

# Автоустановка систем на базе Linux по сети (RHEL-Based и Deb- based)

- [Подготовка и установка CentOS](#)
- [Debian](#)
- [Ubuntu 20.04/21.10](#)

# Подготовка и установка CentOS

Понадобилось мне на работе реализовать автоматическую установку серверов, чистых, без дополнительного ПО, на базе ОС Ubuntu, CentOS и Debian, однако инфы по этому довольно мало. Еще одна статья не помешает.

## Подготовка

Настраивать PXE сервер будем на базе Ubuntu 20.04.

Для установки потребуются следующие пакеты:

- DHCP сервер (isc-dhcp-server), чтобы сказать биосу, что грузить по сети и выдать адрес
- TFTP (tftp-hpa) сервер, чтобы подать биосу файлы загрузчика
- PXELinux (pxelinux) - PXE загрузчик ОС (файлы по пути /usr/lib/PXELINUX)
- Nginx (или Apache), чтобы подавать файлы ядра, системы, репозитории и т.д.

В случае с CentOS/Ubuntu необходимо также примонтировать ISO образ в папку (пусть будет /srv/tftp/iso), откуда будет грузиться ОС. В случае с дебианом надо распаковать netboot установщик в папку.

У нас к серверу подключается 2 линка - один с выходом в интернет и без DHCP, а второй для загрузки по PXE и установке ОС. В принципе можно адаптировать и под другую конфигурацию.

## Конфигурируем DHCP

Указываем путь до файла загрузчика, указываем подсеть, из которой надо выдавать IP адреса:

```
$ cat /etc/dhcp/dhcpd.conf  
allow bootp;
```

```
allow booting;
max-lease-time 1200;
default-lease-time 900;
log-facility local7;

option ip-forwarding false;
option mask-supplier false;

subnet 10.0.1.0 netmask 255.255.255.0 {

    option routers 10.0.1.30;
    option domain-name-servers 127.0.0.1;
    range 10.0.1.20 10.0.1.26;
    next-server 10.0.1.1;
    filename "pxelinux.0";
}
```

# Конфигурируем TFTP

Указываем путь до папки TFTP:

```
$ cat /etc/default/tftpd-hpa
# /etc/default/tftpd-hpa

TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/srv/tftp"
TFTP_ADDRESS=":69"
TFTP_OPTIONS="--secure"
```

# Конфигурируем Nginx

Надо, чтобы он отдавал файлы по IP адресу PXE сервера, создаем конфиг:

```
$ cat /etc/nginx/sites-enabled/default

server {
    listen 80 default_server;
```

```
listen [::]:80 default_server;

root /srv/tftp;

index index.html;

server_name _;

location / {
    autoindex on;
    try_files $uri $uri/ =404;
}

}
```

## Настраиваем загрузчик

Копируем файл загрузчика по пути `/usr/lib/PXELINUX/pxelinux.0` в `/srv/tftp`. Создаем конфиг для загрузчика по пути `/srv/tftp/pxelinux.cfg/default`. Можно также создавать файлы для конкретного MAC адреса (`/srv/tftp/pxelinux.cfg/01-88-99-aa-bb-cc-dd`). Конфиг:

```
$ cat pxelinux.cfg/default
timeout=1
default 1

LABEL 1
    kernel http://10.0.1.1/CentOS-7/images/pxeboot/vmlinuz
    initrd http://10.0.1.1/CentOS-7/images/pxeboot/initrd.img
    append ip=:::server::dhcp inst.ks=http://10.0.1.1/centos.cfg inst.repo=http://10.0.1.1/iso ---
```

Где `inst.ks` - путь до Kickstart конфига, а `inst.repo` - путь до репозитория (примонтированного образа).

