ИУ-10
Системное
Программное
Обеспечение
Администрирование Linux

Работа с Web и TFTP сервером

Москва, 2022

На этом уроке

1. Обзор репозиториев, файлов конфигурации
2. Установка и базовая настройка NGINX
3. Установка и базовая настройка TFTP-server

Содержание

Управление ПО в Linux

В Ubuntu программное обеспечение делится на четыре вида по типу лицензирования и уровню поддержки:

1. **Main** — свободное ПО, официально поддерживаемое компанией Canonical.

2. **Restricted** — проприетарное ПО (в основном драйверы устройств), официально поддерживаемое компанией Canonical.

3. **Universe** — свободное ПО, официально не поддерживаемое компанией Canonical, но поддерживаемое сообществом пользователей.

4. **Multiverse** — проприетарное ПО, не поддерживаемое компанией Canonical.

Все перечисленные виды программного обеспечения представляют из себя набор файлов и библиотек, объединенных для выполнения определенного функционала, которые мы можем установить. Это объединение называется пакетом.

Репозитории и управление репозиториями

Пакеты располагаются в своих хранилищах - репозиториях. Репозиторий может быть размещён локально, может находиться на носителе (флешке, DVD-диске), но чаще всего он размещён в интернете. Условно репозитории можно разделить на три группы:

1. **Стандартные репозитории** — это репозитории, поддерживаемые разработчиками операционных систем. Включают в себя стабильные версии программного обеспечения. Зачастую эти версии отстают на несколько шагов от последних версий пакетов.

2. **Дополнительные репозитории** — репозитории, поддерживаемые разработчиками программного обеспечения. Включают в себя последние стабильные версии ПО. Зачастую узкоспециализированы под конкретный пакет и библиотеки, необходимые для этого пакета.

3. **Неофициальные репозитории** — репозитории, созданные сообществом или одним человеком. Могут содержать в себе как последние стабильные, так и тестируемые версии программного обеспечения.

Официальные репозитории Ubuntu делятся на следующие типы:

1. **$release** — пакеты на момент выхода релиза.
2. **$release-security** — пакеты критических обновлений безопасности.
3. **$release-updates** — пакеты обновления системы, то есть более поздние версии ПО, вышедшие уже после релиза.
4. **$release-backports** — пакеты более новых версий ПО, которое доступно только в нестабильных версиях Ubuntu.
5. **partner** — репозиторий, содержащий ПО компаний-партнёров Canonical.

Информация о подключённых репозиториях в Ubuntu хранится в каталоге **/etc/apt/**, в файле **sources.list**. **Важно!** *Репозитории защищают от подмены при помощи сверки цифровых подписей репозитория и клиента. В репозитории хранится закрытая часть ключа, у клиента — открытая часть ключа.*

В Ubuntu репозитории можно подключить тремя способами: используя графический интерфейс, путём редактирования файла **/etc/apt/source.list** или добавления файла в каталог **/etc/apt/source.list.d/** и используя утилиту apt. Рассмотрим два последних варианта.

Добавление репозитория через добавления файла

В текстовом редакторе открываем файл **/etc/apt/source.list.d/repo.list** и в конце файла вводим строку вида:

```
deb http://адрес_репозитория версия_дистрибутива ветки
```

Например, добавим репозиторий nginx, для этого создадим в каталоге /etc/apt/source.list.d/ файл nginx.list следующего содержания:

```
deb http://nginx.org/packages/ubuntu focal nginx
```

Здесь nginx — название ветки, содержащей пакеты, необходимые для установки nginx, а focal — это кодовое имя версии. Узнать его можно с помощью команды `lsb_release -a`.

```
root@server:~# lsb_release -a
No LSB modules are available.
Distributor ID:    Ubuntu
Description: Ubuntu 20.04.1 LTS
Release:     20.04
Codename:    focal
```

Следующий шаг — это установка публичного ключа репозитория, для этого нужна команда **apt-key**. Скачиваем при помощи **curl** наш ключ и передаём через pipe утилите **apt-key**:

```
curl -fsSL https://nginx.org/keys/nginx_signing.key | sudo apt-key add -
```

И последний шаг — это обновление информации о пакетах **sudo apt update** и установка пакета **sudo apt install nginx -y**.

