КОНТРОЛЛЕР GATEFLOW

Руководство по развертыванию

Версия 3.1.0

Оглавление

Контроллер

Системные требования

Inline SDN контроллер является полностью докеризированным решением. Минимально необходимые требования для развертывания контроллера на хост машине для каждого из 2х узлов кластера и узла графического интерфейса приведены ниже:

Требования к аппаратному обеспечению:

- Число ядер CPU – минимум 8
- Количество памяти – минимум 16 GB
- Дисковое пространство – HDD 128 GB
- Два виртуальных сетевых интерфейса

Требования к программному обеспечению:

- JDK-8
- Python 3.9 и новее
- Ubuntu 22.04.3 LTS и новее
- Docker 24.0.6 и новее

Обратите внимание, каждый из двух узлов SDN контроллера и узел веб-интерфейса может быть развернут внутри виртуальной машины, в этом случае каждая виртуальная машина должна соответствовать требованиям выше

Подготовка к развертыванию и запуск контроллера

Подготовка и запуск контроллера производятся согласно пунктам ниже:

- получение docker образов у разработчика
- установка образов контроллера и Web интерфейса:
  - `docker load < gfn-sdn.tar`
  - `docker load < gfn-ui.tar`
- создание необходимых папок. В основной папке gfn-sdn/ хранится docker-compose.yml файл и .env файл переменных окружения. В папке ignite/ хранится внутренняя база. В папке store/ находятся файлы топологии gateflow.json и лицензии license.key. Например,
  - `mkdir /opt/of-sdn/services/gfn-sdn/`
  - `mkdir /opt/of-sdn/services/gfn-sdn/volumes/ignite/`
  - `mkdir /opt/of-sdn/services/gfn-sdn/volumes/store/`
- создание, подготовка и размещение файлов настроек docker-compose.yml, .env, gateflow.json и gateflow.conf. Шаблоны файлов предоставляет разработчик.

Запуск и остановка непосредственно контроллера и веб-интерфейса производятся следующими командами:

- `sudo su – или sudo –E su`
- `docker compose up –d`
- `docker compose down`

Опция -d означает, что запуск происходит в фоновом режиме.

Файл запуска контейнеров docker-compose.yml и файл переменных .env

Перед запуском приложения должен быть подготовлен файл docker-compose.yml. Примерное содержание файла показано в приложении №1. Точное содержание может варьироваться между

в зависимости от требований. В файле описываются параметры запуска контейнеров контроллера и веб-интерфейса, указаны используемые переменные окружения, внешние ресурсы (volumes), сетевые настройки.

Большинство параметров

Здесь:

- GFN_REMOTEIP — список имен хостов, разделенных точкой с запятой или IP-адреса узлов SDN контроллеров с диапазоном TCP портов. Если необходимо запустить один узел, то GFN_REMOTEIP должен содержать имя хоста или IP-адрес только этого единственного узла
- GFN_LOCALIP – содержит имя хоста или IP-адрес локального узла контроллера SDN
- GFN_NODE – содержит строку инициализации для SDN узла контроллера и в большинстве случаев должен оставаться неповрежденным
- GFN_GW – указывает на имя хоста или IP-адрес 3-го узла, который является арбитром для защиты от split brain. В большинстве случаев выбирается шлюз по умолчанию
- GFN_HTTP, GFN_HTTPS – порты для обработки запросов через REST API
- GFN_OF_PORT – порт для связи с управляемыми коммутаторами
- GFN_CRASH – порт для доступа по ssh на консоль контроллера

В файле .env описываются переменные окружения для запуска контейнеров: названия образов, настройки IP и HTTP, RADIUS и TACACS, параметры аутентификации, местоположение файла лицензий и др.

На запущенном контроллере:

- Для подключения к командной строке внутри docker контейнера используйте команду `docker exec -it <имя_контейнера> sh`
- Для подключения к CLI консоли контроллера используйте ssh на порт указанный в GFN_CRASH параметре
- REST API доступны по портам указанным в GFN_HTTP и GFN_HTTPS

Обратите внимание, по умолчанию контроллер запускается с дефолтными настройками. Для изменения настроек необходимо подготовить новую версию файла конфигурации. Более подробно об этом будет рассказано ниже

Файлы конфигурации

Основными файлами конфигурации SDN контроллера являются gateflow.conf и gateflow.json.

