
sudo journalctl -u nebula -f
```

Замечания

- `ca.key` храните отдельно от серверов сети — компрометация этого файла позволяет выпускать сертификаты для входа в mesh под видом любого узла.
- Сертификаты имеют срок действия (по умолчанию у `nebula-cert sign` — 1 год); истечение приводит к тихому разрыву соединения без явной ошибки в логах на стороне соединяющегося узла — стоит следить за `sudo nebula-cert print -path /etc/nebula/node1.crt` на предмет `details.notAfter`.
- Полный список опций конфига и changelog версии — в релизах на GitHub.