Создание файлов с репозиториями и размещение их в каталоге **/etc/source.list.d/** удобно тем, что мы можем обновить информацию о пакетах из конкретного репозитория **apt update repo_name**.

Добавление репозитория, используя команду apt-add-repository

Эта команда автоматически создаёт записи в файле **/etc/apt/sources.list** или создаёт файл репозитория в каталоге **/etc/apt/sources.list.d/**, а также может удалять информацию о репозиториях. Чаще всего эта утилита используется для добавления PPA-репозиториев.

PPA-репозитории находятся на сайте Launchpad.net, который поддерживается компанией Canonical. Утилита автоматически находит строку для записи в файл репозитория, скачивает и импортирует ключи. Рассмотрим добавление репозитория nginx с использованием PPA-репозитория:

```
apt-add-repository ppa:nginx/stable
```

Здесь утилите apt-add-repository мы говорим, что подключаем PPA-репозиторий, поддерживаемый группой nginx, и подключаем стабильную версию. Утилита автоматически создаст файл **/etc/apt/sources.list.d/nginx-ubuntu-stable-bionic.list** с содержимым, которое мы можем просмотреть при помощи команды cat:

```
cat /etc/apt/sources.list.d/nginx-ubuntu-stable-focal.list

"deb http://ppa.launchpad.net/nginx/stable/ubuntu focal main"
```

Утилита импортирует ключи и обновит список пакетов.

Управление пакетами

В Ubuntu управление пакетами осуществляется тремя способами: с использованием утилиты apt, dpkg или snap. Разберем первые две как наиболее популярные.

Управление пакетами через утилиту apt

apt - пакетный менеджер, который включает в себя набор утилит для управления пакетами. Он позволяет осуществлять поиск, установку и удаление пакетов, обновлять операционную систему, подключать репозитории. Подключение репозиториев с использованием **apt** было рассмотрено в предыдущей части. Рассмотрим параметры утилиты **apt** для управления пакетами:

- `apt search package_name` — поиск пакета;
- `apt show package_name` — посмотреть информацию о пакете;
- `apt install package_name -y` — установить пакет;
- `apt install package_name1 package_name2 -y` — установить два пакета;
- `apt remove package_name` — удалить пакет, при этом сохранятся файлы с настройками;
- `apt purge package_name` — полностью удалить пакет, включая конфигурационные файлы;
- `apt upgrade` — обновить все установленные пакеты;
- `apt update` — обновить информацию о пакетах в репозиториях, указанных в настройках.

Управление пакетами через утилиту dpkg

Dpkg — пакетный менеджер в Debian-подобных системах. Главное отличие от утилиты **apt** состоит в том, что **dpkg** работает только с локальными пакетами, он не умеет искать и устанавливать пакеты с репозиториев. Основные параметры утилиты **dpkg**:

- `dpkg -l` — просмотр списка пакетов;
- `dpkg -i package_name` — установить пакет или группу пакетов;
- `dpkg -r package_name` — удалить пакет или группу пакетов.

TFTP сервер

TFTP или Trivial File Transfer Protocol - это протокол, который можно использовать для быстрого перемещения *небольших файлов*. В отличии от FTP, TFTP работает поверх UDP (порт 69) и в большинстве случаев не использует имя пользователя и пароль. При этом файл передается частями по 512 байт с подтверждением каждого, что замедляет передачу больших файлов.

TFTP используется там, где безопасность не требуется. Вместо этого вам нужен способ легко выгружать файлы на сервер и скачивать их с сервера. Коммутаторы и маршрутизаторы используют протокол TFTP для хранения файлов конфигурации и образов ПО в целях резервного копирования. Протоколы сетевой загрузки, такие как BOOTP, PXE используют этот протокол для загрузки операционных систем по сети. Многие электронные платы и микропроцессоры также используют TFTP для загрузки прошивки в чип.

Давайте рассмотрим установку и использование пакеты **tftpd-hpa** в качестве TFTP-сервера.

