Netplan

Файлы

- `/etc/netplan` - хранилище файлов конфигурации. Файлы могут иметь любое имя, окончание `*.yaml`

Синтаксис

```
поле0:
  поле1: значение
  поле2:
    - элемент1
    - элемент2
    - элемент3
```

Имя поля и его значение разделяется двоеточием. В качестве значения поля можно передавать не только текстовое или числовое значение, но и другое поле, несколько полей или список значений. При передаче списка каждый новый элемент списка должен начинаться с дефиса. Табуляции использовать нельзя. Отступы используются для указания структуры. Например, из примера видно, что **поле1** и **поле2** относятся к **полю0**. Это всё, что касается общего синтаксиса, теперь про Netplan:

```
network:
  version: 2
  renderer: программа_бэкенд
  вид_интерфейса:
    имя_интерфейса:
       параметр: значение
```

Первые две строчки конфигурации стандартны. Первая указывает, что мы будем иметь дело с сетью, а вторая указывает версию стандарта конфигурации, которая будет использоваться. Их лучше не трогать.

- **renderer** - указывает программу, для которой будут преобразоваться ваши настройки. На данный момент поддерживаются только **network-manager** (знач. NetworkManager) и systemd-networkd (знач. networkd);
- **вид_интерфейса** - вид сетевых интерфейсов, которые вы будете настраивать в этой секции. Они делятся на физические: ethernets (проводные), wifis (беспроводные) и виртуальные: vlans , bonds, bridges.
- **имя_интерфейса** - имя сетевого интерфейса в системе, например enp3s0 или eth0;
- **параметры** - настройки, с помощью которых указывается, как нужно подключаться к сети.

Параметры

Мы разобрались с основным синтаксисом, далее разберём команды, с помощью которых мы будем настраивать сеть:

- **renderer** - программа для обработки конфигурации;
- **dhcp4** - получение IPv4 адреса по DHCP;
- **dhcp6** - получение IPv6 адреса по DHCP;
- **dhcp-identifier** - если передать значение "mac", то будет использоваться MAC-адрес в качестве идентификатора DHCP;
- **addresses** - добавляет статические адреса к интерфейсу, можно несколько;
- **gateway4** - указывает шлюз IPv4;
- **gateway6** - указывает шлюз IPv6;
- **nameservers** - указывает DNS-серверы;
- **macaddress** - устанавливает новый MAC-адрес;
- **routes** - позволяет настроить маршруты таблицы маршрутизации;
- **routing-policy** - дополнительная настройка маршрутов, для IP или подсети;
- **access-points** - список точек доступа для Wi-Fi;
- **password** - пароль для точки доступа Wi-Fi;
- **mode** - режим работы сетевой карты Wi-Fi.

Конфигурация

Базовая конфигурация

```
network:
    version: 2
    renderer: networkd
    ethernets:
        ens3:
            dhcp4: true
        ens7:
            dhcp4: no
            addresses: [192.168.122.195/24]
            gateway4: 192.168.122.1
            mtu: 1500
            nameservers:
                addresses: [8.8.8.8, 77.88.8.8]     # Сервера
                search: [ dmosk.local ]             # Суффикс
        ens9:
            dhcp4: no
            addresses: [192.168.1.10/24, 192.168.1.20/24]
            nameservers:
                addresses:
                    - 8.8.8.8
                    - 77.88.8.8
                search: [ dmosk.local, dmosk.ru ]
```

** где:*

- **version** — версия YAML. На момент обновления статьи, была 2.
- **renderer** — менеджер сети (networkd или NetworkManager).
- **ethernets** — настройка сетевых адаптеров ethernet.

