2. Self-Hosted server

Настройка и обслуживание собственного сервера

- Ubuntu server first steps
- SSH customization
- Send telegram message on every ssh login
- Custom domain on Keenetic router

Ubuntu server first steps

Client: Generate keys

This command generate a two files with private and public keys. The public-key file will end with `.pub` extention, a private key file has no extention

```
ssh-keygen
```

Public key you may share with everyone, but private key must be keep in secret.

Client: Share key (publish)

You need to share you public key with target machine (server).

Best way for Ubuntu is add public key to github ssh keys, then it will be avaidable at `github.com/<username>.keys`

Server: Install Ubuntu-server

Becouse you install a server version, the common way to interact is remote control (.ssh), but you need to create base authorization method via physical terminal. The public key made for this purpose, but type public key by hands is takes too long, so we were publish our public key with entire ethernet at `github.com/<username>.keys`

While install, select **import identity** and enter GitHub user name. Ubuntu will automaticaly read public key and save it.

It's strongly recommended use only ssh-keys and disable password authentification.

Connect

For connect from client use this command to connect:

```
ssh -i ~/.ssh/some_server_name username@my.domain.com -p 22
```

where `-i` is identity path (private key path), where `~` - is home client user directory.

where `-p` is target port (default: 22)

where `username` - remote username (you enter while install)

where `my.domain.com` - ip of target machine or domain name

First connection from clinet to server must be as created user (not root).

If connection sucseeded and keys are valid, system ask you to add connection to `known_hosts`, type yes.

> In case of server system reinstall old clients may see message, thats says that identity is changed. This problem rised becouse server keys stored on clinet machine is old. For fix this, in Windows, go to `userfolder/.ssh` and edit `known_hosts` . Delete lines associated with server. And try to connect again.

Root login

After install public key will be stored in `~/.ssh/authorized_keys`

where `~` is user directory: `/home/username`.

For activate root, type this and enter password.

```
sudo su
```

Now you are login as root user and can go to `/root/.ssh`, this folder is already has authorized_keys file, but file is empty. This means that nobody can use ssh for login as root. For fix this you need to copy public key from user authorized_keys to root authorized_keys.

You may use this: (*cat* is print file content, >> is redirect output to new file line).

```
cat /home/<username>/.ssh/authorized_keys >> /root/.ssh/authorized_keys
```

Now you can connet via ssh as root.

Setup after install and connect

Extend disk space

This step may be already done while install, but if not do this:

```
vgdisplay
lvextend -l +100%FREE /dev/mapper/ubuntu--vg-ubuntu--lv
resize2fs /dev/mapper/ubuntu--vg-ubuntu--lv
df -h
```

For laptops:

Disable machine sleep if laptop lid is closed and `reboot` for apply changes:

```
sudo sh -c 'echo "HandleLidSwitch=lock" >> /etc/systemd/logind.conf' && reboot
```

Change hostname

Check information about host:

```
hostnamectl
```

Change hostname:

```
sudo hostname new_hostname
```

Create users

```
useradd -m -s /bin/bash username
```

Cut maximum journal disk size

```
sudo journalctl --vacuum-size=100M
```

Nvidia

```
sudo apt-get purge nvidia-headless-no-dkms-535-server # Or your version
sudo apt-get install nvidia-driver-535 # Or your version
sudo reboot
nvidia-smi # Validate
```

SSH customization

Disable ssh login spam

Rename unnesessary files with `@` at filename start at `/etc/update-motd.d`

```
cd /etc/update-motd.d && for file in *; do mv "$file" "@$file"; done # Rename all files
mv @00-header 00-header
```

also add neofetch as start message if you want to do next step.

```
cd /etc/update-motd.d && echo "neofetch" >> 00-header
```

Customize ssh login text

Message by SSH `/etc/update-motd.d`: use neofetch:

```
apt install neofetch -y
```

Add this line to `/etc/update-motd.d/00-header`

```
neofetch --config /etc/update-motd.d/config.conf --source /etc/update-motd.d/ascii_art.txt
```

