Первоначальная настройка 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.