Установим,включим и добавим службу в автоматическую загрузку.

```
root@server:~# apt-get install tftpd-hpa
Reading package lists... Done
Building dependency tree
Reading state information... Done
...
root@server:~# systemctl start tftpd-hpa.service
root@server:~# systemctl enable tftpd-hpa.service
tftpd-hpa.service is not a native service, redirecting to systemd-sysv-
install.
Executing: /lib/systemd/systemd-sysv-install enable tftpd-hpa
root@server:~#
root@server:~#
root@server:~# systemctl status tftpd-hpa.service
● tftpd-hpa.service - LSB: HPA's tftp server
    Loaded: loaded (/etc/init.d/tftpd-hpa; generated)
    Active: active (running) since Wed 2021-01-13 11:50:06 UTC; 2min 31s
   ago
      Docs: man:systemd-sysv-generator(8)
     Tasks: 1 (limit: 2282)
    Memory: 616.0K
    CGroup: /system.slice/tftpd-hpa.service
            └4642 /usr/sbin/in.tftpd --listen --user tftp --address :69
--secure /srv/tftp
```

```
Jan 13 11:50:06 server systemd[1]: Starting LSB: HPA's tftp server...
Jan 13 11:50:06 server tftpd-hpa[4634]: * Starting HPA's tftpd in.tftpd
Jan 13 11:50:06 server tftpd-hpa[4634]:   ...done.
Jan 13 11:50:06 server systemd[1]: Started LSB: HPA's tftp server.
root@server:~#
```

Конфигурация сервера находится в /etc/default/tftpd-hpa и содержит следующие параметры.

- **TFTP_USERNAME="tftp"**. Пользователь, от имени которого будет запущен сервис, **а не логин доступа к серверу**.
- **TFTP_DIRECTORY="/srv/tftp"**. Каталог, доступный по tftp.
- **TFTP_ADDRESS=":69"**. Порт, на котором работает протокол.
- **TFTP_OPTIONS="--secure"**. Эта переменная устанавливает изменение каталога TFTP на тот, который установлен в переменной **TFTP_DIRECTORY** при подключении к серверу TFTP. Это функция безопасности. Если этот параметр не установлен, после подключения придется установить каталог вручную. Это очень хлопотно и небезопасно.

Добавим опцию **--create**, так как без нее на сервер невозможно будет загружать новые файлы.

```
root@server:~# cat /etc/default/tftpd-hpa
# /etc/default/tftpd-hpa

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

Для корректного доступа к файлам сменим владельца и группу папки **/srv/tftp**. Рестартуем сервис и создадим файл в папке.

```
root@server:~# chown tftp:tftp /srv/tftp/
root@server:~# systemctl restart tftpd-hpa.service
root@server:~# systemctl status tftpd-hpa.service
● tftpd-hpa.service - LSB: HPA's tftp server
    Loaded: loaded (/etc/init.d/tftpd-hpa; generated)
    Active: active (running) since Wed 2021-01-13 12:22:53 UTC; 4s ago
      Docs: man:systemd-sysv-generator(8)
         Process: 4981 ExecStart=/etc/init.d/tftpd-hpa start (code=exited,
status=0/SUCCESS)
     Tasks: 1 (limit: 2282)
    Memory: 492.0K
    CGroup: /system.slice/tftpd-hpa.service
            └─4998 /usr/sbin/in.tftpd --listen --user tftp --address :69
--secure --create /srv/tftp
```

```
Jan 13 12:22:53 server systemd[1]: Starting LSB: HPA's tftp server...
Jan 13 12:22:53 server tftpd-hpa[4981]: * Starting HPA's tftpd in.tftpd
Jan 13 12:22:53 server tftpd-hpa[4981]: ...done.
Jan 13 12:22:53 server systemd[1]: Started LSB: HPA's tftp server.
root@server:~# echo "Some data on TFTP!"> /srv/tftp/tftp_data
```

Для доступа к TFTP-серверу понадобится клиентская программа TFTP. Она вам понадобится только для тестирования, потому что на устройствах, которые будут использовать TFTP-сервер, скорее всего уже будет установлена клиентская программа. Например, почти на всех маршрутизаторах и коммутаторах уже есть клиент.

```
user-client_machine:~ user$ tftp 192.168.1.84
tftp> status
Connected to 192.168.1.84.
Mode: netascii Verbose: off Tracing: off
Rexmt-interval: 5 seconds, Max-timeout: 25 seconds
tftp> get tftp_data
getting from 192.168.1.84:tftp_data to tftp_data [netascii]
Received 20 bytes in 0.0 seconds [inf bits/sec]
Sent 12 bytes in 0.0 seconds
tftp> put local_data
tftp> quit
user-client_machine:~ user$ cat tftp_data
Some data on TFTP!
```

Таким образом мы можем загружать и скачивать файлы, настройка сервера закончена.