Config example:

```
# See this wiki page for more info:
# https://github.com/dylanaraps/neofetch/wiki/Customizing-Info
print_info() {
    info title
    info "OS" distro

    info "Kernel" kernel

    info "Packages" packages
```

```
    info "Shell" shell
    #info "Resolution" resolution
    info "DE" de
    info "WM" wm
    info "WM Theme" wm_theme
    info "Theme" theme
    info "Icons" icons
    info "Terminal" term
    info "Terminal Font" term_font
    info underline
    info "Host" model
    info "CPU" cpu
    info "GPU" gpu

    info underline
    info "Uptime" uptime
    info "Local IP" local_ip
    info "Public IP" public_ip
    info "Users" users

    info underline
    info "Memory" memory
    #info "GPU Driver" gpu_driver  # Linux/macOS only

    info "CPU Usage" cpu_usage
    info "Disk" disk
    info "Battery" battery
    #info "Font" font
    #info "Song" song
    # [[ "$player" ]] && prin "Music Player" "$player"
    #info "Locale" locale  # This only works on glibc systems.

    #info cols
}
```

Colored root bash

Edit `/root/.bashrc` and replace similar liens to this (append -256 color)

```
# set a fancy prompt (non-color, unless we know we "want" color)
case "$TERM" in
    xterm-color|*-256color) color_prompt=yes;;
esac
```

Beautiful screensaver

Screensaver for you server monitor

```
apt install cmatrix
cmatrix -s -b
```

Send telegram message on every ssh login

Add script

Create file `/etc/login-notify.sh`

Modify TELEGRAM_BOT_TOKEN and TELEGRAM_SEND_TO variables. Optional set EXCLUDE_USERS for users about whom a message will not be sent.

```
#!/bin/sh

TELEGRAM_SEND_TO=123456789
TELEGRAM_BOT_TOKEN=123456789:someLETTERS
EXCLUDE_USERS="some_excluded_user another_excluded_user"

if ! echo "${EXCLUDE_USERS}" | grep -q "\<${PAM_USER}\>"; then
if [ "$PAM_TYPE" != "close_session" ]; then
    SSH_KEY=$(grep "Accepted publickey" /var/log/auth.log | tail -n 1 | awk '{print $NF}')
    WHERE_KEY=$(grep "found at" /var/log/auth.log | tail -n 1 | awk '{print $NF}')

    KEYS_PATH=$(echo "$WHERE_KEY" | cut -d ':' -f 1)
    KEYS_LINE=$(echo "$WHERE_KEY" | cut -d ':' -f 2)
    KEY_LINE=$(sed -n "${KEYS_LINE}p" "$KEYS_PATH")
    KEY_NAME=$(echo "$KEY_LINE" | cut -d ' ' -f 3)

    MESSAGE="Server: ${PAM_USER}@`hostname`%0ALogin: ${PAM_RHOST} ${KEY_NAME}"
    curl -s -X POST https://api.telegram.org/bot${TELEGRAM_BOT_TOKEN}/sendMessage -d
chat_id=${TELEGRAM_SEND_TO} -d text="$MESSAGE" > /dev/null &
fi
fi
```

Modify to make it executable

```
chmod +x /etc/login-notify.sh
```

Add script to execute for every login

Do it by modifying file `/etc/pam.d/sshd`, just add line to end of file by echo:

```
echo 'session optional pam_exec.so seteuid /etc/login-notify.sh' >> /etc/pam.d/sshd
```

Increase a log level

Script search for fingerprint, but doesn't know witch `authorized_keys` file used for auth. For get `authorized_keys` file location, we need to print location to `/var/log/auth.log`

Increase log level:

```
echo 'LogLevel VERBOSE' >> /etc/ssh/sshd_config
```

Restart ssh for updated log level:

```
sudo systemctl restart ssh
```

Custom domain on Keenetic router

> Инструкция работает если у вас есть белый статичный ip-адрес и свой домен

Check settings:

- `Domain name` must Enabled `Remote web interface connections`

Setup certificate for domain

- Connect to CLI as `<router_ip>/a`
- Check existing domains:

```
ip http ssl acme list
```

- Add your domain name:

```
ip http ssl acme get <subdomain>.annndruha.space
```

Setup proxy for subdomain

- Check current proxy rules

```
show ip http proxy
```

- Add new proxy route

(config)>

```
ip http proxy <subdomain>
```

(config-http-proxy)>

```
domain static annndruha.space
upstream http 192.168.1.3 4242
```

```
allow public
```