В gateflow.conf содержится общая конфигурация контроллера, в gateflow.json конфигурация сетевых элементов и связей внутри фабрики брокера пакетов.

Файл конфигурации gateflow.conf находится внутри контейнера контроллера в файле конфигурации /gateflow/gateflow.conf. Все параметры конфигурации в конфигурационном файле оптимизированы для большинства сценариев развертывания. Если нужно настроить некоторые конфигурационные параметры для конкретного варианта использования, пожалуйста, обратитесь к обратитесь в службу технической поддержки. Если вы изменяете любой параметр в файле конфигурации, то узел контроллера должен быть перезапущен для применения изменений, которые необходимо реализовать.

Файл конфигурации сети находится в volumes/store/ контейнера контроллера в файле /gateflow.json в формате JSON. Для адаптации конфигурационного файла сети для конкретного сценария развертывания, этот файл должен быть отредактирован в любом текстовом редакторе.

Файл конфигурации сети состоит из 3 разделов:

- Драйверы устройств
- Устройства
- Внутренние линки для построения фабрики

Пример описания драйвера устройства показан ниже:

```
"Accton:AS5812-54X:PicOS-2.11.16-GA": {
    "manufacturer": "Accton",
    "model": "AS5812-54X",
    "software": "PicOS-2.11.16-GA",
    "fromMeterId": 1,
    "toMeterId": 16384,
    "fromGroupId": 5,
    "toGroupId": 512,
    "tablesLimits": {
        "0": 8180,
        "251": 20000
    }
}
```

Пример конфигурации устройства показан ниже:

```
"00:00:00:00:00:00:00:01": {
    "sDpid": "00:00:00:00:00:00:00:01",
    "name": "DR-1",
    "model": "OVS",
    "protocol": "OF13",
    "manufacturer": "Nicira",
    "hardware": "OVS",
    "software": "OVS-2.9.2",
    "driver": "OVS",
    "latitude": 56.970939,
    "longitude": 23.937248
}
```

Пример конфигурации внутренних линков (между коммутаторами) показан ниже:

```
{
    "key": "00:00:00:00:00:00:00:01_3_00:00:00:00:00:00:00:02_3",
    "sources": [
        {
            "switch": "00:00:00:00:00:00:00:01",
            "port": 3
        },
        {
            "switch": "00:00:00:00:00:00:00:02",
            "port": 3
        }
    ],
    "type": "DIRECT",
```

```
    "speed": 10000000,
    "media": "OPTICAL",
    "latency": 1,
    "cost": 1,
    "utilization": 0,
    "state": "UP"
}
```

Здесь:

- Параметр type может принимать значения (регистр важен):
  - DIRECT – прямой канал
  - EXTERNAL – канал через транзитные узлы
  - SERVICE – служебный канал
- Параметр media может принимать значения (регистр важен):
  - OPTICAL - для DWDM канала с keep-alive пакетами для обнаружения падения линка при физическом состоянии порта в UP
  - FIBER - для волокна или меди, без keep-alive
- Параметр state может принимать значение (регистр важен):
  - UP – используется для расчёта пути и отравки/приёма трафика
  - DOWN – канал не используется
  - ADMIN_DOWN – канал не используется

Обратите внимание, что некорректно выставленные параметры в файле конфигурации могут привести к неработоспособности узла контроллера.

Конфигурационные файлы gateflow.conf и gateflow.json не синхронизируются между узлами кластера и настраиваются отдельно для каждого узла.

Отказоустойчивость

Отказоустойчивость достигается за счет использования технологии кластера для узлов контроллера. Для каждого узла в момент развертывания через параметры указывается текущая роль в кластере, это может быть или master или slave. Узлы кластера имеют общую область памяти, т.н. кеш, для синхронизации параметров настройки текущих сервисов.

В момент failover основного узла кластера роль master переходит на резервный узел, при возобновлении работы основного узла к нему возвращается роль master, т.к. автоматически срабатывает функционал preempt.

Для защиты от split brain используется механизм проверки доступности «внешнего арбитра». В качестве арбитра обычно выбирается шлюз по умолчанию, данный параметр настраивается через GFN_GW в момент развертывания узла контроллера.

Принцип работы защиты от split brain - если один из узлов контроллера не видит арбитра, то он переходит в неактивное состояние, восстановление работоспособности узла возможно путем перезапуска контейнера.