```
root@server:~# ls -la /srv/tftp/
total 16
drwxr-xr-x 2 tftp tftp 4096 Jan 13 12:32 .
drwxr-xr-x 3 root root    4096 Jan 13 11:49 ..
-rw-rw-rw- 1 tftp tftp      11 Jan 13 12:32 local_data
-rw-r--r-- 1 root root      19 Jan 13 12:26 tftp_data
```

Web сервер

Веб-сервер — сервис, принимающий HTTP-запросы и выдающий ответы, как правило, содержащие HTML-код (но могут присутствовать и двоичные файлы).

HTTP сам по себе - это простой текстовый протокол, позволяющий запросить у веб-сервера информацию, а серверу ее отдать. HTTP-протокол позволяет передавать заголовки, на основе которых выполняется обработке данных. HTTP-клиент, в роли которого, как правило, выступает брау-

зер, сообщает информацию о себе через user agent. curl и wget тоже имеют user agent, таким образом, не каждый HTTP-клиент — браузер. Веб-сервер сообщает информацию о себе, метаинформацию о документе, кодировку и тип содержимого (MIME-type), что позволит клиенту корректно раскодировать информацию.

Существует несколько типов запросов, но наиболее часто применяются GET и POST.

GET служит для запроса страницы. Как правило, в качестве адреса ресурса в GET передаётся путь к файлу относительно веб-директории. Например, **/articles/article.html**.

С помощью POST отправляются новые файлы на веб-сервер (upload), записи на форумах и блогах, логин и пароль для входа в ту или иную систему.

Протокол HTTP, как правило, работает через TCP-сессию. Поэтому для приёма нескольких одновременных соединений веб-сервер должен использовать процессы или потоки.

Nginx

Одним из самых распространенных Веб-серверов является NGINX, позиционируется как простой, быстрый и надежный сервер. В целом, NGINX скорее веб-прокси, чем полноценный веб-сервер, так как для работы с динамическим контентом используются внешние приложения.

Nginx использует следующий механизм: он создает один мастер-процесс и один или более процессов-воркеров (worker). Как правило, на одном CPU используется один воркер, но может применяться и больше, например при интенсивном вводе-выводе. Все входящие подключения распределяются по воркерам. Каждый воркер использует одну очередь и низкоуровневые механизмы, что позволяет обработать довольно значительное число соединений. Используя низкоуровневые возможности операционной системы, и вместо того, чтобы доверять переключение между задачами процессорного времени с помощью потоков, Nginx самостоятельно реализует обработку одновременных соединений. Этим достигается высокое быстродействие, отказоустойчивость, но именно поэтому Nginx может работать только со статическим содержимым. Чтобы использовать динамическое содержимое, генерируемое скриптами, Nginx может получать результаты выполнения от сторонних приложением, например PHP, работающего через модуль

php-fpm. Так как в таком случае скрипты выполняются в другом процессе, работа Nginx безопасна. Nginx будет хорошо работать, если файлы, к которым обращаются HTTP-запросы, не блокируются другими приложениями.

Для работы nginx нам понадобится свободный 80 порт, поэтому убедитесь что он не занят другим процессом.

```
root@server:~# netstat -tulapn | grep 80
root@server:~#
```

В первой части урока мы добавили репозиторий, содержащий нужный нам пакет, поэтому осталось только его установить и запустить.

```
root@server:~# apt-get update
root@server:~# apt-get install nginx
root@server:~# systemctl start nginx.service
root@server:~# systemctl status nginx.service
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor
preset: enabled)
     Active: active (running) since
```

Nginx доступен с набором настроек по умолчанию и стартовая страница выглядит следующим образом.

Рис.1 - Стартовая страница NGINX

Базовые настройки

Файлы конфигурации находятся в каталоге /**etc/nginx**. Основным является является /**etc/nginx/nginx.conf**.

