

# Annet



Это менеджер конфигурации для сетевых устройств.

С помощью Annet можно сгенерировать целевую конфигурацию оборудования, рассчитать разницу между текущей и желаемой, сформировать и применить патч на устройстве.

Написана на Python, удобно расширять.



## Список основных команд

1

```
annet gen lab-r1.nh.com
```

```
interface GigabitEthernet1/0
description to_router2_Gi2/0
mtu 4000

ip ip-prefix PRFX_LOOPBACKS index 5 permit
172.16.1.0 24 greater-equal 32 less-equal 32
```

### Полезные аргументы:

`-g` — список генераторов для которых генерируется конфигурация

`-G` — список генераторов, которые необходимо исключить из генерации конфигурации

`--annotate` — показывать какая строка генератора генерирует команду

2

```
annet diff lab-r1.nh.com
```

```
interface GigabitEthernet1/0
+   description to_router2_Gi2/0
+   mtu 4000

- ip ip-prefix PRFX_LOOPBACKS index 5 permit
172.16.0.0 24 greater-equal 32 less-equal 32
+ ip ip-prefix PRFX_LOOPBACKS index 5 permit
172.16.1.0 24 greater-equal 32 less-equal 32
```

3

```
annet patch lab-r1.nh.com
```

```
interface GigabitEthernet1/0
description to_router2_Gi2/0
mtu 4000
exit

undo ip ip-prefix PRFX_LOOPBACKS index 5
ip ip-prefix PRFX_LOOPBACKS index 5 permit
172.16.1.0 24 greater-equal 32 less-equal 32
```

4

```
annet deploy lab-r1.nh.com
```

### Полезные аргументы:

`--log-level debug` — вывод отладочной информации на экран

`--no-ask-deploy` — применение конфигурации без подтверждения

## Генератор

Это python-класс, генерирующий строки конфигурации оборудования в зависимости от входных параметров (производитель, софт и другие атрибуты в инвентарной системе).

Класс имеет два типа методов:

`acl` для описания зоны действия генератора

`run` для непосредственного создания строк конфигурации.

Методы определяются в генераторе в зависимости от производителя оборудования согласно шаблону `acl_<vendor>, run_<vendor>`.

```
class IfaceDescriptions(PartialGenerator):
```

```
    TAGS = ["description"] # можем использовать с ключами -g, -G
```

```
# Определяем зону действия генератора для Cisco
```

```
    def acl_cisco(self, device):
```

```
        return ""
```

```
        interface %cant_delete
```

```
            description
```

```
        """
```

```
# Функция ниже отвечает за генерацию конфигурации для Cisco
```

```
    def run_cisco(self, device):
```

```
        for interface in device.interfaces:
```

```
            with self.block
```

```
(f"interface {interface.name}":
```

```
            yield "description test"
```

# ACL

ACL определяет зону действия генератора. Генератор может выдавать только команды, подходящие под ACL.

Реальная конфигурация с устройства фильтруется через ACL перед сравнением с выводом генератора.

```
def acl_cisco(self, device):
    return """
    interface # зона действия - все интерфейсы
        description # любые строки после
description попадут под ACL
    """
```

В строках ACL можно использовать специальные выражения.

```
# Матчит одно слово ([^\s]+ в терминах regexp)
dns domain *

# Матчит любое ненулевое количество слов (+)
header login information ~

# Литералы можно комбинировать в строке
info-center source * channel *

# Запрет на удаление строки генератором
%cant_delete
```

# Gnetcli

Инструмент для работы с командным интерфейсом сетевого оборудования через SSH и Telnet. Поддерживает интерактивное взаимодействие, обработку ошибок и пагинацию. Реализована работа с CLI основных производителей сетевого оборудования «из коробки». Написан на Golang и отличается высокой скоростью работы.

## Сценарии использования

### Golang-библиотека

**Когда:** в проектах автоматизации на Golang



**Как:** по аналогии с примером из документации

Подробнее можно ознакомиться в статье



### CLI-утилита

**Как:**

1 Установить утилиту cli:

```
go env -w GOBIN=$HOME/go/bin; go install
github.com/annetutil/gnetcli/cmd/cli@latest
```

2 Пример получения вывода команды dis clock на устройстве test.lab.net с выводом в json-формате. Аутентификация на оборудовании согласно настройкам из ~/.ssh/config:

```
~/go/bin/cli -hostname test.lab.net -
devtype huawei -command 'dis clock' -use-
ssh-config -json
```

### gRPC-сервер

**Когда:** во всех сценариях где используется SOA (Service-oriented architecture) подход к автоматизации

**Как:**

1 Установить gnetcli\_server:

```
go env -w GOBIN=$HOME/go/bin; go install
github.com/annetutil/gnetcli/cmd/
gnetcli_server@latest
```

2 Добавить приватный ключ для доступа на оборудование в ssh-agent:

```
eval $(ssh-agent -s); ssh-add
<путь к ключу>
```

3 Запустить сервер с HTTP Basic аутентификацией gRPC запросов и аутентификацией по ключу пользователя netadmin из ssh-агента на оборудовании:

```
~/go/bin/gnetcli_server -basic-auth
mylogin:mysecret -dev-login netadmin -dev-
use-agent
```

4 Пример выполнения команды dis clock на устройстве test.lab.net:

```
grpcurl -H "Authorization: Basic $(echo -n
"mylogin:mysecret" | base64)" -plaintext -d
'{"host": "test.lab.net", "cmd": "dis
clock", "string_result": true,
"host_params": {"device": "huawei"}}'
localhost:50051 gnetcli.Gnetcli.Exec
```