## Готовим CentOS 7/8

На деле самая простая и дружелюбная в автодепное ОСь. Необходимо примонтировать в `/srv/tftp/iso` полный DVD образ установщика и создать Kickstart файл `/srv/tftp/centos.cfg`. Kickstart:

```
$ cat centos.cfg
#version=Centos7
auth --enablesshadow --passalgo=sha512
install
reboot --eject
eula --agreed
url --url="http://10.0.1.1/iso"

# Use graphical install
#graphical
# Use text mode install
text

keyboard --vckeymap=us --xlayouts='ru','us' --switch='grp:alt_shift_toggle'
lang en_US.UTF-8 --addsupport=ru_RU.UTF-8
timezone Europe/Moscow --isUtc --ntpservers=ntp2.stratum2.ru,ntp5.stratum2.ru
firstboot --disable

# Внешний мир
network --bootproto=static --device=00:11:22:33:44:55 --gateway=192.168.100.1 --ip=192.168.100.252 --
nameserver=8.8.8.8 --netmask=255.255.255.0 --activate
# Интерфейс PXE
network --bootproto=dhcp --device=55:66:77:88:99:00 --ipv6=auto --activate
network --hostname=coolserver

# Users
#dedic
rootpw --iscrypted PASSWORD
sshkey --username=root "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQ3b....."

### Partitioning
# System bootloader configuration
bootloader --append=" crashkernel=auto" --location=mbr --boot-drive=sda

ignoredisk --only-use=sda,sdb
clearpart --all
part raid.sda1 --size 512 --asprimary --ondrive=sda
part raid.sdb1 --size 512 --asprimary --ondrive=sdb
```

```
part raid.sda2 --size 10240 --grow --ondrive=sda
part raid.sdb2 --size 10240 --grow --ondrive=sdb

raid /boot --fstype ext4 --device md0 --level=RAID1 raid.sda1 raid.sdb1
raid / --fstype ext4 --device md1 --level=RAID1 raid.sda2 raid.sdb2
```

### Packages

```
%packages
@^minimal
@core
%end

%addon com_redhat_kdump --disable --reserve-mb='auto'
%end

%anaconda
pwpolicy root --minlen=6 --minquality=1 --notstrict --nochanges --notempty
pwpolicy user --minlen=6 --minquality=1 --notstrict --nochanges --emptyok
pwpolicy luks --minlen=6 --minquality=1 --notstrict --nochanges --notempty
%end
```

Обратить внимание следует на настройку сети и реквизиты. Будет создано 2 раздела: /boot на 512 Мб и / на все оставшееся место. Также можно добавить в конце дополнительные команды в раздел `%post`, например:

```
%post
/usr/bin/sed -i "s%#Port 22%Port 43389%g" "/etc/ssh/sshd_config"
/usr/bin/sed -i "s%#PermitRootLogin yes%PermitRootLogin no%g" "/etc/ssh/sshd_config"
%end
```

Пароль можно сгенерировать таким образом: `python -c 'import crypt,getpass; print(crypt.crypt(getpass.getpass(), crypt.mksalt(crypt.METHOD_SHA512)))'`

# Готово!

ACHTUNG! ACHTUNG! При загрузке установщик не будет спрашивать дополнительное подтверждение, диски сразу же затруются.

pxeboot  
pxefont not found or type unknown

# SRC

- <http://www.bog.pp.ru/work/kickstart.html>
- <https://www.golinuxhub.com/2018/05/sample-kickstart-partition-example-raid/>
- [https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/7/html/installation\\_guide/sect-kickstart-howto](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/installation_guide/sect-kickstart-howto)
- <https://docs.centos.org/en-US/centos/install-guide/Kickstart2/>

# Debian

Итак, продолжим, теперь будем ставить Debian. Надо распаковать архив с netboot загрузчиком и подправить конфиг загрузчика. Nginx тут по сути не нужен, ибо вся система умещается в initrd и грузится по PXE:

```
01:51:38 konflicker@pxe-installer:/srv/tftp
$ ls -l iso/
debian-installer
ldlinux.c32
netboot.tar.gz
pxelinux.0
pxelinux.cfg
version.info
```

## Загрузка ОС

В папку `/iso` (относительно ранее настроенного `/srv/tftp`) загружаем образ netboot по ссылке: <https://mirror.yandex.ru/debian/dists/bullseye/main/installer-amd64/current/images/netboot/netboot.tar.gz> и распаковываем в эту же папку.