```
root@server:~# cat /etc/nginx/nginx.conf | grep -vP '^\s*#|^$'
user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;
events {
      worker_connections 768;
}
http {
      sendfile on;
      tcp_nopush on;
      types_hash_max_size 2048;
      include /etc/nginx/mime.types;
      default_type application/octet-stream;
      ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3; # Dropping SSLv3, ref:
POODLE
      ssl_prefer_server_ciphers on;
      access_log /var/log/nginx/access.log;
      error_log /var/log/nginx/error.log;
      gzip on;
      include /etc/nginx/conf.d/*.conf;
      include /etc/nginx/sites-enabled/*;
}
```

Основными понятиями в конфигурации являются контекст и директива. Контекст - это набор директив, заключенных в фигурные скобки.

```
events {
      worker_connections 768;
}
```

В данном примере `worker_connections` находятся в контексте `events`. При этом `events` является директивой основного контекста. **Примечание!** *Одинаковые директивы могут находиться в разных контекстах.В этом случае будет применяться последняя.Далее будут приведены примеры таковых*

Начнем составлять конфигурацию с нуля.

```
events {}

http{

server {
        listen 80;
        server_name 192.168.1.84;

        }
}
```

- `events {}` относится к обработке соединений, она должна существовать всегда, но может быть пустой

- `http` описывает протокол HTTP.NGINX может обрабатывать и почтовые запросы (директива mail)

- `listen` указывает, на каком порту необходимо открыть соединение

- `server_name` проверяет значение HTTP-заголовка “Host”. В данный момент стоит указать IP адрес нашего сервера Linux.

Проверим конфигурацию на ошибки и перечитаем ее.

```
root@server:~# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
root@server:~# systemctl reload nginx.service
```

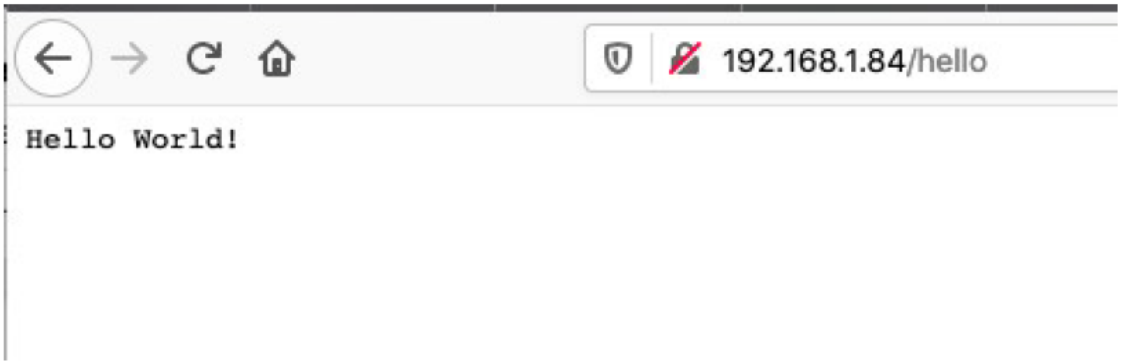
Добавим директивы `location` ,определяющие шаблоны, по которым ищут совпадения запроса после “/” (слэш). В перед описанием шаблона может ничего не стоять, тогда шаблон будет означать префикс, = точное совпадение шаблона, ∼ регулярное выражение, * case insensitive .

```
events {}

http{

server {
        listen 80;
        server_name 192.168.1.84;
         location /hello {
                        return 200 'Hello World!';
                }
         location ~*/hi[0-9] {
                        return 200 'Hi There!';
                }
        }
}
```

В браузере мы увидим совпадение первого



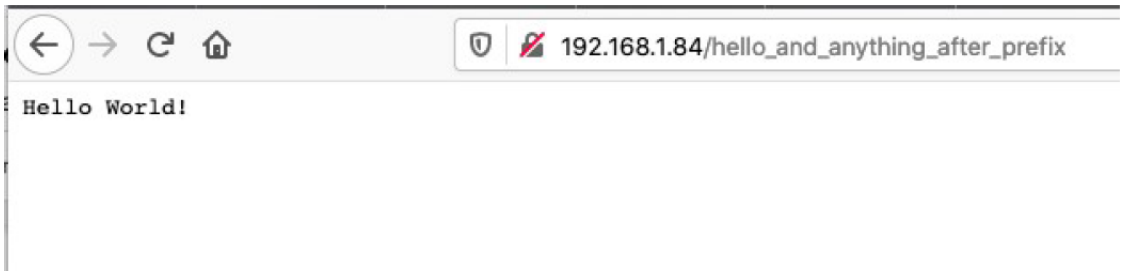


Рис2-3. Работа директивы location

Статические файлы

Директива root определяет корневой каталог, который будет использоваться для поиска файла. Чтобы получить путь к запрошенному файлу, NGINX добавляет URI запроса к пути, указанному в корневой директиве. Директива может быть размещена на любом уровне в контексте http, server или location.