# Лабораторный стенд



## Список сценариев

- |   |   |
|---|---|
| <b>Lab00.</b><br><b>Basic - Cisco</b>                   | <ul style="list-style-type: none"><li>• Познакомимся с принципом работы <code>PartialGenerator</code></li><li>• Применим настройки MTU и <code>description</code> на Cisco</li><li>• Данные будем брать из Netbox</li></ul> |
| <b>Lab01.</b><br><b>Basic - FRR</b>                     | <ul style="list-style-type: none"><li>• Будем использовать <code>Entire</code> генераторы на FRR</li><li>• Настроим BGP-пикинги</li></ul>   |
| <b>Lab10.</b><br><b>DC - Cisco</b>                      | <ul style="list-style-type: none"><li>• Рассмотрим продвинутый подход к генерации PoD'а сети</li><li>• Будем управлять генерацией конфигурации оборудования, используя роли устройства</li></ul>                            |
| <b>Lab11.</b><br><b>DC - FRR</b>                        | <ul style="list-style-type: none"><li>• То же самое, что в Lab10, но на FRR</li></ul>   |
| <b>Lab12.</b><br><b>DC - Arista</b><br><b>Cisco FRR</b> | <ul style="list-style-type: none"><li>• Попробуем управлять конфигурацией на мультивендорной сети</li></ul> <p><b>Внимание! Запускается только на Linux</b></p>   |

## Подготовка:

- 1** Установите `docker`на ваше устройство.
- 2** Поместите образы ПО в папку `vm_images`.
- 3** Скачайте репозиторий и соберите лабораторное окружение:  

```
git clone https://github.com/annetutil/contribs.git
cd contribs/labs
make build
```
- 4** Запускайте лабораторные сценарии командой:  

```
make labXX # где XX – номер сценария
```
- 5** После завершения работы со сценарием:  

```
make services_stop
```

## Как попасть в Netbox UI:

- `http://localhost:8000`
- `login: annet`
- `password: annet`

## Как попасть в контейнеры:

- `annet:`  
`docker exec -u root -t -i annet /bin/bash`
- `netbox:`  
`docker exec -u root -t -i netbox /bin/bash`

## Как попасть на оборудование:

- Cisco:  
`telnet — annet/annet`
- FRR, Arista:  
`ssh — annet/annet`

## Подготовка окружения

### Список контейнеров:

Netbox	Network Inventory
Annet	Network Configuration Manager
Gnetcli	Network Configuration Executor
Devices	Network Emulator

# Пример прохождения сценария lab00

1

Добавляем в папку `vm_images` образ `c7200-jk9s-mz.124-13a.bin`

Найти и скачать его (как и другие образы ОС) вам предлагается самостоятельно

2

Собираем стенд командой

```
make lab00
```

После этого шага произойдёт автоматический вход внутрь контейнера с `annet`. Если этого не произошло или вы вышли, воспользуйтесь командой

```
docker exec -u root -t -i annet /bin/bash
```

Через некоторое время появится доступ на роутеры.

3

Генерируем желаемую конфигурацию

```
annet gen lab-r1.nh.com lab-r2.nh.com  
lab-r3.nh.com
```

4

Смотрим разницу между желаемой и реальной конфигурацией

```
annet diff lab-r1.nh.com lab-r2.nh.com  
lab-r3.nh.com
```

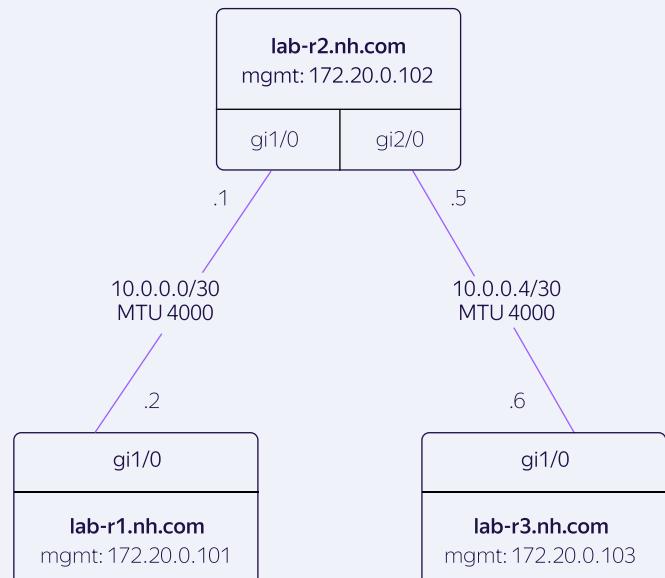
5

Готовим патч командой

```
annet patch lab-r1.nh.com lab-r2.nh.com  
lab-r3.nh.com
```

При желании, можно:

- Попробовать поменять параметры MTU в Netbox или правила генерации `description` в генераторе `./src/lab_generators/interfaces.py` и убедиться, что изменения отражаются в `annet diff`
- Вручную изменить конфигурацию на устройстве и убедиться, что `annet` генерирует `diff`



Annet



Gnetcli



Labs

6

Применяем конфигурацию

```
annet deploy lab-r1.nh.com lab-r2.nh.com  
lab-r3.nh.com
```



Community-чат



Онлайн версия