distributor Documentation

Выпуск 2.3

m_messiah

28 April 2016

“Покажи мне где твои сервисы и я скажу, что ты ошибаешься.”

Когда у вас много серверов-фронтов, попытка уследить какие адреса и приложения слушаются на каком из них становится большой головнй болью, в особенности, если необходимо часто перемещать различные приложения между разными группами балансировщиков. Более того, при таком количестве доменов, уследить за их актуальности также не представляется возможным без использования автоматизации.

Distributor - это утилита, которая получает конфигурацию Nginx и Haproxy, умеет получать трансферы DNS зон, и получать список доменов с NIC.ru, после чего строит из всего этого удобную вебстраничку с соответсвиями между сервисами.

Глава 1

Как это работает?

В текущей версии **Distributor** умеет забирать конфигурационные файлы с GitLab, куда балансировщики пушат свою конфигурацию.

Если возможности и желания реализовывать такую схему нет, необходима кастомизация кода.

1.1 Конфигурация

Вся настройка производится через ini файл, передаваемый в параметре `--conf (-c)`

1.1.1 GitLab

```
[git]
# Адрес gitlab сервера
host = git.example.com
# Ваш API токен, для доступа к репозиториям
token = ""
# GitLab-группа, в которой размещены репозитории с конфигами
group = ""
# Список серверов-названий проектов в git
servers = eu-front1,eu-front2,us-front1,us-front2,int-front
# Смерджит конфиги eu-front1 и eu-front2, отображая их как "eu"
# Отображает имя сервера на .group(1) - то, что в скобках
same_host = (\w\w)-front\d
# Если skipped в названии сервера, то к nginx не будут применяться проверки доступности. (например, если там локаль
skipped = int
```

Nginx

Чтобы корректно отображалась колонка автора в **Distributor** необходимо, чтобы хотя бы в одной строке внутри блока `server {}` в конце строки присутсвовал комментарий с текстом `author: <author>`. Например, удобно обозначать e-mail автора для более быстрого рагирования по инцидентам:

```
server {
    server_name www.example.com; # author: webmaster@example.com
    listen 80;
    location / {...};
}
```

Отдельные приложения можно вынести в группу `promo` и они будут более тщательно проверяться **Distributor**. Маркер promo также необходимо использовать в комментариях в секции `server {}`. Можно расширить предыдущий пример:

```
server {
    server_name www.example.com; # promo author: webmaster@example.com
    listen 80;
    location / {...};
}
```

Осуществляемые проверки описаны на странице *Проверки*

1.1.2 BIND/DNS

Сейчас DNS трансфер зон осуществляется с использованием TSIG ключа:

```
[dns]
server = ns.example.com
tsig_type = HMAC-SHA512
tsig_name = TSIGKEY
tsig_key = STRONGKEY==
# Список зон, которые необходимо трансферить и отображать.
domains = example.com,example.net
```

1.1.3 NIC.ru

Возможно использование **Distributor** для получения информации о делегировании доменов, зарегистрированных в nic.ru, с последующей проверкой их SOA на соответствие реальности и проверкой на наличие SPF и DMARC записей у зон. Для этого необходимо вписать свой логин и пароль от ЛК nic.ru (на данный момент у них нет АПИ, позволявшего бы получить ту же информацию):

```
[nic]
login = 123456
password = password
```

1.2 Использование

1.2.1 Установка

```
pip install distributor
```

Может использоваться как с Python2.7+, так и с Python3+.

Кроме того, вам может понадобиться веб-сервер для раздачи статических html файлов.

1.2.2 Запуск

```
sudo -u www-data distributor-gen -c config.ini -l distributor.log -o /var/www/
```

1.2.3 Пример использования

```
# /etc/crontab
# Daily distributor-gen
0 5 * * * nginx /usr/bin/distributor-gen -c /etc/distributor.ini -o /var/www/distributor -l /var/www/distributor/lo
```

1.3 Проверки

1.3.1 Nginx

Все `server {}` из nginx делятся на два типа: HTTP и TCP Stream, валидацию проходят только первые.

HTTP

Для веб-сервисов, не помеченных как промо проверятся:

0. Определено точное доменное имя или .example.com
1. Доступность в течение 5 секунд
2. Происходил ли редирект с указанного server_name куда-либо еще при обращении на `<scheme>://server_name/`
3. Присутствует ли в заголовках `X-Powered-By`
4. Присутсвуют ли дублирующиеся заголовки в ответе (только полное совпадение)
5. Не было ли проблем с сертификатом, если подключение по https.

Promo

Если сервер помечен как промо (смотри *Nginx*), то в дополнение к обычным проверкам http добавляются следующие:

6. **Доступность и правильный mime-type:**
   (a) /favicon.ico
   (b) /robots.txt
   (c) /sitemap.xml
7. **Присутствие на главной странице тегов:**
   (a) `<h1>`
   (b) `<title>`
   (c) `<meta name="description"...`

1.3.2 HAProxy

Сервисы Haproxy проверяются лишь по названию (поиск ключевых слов) или по известным портам, вследствие чего раскладываются по категориям:

- ssh

- sql
- rdp
- mail
- http
- ldap
- sms

1.3.3 NIC.ru

Если же включено получение списка доменов с nic.ru, то к ним применяются следущие проверки:

1. Делегирован или нет
2. Все ли NS, указанные в зоне отвечают и знают, что они авторизованные.
3. Присутствует ли TXT запись с v=spf1 и нет ли в ней +all
4. Присутвует ли TXT запись с v=DMARC1 ...

1.4 API

По ссылке */categories/api.json* генерируется json с данными об авторах и формате логов по сервисам.

Структура представлена ниже:

- api.json словарь + web словарь
  - log_format - словарь форматов логов {имя -> формат}
  - authors - словарь авторов {имя -> список сервисов}
  - servers - словарь сервисов + название сервиса -> словарь
    - log_format - формат
    - log - путь к логу
    - author - автор

1.5 Замена GitLab

`fetch(`*configs_dir*`)`
Функция, в которой осуществляется вся предварительная выгрузка конфигурационных файлов для анализа.

В секции GitLab вызывается функция `fetch_git()`, которая отвечает за выгрузку из Git. Можно исправить ее, или добавить свою секцию в конфигурационный файл и дописать еще одну функцию.

Например, для прямого получения по ssh можно использовать что-то вроде:

```
def fetch_ssh(self):
    from subprocess import call, check_output

    # Imported in __init__.py
    # from os.path import join as pjoin
```

```
    for server in servers:
        call([
            "scp",
            "%s.example.com:/etc/nginx/nginx.conf" % server,
            pjoin(self.configs, "nginx.%s.all" % server)
        ])
        config = open(pjoin(self.configs, "nginx.%s.all" % server), "r").read()

        with open(pjoin(self.configs, "nginx.%s.all" % server), "a") as main:
            for incl in re.findall(r"(?:^i|^ +i)nclude (.+);$", config, re.M):
                inc_conf = check_output([
                    "ssh", "%s.example.com" % server,
                    "cat", "/etc/nginx/%s" % incl
                ])
                config = config.replace("include " + incl + ";", inc_conf)
            main.write(conf)

        call(["scp",
              "%s.example.com:/etc/haproxy/haproxy.cfg" % server,
              pjoin(self.configs, "haproxy.%s.all" % server)])
```

F