```
root@server:~# cat /etc/nginx/nginx.conf
events {}

http{

server {
        listen 80;
        server_name 192.168.1.84;
        root /srv/www;
        location /hello {
                        return 200 'Hello World!';
                }
        location ~*/hi[0-9] {
                        return 200 'Hi There!';
                }
        }
}
root@server:~# echo "Web data!" > /srv/www/web_data
root@server:~#
root@server:~# wget http://192.168.1.84:/web_data
--2021-01-13 14:30:19-- http://192.168.1.84/web_data
Connecting to 192.168.1.84:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10 [text/plain]
Saving to: 'web_data'

web_data
100%[==========================================================================
==========>]       10 --.-KB/s     in 0s

2021-01-13 14:30:19 (1024 KB/s) - 'web_data' saved [10/10]

root@server:~# cat web_data
Web data!
```

Мы можем создать отдельные папки под хранение файлов конфигураций и образов, программного обеспечения. Для того, чтобы не запоминать весь путь, а опираться на расширения запрашиваемых файлов, в разных контекстах `location` использовать требуемую директиву `root`.

```
root@server:~# mkdir /srv/www/conf
root@server:~# mkdir /srv/www/iso
root@server:~#
root@server:~# cat /etc/nginx/nginx.conf
events {}

http{

server {
        listen 80;
        server_name 192.168.1.84;
        location ~ \.cfg$ {
            root /srv/www/conf;
                }
        location ~
            \.iso$ { root
            /srv/www/iso;
                }
        }
}
```

В данном примере iso файлы будут забираться из **/srv/www/iso**, файлы с расширением cfg из папки **/srv/www/conf**.

Практическое задание

1. Установить и настроить TFTP-сервер, загрузить и скачать оттуда текстовый файл с содержимым на свое усмотрение.

2. Установить и настроить NGINX. Сделать так, чтобы по запросу **http://<ip адрес>/main** можно было скачать файл **/srv/www/main**, а по запросу **http://<ip адрес>/main.cfg** - файл **/srv/www/conf/main.cfg**. Содержимое файлов может быть произвольным.

Глоссарий

Репозиторий — место, где хранятся и поддерживаются какие-либо данные. Чаще всего данные в репозитории хранятся в виде файлов, доступных для дальнейшего распространения по сети.

Пакет — под пакетами в Linux подразумевается программное обеспечение, которое можно установить, то есть набор файлов, объединённых для выполнения определённого функционала. Пакеты как правило хранятся в репозиториях.

PPA (сокр. от англ. Personal Packages Archive) — персональный архив пакетов. В отличие от других репозиториев Ubuntu, PPA-репозиторий содержит версии только какой-то одной программы.

Apt — программа для установки, обновления и удаления программных пакетов в операционных системах Debian и основанных на них. В Apt есть коровья суперсила.

Dpkg — это пакетный менеджер для Debian-систем. Он может устанавливать, удалять и создавать пакеты, но, в отличие от других систем управления пакетами, не может автоматически загружать и устанавливать пакеты или их зависимости.

Порт — целое неотрицательное число, записываемое в заголовках протоколов транспортного уровня модели OSI (TCP, UDP, SCTP, DCCP). Используется для идентификации процесса-получателя, т.е помимо адреса хоста необходимо идентифицировать приложения - это происходит с помощью номера порта. Номера портов уникальны в пределах одного хоста.

Дополнительные материалы

https://docs.nginx.com/nginx/admin-guide/

Основы управления пакетами Ubuntu

Установка snap-пакетов

Используемые источники

http://nginx.org/en/docs/ngx_core_module.html

https://docs.nginx.com/nginx/admin-guide/web-server/serving-static-content/

https://linuxhint.com/install_tftp_server_ubuntu/

Управление пакетами в Ubuntu