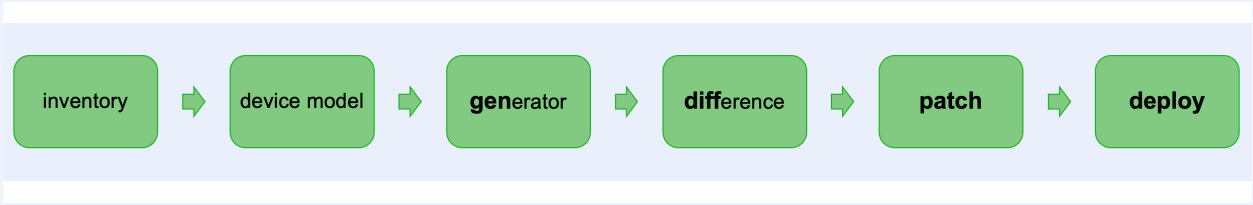
Александр Балезин

Annet

Обновлено 24 ноября 2024, 18:04

Примечание

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

`annet gen lab-r1.nh.com`

```
interface GigabitEthernet1
  description to_router2_G
  mtu 4000

ip ip-prefix PRFX_LOOPBACK
 permit 172.16.1.0 24 grea
```

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

`annet diff lab-r1.nh.com`

```
   interface GigabitEtherne
+    description to_router2
+    mtu 4000

- ip ip-prefix PRFX_LOOPBA
  permit 172.16.0.0 24 grea
+ ip ip-prefix PRFX_LOOPBA
  permit 172.16.1.0 24 grea
```

`annet patch lab-r1.nh.com`

```
interface GigabitEthernet1
  description to_router2_G
  mtu 4000
  exit

undo ip ip-prefix PRFX_LOO
ip ip-prefix PRFX_LOOPBACK
 permit 172.16.1.0 24 grea
```

`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:

1. `acl` для описания зоны действия генератора
2. `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  # для всех интерфейсов, при этом не даем их удалять
            description          # разрешаем задавать description, а остальное - нет
        """

    # 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.

```
dns domain *                    # матчит одно слово ([^\s]+ в терминах regexp)
header login information ~      # матчит любое ненулевое количество слов (.+)
info-center source * channel *  # литералы можно комбинировать в строке
%cant_delete                    # запрет на удаление строки генератором
```

Gnetcli

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

Когда: в ручной эксплуатации и автоматизации на отличных от Golang языках

Как:

1. Установить утилиту `cli`

```
go env -w GOBIN=$HOME/go/bin;\
  go install github.com/annetutil/g
```

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

```
~/go/bin/cli -hostname test.lab -de
```

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 githu
```

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 аутентификаций gRPC запросов и аутентификаций по ключу пользователя `netadmin` из ssh-агента на оборудовании:

```
~/go/bin/gnetcli_server -basic-auth mylogin:my
```

4. Пример выполнения команды `dis clock` на устройстве `test.lab.net` (производитель Huawei)

```
grpcurl -H "Authorization: Basic $(echo -n "my
 -plaintext -d '{"host": "test.lab.net", "cmd"
```

```
"string_result": true, "host_params": {"device
 localhost:50051 gnetcli.Gnetcli.Exec
```