## Настраиваем загрузчик

Указываем путь до preseed файла, настроим вторую сетевуху по DHCP:

```
label 3
    kernel iso/debian-installer/amd64/linux
    append auto=true vga=788 initrd=iso/debian-installer/amd64/initrd.gz --- quiet
    url=http://10.0.1.1/preseed.cfg net.ifnames=0 biosdevname=0 interface=eth1 priority=critical
```

## Готовим preseed



Файл поместим в `/preseed.cfg`.

Тут уже не все так чисто. Не получится подготовить файл без костылей. Hostname не задается через preseed файл из коробки, но если устраивает стандартный "debian", то можно оставить его.

Настроить отдельно 2 сетевухи тоже не выйдет, надо получать на второй IP по DHCP, а первую конфигурировать через простые команды. Ставиться система будет с зеркал яндекса, но при желании можно скачать себе репозиторий дебиана и указать его в `mirror/http/hostname`.

Соответственно в late\_command мы опишем все эти команды по настройке хостнейма и сети (тут пригодился бы pastebin, но у меня его нет).

```
$ cat preseed.cfg
#_preseed_V1
d-i debian-installer/locale string en_US
d-i localechooser/supported-locales multiselect en_US.UTF-8, ru_RU.UTF-8
d-i keyboard-configuration/xkb-keymap select us
d-i clock-setup/utc boolean true
d-i time/zone string Europe/Moscow
d-i clock-setup/ntp boolean true

### Network
d-i netcfg/choose_interface select eth1
d-i preseed/early_command string ip a add 192.168.100.209/24 dev eth0 ; ip l set up dev eth0 ; echo
'nameserver 8.8.8.8' > /etc/resolv.conf ; ip r del default ; ip r add default via 192.168.100.1 dev eth0
d-i netcfg/hostname string server

# If non-free firmware is needed for the network or other hardware, you can
# configure the installer to always try to load it, without prompting. Or
# change to false to disable asking.
#d-i hw-detect/load_firmware boolean true

### Mirror settings
d-i mirror/country string manual
d-i mirror/http/hostname string mirror.yandex.ru
d-i mirror/http/directory string /debian
d-i mirror/http/proxy string

### User settings
d-i passwd/root-login boolean true
```

d-i passwd/root-password-encrypted password ....

d-i passwd/make-user boolean false

### ### Partitioning

d-i partman-auto/disk string /dev/sda /dev/sdb

d-i partman-auto/method string raid

d-i partman-auto-raid/recipe string 1 2 0 ext4 / /dev/sda1#/dev/sdb1 .

d-i partman-auto/expert\_recipe string multiraid :: 20000 20000 -1 raid \$lvmignore{ } bootable{ } \$primary{ }  
method{ raid } . 20000 20000 -1 ext4 \$defaultignore{ } lvmok{ } method{ format } format{ } use\_filesystem{  
}.