- **ens3, ens7, ens9** — настройки для соответствующих сетевых адаптеров. В данном примере мы настраиваем 3 сетевых адаптера.
- **dhcp4** — будет ли получать сетевой адаптер IP-адрес автоматически. Возможны варианты yes/true — получать адрес автоматически; no/false — адрес должен быть назначен вручную.
- **addresses** — задает IP-адреса через запятую.
- **gateway4** — шлюз по умолчанию. В данном примере указывается только для интерфейса ens7.
- **mtu** — при желании, можно задать значение MTU.
- **nameservers** — настройка серверов имен (DNS).
- **nameservers addresses** — указываем серверы DNS. Обратите внимание на разный формат записи для ens7 и ens9. Приемлемы оба варианта.
- **nameservers search** — дописывает окончание домена, если мы обращаемся к узлу сети только по его имени. Стоит обратить внимание, что мы можем указать несколько доменов через запятую.

Статический маршрут

```
network:
    version: 2
    renderer: networkd
    ethernets:
        ens9:
            dhcp4: no
            addresses: 192.168.1.10/24
            nameservers:
                addresses:
                    - 8.8.8.8
                    - 77.88.8.8
            routes:
              - to: 192.168.0.0/24
                via: 192.168.1.1
                on-link: true
```

*в данном примере мы настроили маршрут для сетевого интерфейса **ens9**. Данная настройка задается параметром **routes**:*

- **to** — направление маршрута (в какую сеть мы должны попадать). В данном примере, 192.168.0.0/24.
- **via** — через какой шлюз мы попадаем в сеть to.
- **on-link** — активация маршрута при поднятии линка на сетевом интерфейсе.

Объединение интерфейсов (bonds)

С помощью bonds мы можем объединить интерфейсы с целью обеспечения отказоустойчивости и/или повышения пропускной способности.

```
network:
    version: 2
    renderer: networkd
    ethernets:
```

```
        ens2f0: {}
        ens2f1: {}
    bonds:
        bond0:
            dhcp4: no
            interfaces:
            - ens2f0
            - ens2f1
            parameters:
                mode: active-backup
            addresses:
                - 192.168.122.195/24
            gateway4: 192.168.122.1
            mtu: 1500
            nameservers:
                addresses:
                    - 8.8.8.8
                    - 77.88.8.8
```

*в данном примере мы объединяем физические интерфейсы **ens2f0** и **ens2f1**; настройка **parameters mode** указываем на тип объединения — доступны варианты:*

- **balance-rr** - задействуются оба интерфейса по очереди, распределение пакетов по принципу Round Robin).
- **active-backup** - используется только один интерфейс, второй активируется в случае неработоспособности первого).
- **balance-xor** - задействуются оба интерфейса по очереди, распределение пакетов на основе политики хеширования xmit_hash_policy).
- **broadcast** - задействуются оба интерфейса одновременно, пакеты передаются все интерфейсы).
- **802.3ad** - задействуются оба интерфейса по очереди, распределение пакетов на основе политики хеширования xmit_hash_policy)
- **balance-tlb** - задействуются оба интерфейса по очереди, пакеты распределяются в соответствии с текущей нагрузкой)

Сетевой мост (bridge)

Сетевой мост позволяет пропускать сетевой трафик через другой сетевой адаптер. Это можно применить, например, для организации хоста виртуальных машин (для трансфера трафика к виртуальным машинам KVM через единственный сетевой интерфейс сервера).

```
network:
    version: 2
    renderer: networkd
    ethernets:
        ens2f0: {}
    bridges:
        br0:
            macaddress: ce:ce:ce:45:45:45
            interfaces:
                - ens2f0
            addresses:
                - 192.168.1.15/24
            gateway4:
```

```
            nameservers:
                addresses:
                    - 77.88.8.8
                    - 8.8.8.8
            mtu: 1500
            parameters:
                stp: true
                forward-delay: 4
            dhcp4: false
            dhcp6: false
```

** где:*

- **bridges** — настройки для интерфейсов bridge.
- **bridges br0** — настройка интерфейса br0.
- **macaddress** — физический адрес (MAC) интерфейса. Настройка важна для некоторых провайдеров VPS — без нее бридж может не заработать.
- **interfaces** — перечисление интерфейсов, из которых собираем мост. В данном примере ens2f0.
- **addresses, gateway4, nameservers** — сетевые настройки (IP-адрес, шлюз, сервер имен).
- **mtu** — одноименный параметр. Для сетей ethernet обычно равен 1500.
- **parameters stp** — включает или отключает устранение петель в сети. В данном примере включено.
- **parameters forward-delay** — время в секундах в течение которого мост будет оставаться в состояниях «Listening» и «Learning».
- **dhcp4, dhcp6** — включает или отключает автоматическое получение IP-адреса. В нашем случае, отключает.