Включение функционала split brain происходит через активацию параметра в gateflow.conf:
`net.gateflow.cluster.core.ClusterManager.splitbrainDetection-support=false`

Дополнительные компоненты

При развертывании SDN контроллера существует возможность дополнительной установки сторонних компонентов для сбора и отображения телеметрии устройств.

- cadvisor – для сбора телеметрии с докер-контейнеров
- snmp-exporter – для сбора телеметрии с коммутаторов
- nginx – ssl proxy для доступа к web интерфейсу контроллера через https
- Prometheus – TSBD для хранения собранной телеметрии
- Grafana – отображение собранных данных

Данные компоненты устанавливаются и работают в виде докер-контейнеров. Параметры установки, кастомизации и запуска выходят за рамки описания данной инструкции.

Коммутаторы

Контроллер Gateflow работает с устройствами, поддерживающими протокол OpenFlow версий 1.0, 1.1, 1.3, 1.4 и 1.5. Правила обработки трафика настраиваются на контроллере, который затем разбивает правило на flow для каждого коммутатора на пути прохождения данных и прописывает flow на коммутаторах.

Для успешного подключения и работы коммутатора необходимо:

- обеспечить IP связность между IP адресами управления контроллера и коммутатора
- настроить виртуальный бридж для Openflow на коммутаторе
- настроить Datapath ID (DPID – идентификатор коммутатора в Openflow) на коммутаторе
- настроить IP адрес и порт контроллера на коммутаторе
- настроить физические и виртуальные порты в Openflow режиме и привязать их к бриджу
- внести данные о коммутаторе и линках в файл топологии gateflow.json
- перезапустить контейнер контроллера

Далее для примера будет дана краткая инструкция по первоначальной установке сетевой операционной системы Pica8, более расширенное описание можно найти в соответствующих документах на сайте производителя. IP адреса и порты выбираются исходя из сетевых настроек.

- Предварительно скачиваем образ ОС с сайта производителя и помещаем его на tftp, ftp или http сервер
- Включаем коммутатор и подключаемся к нему через консольный порт
- Во время загрузки выбираем режим ONIE
- Далее выбираем режим ONIE: Rescue для установки новой системы
- Коммутатор попытается получить сетевые настройки для менеджмента интерфейса через dhcp, если в сети нет dhcp необходимо сделать ручные сетевые настройки для связности коммутатора и tftp сервера. (прим - `# ip addr add 10.10.10.10/24 dev eth0`)
- Загружаем новую версию ОС - `onie-nos-install tftp://10.10.10.1/onie-installer-picos.bin`
- После успешной загрузки инсталляции ОС заходим на коммутатор с учетной записью - `admin/pica8`. Система предложит сменить пароль по умолчанию
- Запускаем shell - `admin@Xorplus> start shell sh`
- Запускаем конфигурационный скрипт - `admin@PICOS-OVS:~$ sudo picos_boot`

```
Configure default startup options:
 (Select key 3 if no change)
    [1]  PICOS L2/L3
```

```
        [2]  PICOS Open vSwitch/OpenFlow   * default
        [3]  Quit
    Enter your choice (1,2,3):2

    PICOS Open vSwitch/OpenFlow is selected.

    Configure management interface(IPv4):
        [1] DHCP
        [2] Static    * default
    Enter your choice(1,2):2

      Note: The OVS server, by default, uses static local management IP and port 6640.
            vSwitch, by default, connects to the server via PTCP.

    Set a static IP and netmask for the switch (192.168.1.1/24):10.10.10.10/24
    Set the gateway IP (192.168.1.2):10.10.10.1

    Configure management interface(IPv6):
        [1] DHCP
        [2] Static
        [3] SLAAC
        [4] Disable    * default
    Enter your choice(1,2,3,4):4

    Start OVS web user interface?(y|n)[n]:n

    Start OVS network snmp?(y|n)[n]:y

    Please restart the PICOS service
    admin@XorPlus:~$ sudo systemctl restart picos
```

После перезагрузки Picos коммутатор готов к конфигурированию сетевых параметров

Настройка связи с контроллером

Параметры для связи с контроллером:

- Создать бридж

```
ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8
```

- Прописать уникальный DPID на коммутаторе

```
ovs-vsctl set bridge br0 other_config:datapath-id=0000000000000001
```

Datapath-id (DPID) состоит восьми байт (16 символов) записанных в шестнадцатеричной системе счисления. Отдельные символы могу принимать значения: 0-9, a, b, c, d, e, f. Если вводимое значение не соответствует требованиям, то коммутатор устанавливает DPID самостоятельно, используя значения MAC адреса интерфейса управления eth0 и добавляя к последнему байту 1, 2 или 3 в зависимости от количества служебных интерфейсов на платформе. Два старших бита автоматически дополняются произвольными значениями. Пример автоматически сгенерированного 643aa82bb50d93c9. При этом MAC адрес интерфейса eth0 - a8:2b:b5:0d:93:c8.