d-i partman-swapfile/percentage string 0

d-i partman-auto-lvm/guided\_size string max

d-i partman-basicmethods/method\_only boolean false

d-i partman-basicfilesystems/no\_swap boolean false

d-i mdadm/boot\_degraded boolean true

d-i partman-md/confirm boolean true

d-i partman-md/confirm\_nooverwrite boolean true

d-i partman-md/device\_remove\_md boolean true

d-i partman/confirm\_write\_new\_label boolean true

d-i partman/confirm\_nooverwrite boolean true

d-i partman/confirm boolean true

d-i partman-lvm/device\_remove\_lvm boolean true

d-i partman-auto-lvm/new\_vg\_name string vg

d-i partman-partitioning/confirm\_write\_new\_label boolean true

d-i partman/choose\_partition select finish

### ### Packages

tasksel tasksel/first string standard

d-i pkgsel/upgrade select safe-upgrade

d-i pkgsel/include string openssh-server initramfs-tools curl wget

### ### Post-config installed system

d-i preseed/late\_command string in-target bash -c \

'echo "auto eth0" >> /etc/network/interfaces; \

echo "iface eth0 inet static" >> /etc/network/interfaces ; \

echo " address 192.168.100.99" >> /etc/network/interfaces ; \

echo " netmask 255.255.255.0" >> /etc/network/interfaces ; \

```
echo " gateway 192.168.100.1" >> /etc/network/interfaces ; \
mkdir -p /root/.ssh ; \
echo ssh-rsa "AAAAB..." > /root/.ssh/authorized_keys ; \
sed -i "s/#PermitRootLogin prohibit-password/PermitRootLogin yes/" /etc/ssh/sshd_config ; \
echo "server" > /etc/hostname ; \
sed -i "s/debian/server/" /etc/hosts ; \
sed -i 's/eth1/d/' /etc/network/interfaces ; \
echo "nameserver 8.8.8.8" > /etc/resolv.conf';

### Etc
d-i save-logs/menu select
d-i save-logs/directory string /target/var/log
d-i grub-installer/only_debian boolean true
d-i grub-installer/with_other_os boolean false
d-i grub-installer/bootdev string default
d-i finish-install/reboot_in_progress note
```

Будет создан один юзер root с авторизацией по паролю через SSH и будет добавлен SSH ключ. При желании для секурности можно создать не привилегированного юзера и запретить авторизацию под рутом.

# Загружаемся

Вы восхитительны!

debian  
Package not found or type unknown

## SRC:

Основная статья: <https://www.howtoforge.com/tutorial/install-debian-9-stretch-via-pxe-network-boot-server/>

- <https://forums.debian.net/viewtopic.php?t=54242>
- <https://www.debian.org/releases/stable/example-preseed.txt>
- <https://debian-handbook.info/browse/stable/sect.automated-installation.html>
- [https://wiki.debian.org/DebianInstaller/Preseed#Autoloading\\_the\\_preseeding\\_file\\_from\\_a\\_webserver\\_via\\_DHCP](https://wiki.debian.org/DebianInstaller/Preseed#Autoloading_the_preseeding_file_from_a_webserver_via_DHCP)
- <https://wiki.debian.org/AutomatedInstallation>
- <https://gist.github.com/annttu/a5b9a57bf03bfc1361ea806fa1bdd116>



# Ubuntu 20.04/21.10

Теперь пришло время самого сложного - установка Ubuntu через cloud-init. Читать это конечно легко, но разобраться было непросто.

## Подготовка

Для автоустановки понадобится 2 файла - `meta-data` и `user-data`. Все действие происходит в `user-data`, а в `meta-data` указываются метаданные установщика, но они нам не пригодятся, так как мы не в облаке. Только спустя долгие мучения понял, что нужно существование *обоих* файлов, даже учитывая, что у нас не облако.

В `meta-data` записываем следующую строку:

```
instance-id: test
```

Можно указать любое значение.

## Настраиваем загрузчик

В `/iso` монтируем образ и указываем до него путь (`url=`), указываем путь до папки (`s=`) с cloud-init скриптами (`meta-data/user-data`)

```
label 6
menu label ^Install Ubuntu
kernel http://10.0.1.1/iso/casper/vmlinuz
initrd http://10.0.1.1/iso/casper/initrd
append url=http://10.0.1.1/ubuntu-21.10-live-server-amd64.iso autoinstall ds=nocloud-net;s=http://10.0.1.1/
cloud-config-url=/dev/null ip=::::::eth1:dhcp fsck.mode=skip net.ifnames=0 biosdevname=0 ---
```