VLAN

```
network:
    version: 2
    renderer: networkd
    ethernets:
        ens3: {}
    vlans:
        vlan5:
            id: 5
            link: ens3
            dhcp4: no
            addresses: [10.0.0.15/24]
            gateway: 10.0.0.1
```

** в данном примере мы настроили интерфейс с тегом **5** на физическом адаптере **ens3**.*

WiFi

```
network:
    version: 2
    renderer: networkd
    wifis:
```

```
        wlp2s0b1:
            dhcp4: no
            dhcp6: no
            addresses: [192.168.2.10/24]
            gateway4: 192.168.2.1
            nameservers:
                addresses: [192.168.2.1, 77.88.8.8]
            access-points:
                <имя WiFi сети (SSID)>:
                    password: wifi_password
```

** где:*

- **wifis** — определяет настойки для WiFi.
- **wlp2s0b1** — настройка для беспроводного сетевого адаптера.
- **dhcp4, dhcp6** — включает или отключает автоматическое получение IP-адреса.
- **addresses, gateway4, nameservers** — настройка сети (IP-адрес, шлюз, сервер DNS).
- **access-points** — настройка для подключения к беспроводной сети.
- **<имя WiFi сети (SSID)>** — имя беспроводной сети, к которой будем подключаться.
- **password** — пароль для подключения к беспроводной сети.

Управление

Синтаксис команды: `netplan <опции> <команда>`

`--debug` - более подробно

- **try** - попробовать применить конфигурацию с возможностью отмены;
- **apply** - применить конфигурацию;
- **generate** - проверка текущей конфигурации и запись на диск;
- **config** - записать текущую конфигурацию сети в YAML.

От Netplan к Interfaces

При желании, мы можем вернуть привычный принцип настройки сети. Для этого выполним несколько шагов.

1. Открываем настройку grub. Находим опцию **GRUB_CMDLINE_LINUX** и дописываем в нее параметр:

   ```
   vi /etc/default/grub

   GRUB_CMDLINE_LINUX="netcfg/do_not_use_netplan=true"
   ```

   ** если **GRUB_CMDLINE_LINUX** содержит другие настройки, то наш параметр добавляем через пробел.*

2. Устанавливаем пакет ifupdown:

```
apt install ifupdown
```

3. Настраиваем сеть в файле:

```
vi /etc/network/interfaces
```

... например:

```
auto lo
iface lo inet loopback

auto ens5
iface ens5 inet dhcp
```

** в данном примере мы настраиваем сетевой интерфейс **ens5** на автоматическое получение IP-адреса.*

4. Применяем настройки загрузчика и перезагружаем систему::

```
update-grub
shutdown -r now
```

Ошибки

1. Error in network definition *.yaml line xxx column yyy: expected mapping

Ошибка появляется при проверке (generate) или применении (apply) настроек сети.

Причина: ошибка синтаксиса YAML.

Решение: внимательно смотрим на количество отступов, которое сделано для строки xxx. Количество пробелов должно точно соответствовать количеству в других строках. Если параметр вложенный, он также должен отделяться от родителя нужным количеством пробелов. Пример неправильной настройки:

```
network:
  version: 2
 renderer: networkd
```

** обратите внимание, что **version** имеет 4 пробела для отступа, а **renderer** — 2. Так как **version** и **renderer** равнозначные параметры для родителя **network**, они должны иметь одинаковое количество пробелов.*

Источники

https://www.dmosk.ru/miniinstruktions.php?mini=network-netplan

https://habr.com/ru/post/448400/

https://losst.ru/nastrojka-seti-netplan-v-ubuntu