- Указать версию протокола OpenFlow для коммуникации с контроллером

```
ovs-vsctl set bridge br0 protocol=OpenFlow10,OpenFlow13
```

- Активировать функционал сбора телеметрии с flow

```
ovs-vsctl set-flow-counter-mode both
```

- Активировать SNMP на коммутаторе

```
ovs-vsctl set-snmp-enable true
```

- Прописать основной и резервный контроллеры

```
ovs-vsctl set-controller br0 tcp:<ip node_master>:6633 tcp:<ip node_slave>:6653
```

Порт контроллера настраивается в параметре GFN_OF_PORT при его развертывании

- Общая команда для активации порта коммутатора

```
ovs-vsctl add-port br0 te-1/1/1 vlan_mode=trunk tag=1 -- set Interface te-1/1/1 type=pica8
options:link_speed=10G
```

Синтаксис и описание параметров командной строки можно найти в соответствующей документации к ОС в разделах PICOS Open vSwitch Configuration Guide и PICOS Open vSwitch Command Reference.

Приложение 1. Пример docker-compose.yml файла

```
services:
  init:
    command:
    - init
    environment:
    - APP_SECRET_KEY
    hostname: init
    image: ${GFN_UI_IMAGE}
    networks:
    - gfn-ui
    restart: "no"
  unicorn:
    command:
    - unicorn
    depends_on:
      init:
        condition: service_completed_successfully
    environment:
    - APP_DEBUG
    - APP_SECRET_KEY
    - GFN_GRAFANA
    - GUNICORN_WORKERS
    - GUNICORN_THREADS
    - GUNICORN_TIMEOUT
    - SDN_ENDPOINT_1
    - SDN_ENDPOINT_2
    - OIDC_CLIENT_ID
    - OIDC_CLIENT_SECRET
    - OIDC_TOKEN_URL
    image: ${GFN_UI_IMAGE}
    networks:
    - gfn-ui
    - gfn-sdn
    restart: unless-stopped
  nginx:
    command:
    - nginx
    depends_on:
    - unicorn
    image: ${GFN_UI_IMAGE}
    networks:
    - gfn-ui
    ports:
    - ${APP_UI_PORT:-80}:80
    restart: unless-stopped
  gfn-sdn:
    environment:
    - GFN_ROLE
```

```
    - GFN_ID
    - GFN_LOCALIP
    - GFN_REMOTEIP
    - GFN_NODE
    - GFN_GW
    - GFN_AUTH_PROTOCOL
    - GFN_RADIUS_HOSTS
    - GFN_RADIUS_SECRET
    - GFN_RADIUS_AUTH_PORT
    - GFN_RADIUS_ACCT_PORT
    - GFN_TACACS_HOST
    - GFN_TACACS_KEY
    - GFN_TACACS_SERVICE_ADMIN
    - GFN_TACACS_SERVICE_USER
    - GFN_LICENSE_KEY_FILE
    - GFN_LICENSE_TILL
    - GFN_LICENSE_SWITCHES
    - GFN_LICENSE_SALT
    - GFN_MODULE_LOADING_ORDER
    - JAVA_ARGS
    - JAVA_HEAP_ARGS
    - OIDC_CLIENT_UUIDS
    - OIDC_JWT_SECRET
    - OIDC_TOKEN_ACCESS_TTL
    - OIDC_TOKEN_REFRESH_TTL
    image: ${GFN_SDN_IMAGE}
    networks:
    - gfn-sdn
    ports:
    - ${GFN_OF_PORT:-6653}:6653
    restart: unless-stopped
    volumes:
    - $PWD/volumes/ignite:/gateflow/ignite
    - $PWD/volumes/store:/gateflow/store
networks:
  gfn-ui:
    name: gfn-ui
  gfn-sdn:
    name: gfn-sdn
```