# Настраиваем автоматическую установку

```
$ cat user-data
#cloud-config
autoinstall:
  apt:
    geoip: true
    preserve_sources_list: false
    primary:
      - arches: [amd64, i386]
        uri: http://archive.ubuntu.com/ubuntu
  identity: {hostname: server, password: CRYPTED_PASS,
    realname: ubuntu, username: ubuntu}
  keyboard: {layout: us, toggle: null, variant: ""}
  locale: en_US.UTF-8
  network:
    ethernets:
      eth0:
        critical: true
        addresses: [192.168.100.99/24]
        gateway4: 192.168.100.1
        nameservers:
          addresses: [8.8.8.8]
        eth1: {dhcp-identifier: mac, dhcp4: false}
      version: 2
  ssh:
    allow-pw: true
    authorized-keys:
      - ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQ3b.....
    install-server: true
  storage:
    config:
      - {ptable: gpt, path: /dev/sda, wipe: superblock-recursive,
        preserve: false, name: "", grub_device: true, type: disk, id: disk-sda}
      - {ptable: gpt, path: /dev/sdb, wipe: superblock-recursive,
```

preserve: false, name: "", grub\_device: false, type: disk, id: disk-sdb}

- {device: disk-sda, size: 1048576, flag: bios\_grub, number: 1, preserve: false, grub\_device: false, type: partition, id: partition-sda-0}
- {device: disk-sdb, size: 1048576, flag: bios\_grub, number: 1, preserve: false, grub\_device: false, type: partition, id: partition-sdb-0}
- {device: disk-sda, size: 536870912, wipe: superblock, flag: "", number: 2, preserve: false, grub\_device: false, type: partition, id: partition-sda-1}
- {fstype: ext4, volume: partition-sda-1, preserve: false, type: format, id: format-sda-0}
- {device: disk-sdb, size: 536870912, wipe: superblock, flag: "", number: 2, preserve: false, grub\_device: false, type: partition, id: partition-sdb-1}
- {fstype: ext4, volume: partition-sdb-1, preserve: false, type: format, id: format-sdb-0}
- {device: disk-sda, size: -1, wipe: superblock, flag: "", number: 3, preserve: false, grub\_device: false, type: partition, id: partition-sda-2}
- {device: disk-sdb, size: -1, wipe: superblock, flag: "", number: 3, preserve: false, grub\_device: false, type: partition, id: partition-sdb-2}
- name: md0  
raidlevel: raid1  
devices: [partition-sda-2, partition-sdb-2]  
spare\_devices: []  
preserve: false  
wipe: superblock-recursive  
ptable: gpt  
type: raid  
id: raid-0
- {device: raid-0, size: -1, wipe: superblock, flag: "", number: 1, preserve: false, grub\_device: false, type: partition, id: partition-root-md}
- {fstype: ext4, volume: partition-root-md, preserve: false, type: format, id: format-1}
- {path: /, device: format-1, type: mount, id: mount-1}
- {path: /boot, device: format-sda-0, type: mount, id: mount-0}

version: 1

late-commands:

- echo 'ubuntu ALL=(ALL) NOPASSWD:ALL' > /target/etc/sudoers.d/ubuntu
- sed -ie 's/GRUB\_CMDLINE\_LINUX=.\*GRUB\_CMDLINE\_LINUX="net.ifnames=0 ipv6.disable=1 biosdevname=0"/ /target/etc/default/grub

```
- curtin in-target --target /target update-grub2
```

Разметка должна быть именно такой. Т.е. чтобы RAID root был создан как 2 отдельных отдела, т.е. sda2 и sdb2. Если /dev/sda и /dev/sdb использовать как RAID устройства, то установка не пройдет. Если на одном диске создать /boot и рейд из раздела и диска, то работать не будет.

# Additional

Логи лежат по пути `/var/log/installer` У скрипта автоустановки есть интересная опция `phone_home` - запрос по URL по окончании установки. Можно сделать своеобразный хук. Также cloud-init умеет в chef/puppet.

# SRC

Autoinstall - <https://ubuntu.com/server/docs/install/autoinstall>

PXE Boot - [https://www.golinuxcloud.com/pxe-boot-server-cloud-init-ubuntu-20-04/#Step-4\\_Configure\\_PXE\\_Boot\\_Server](https://www.golinuxcloud.com/pxe-boot-server-cloud-init-ubuntu-20-04/#Step-4_Configure_PXE_Boot_Server)