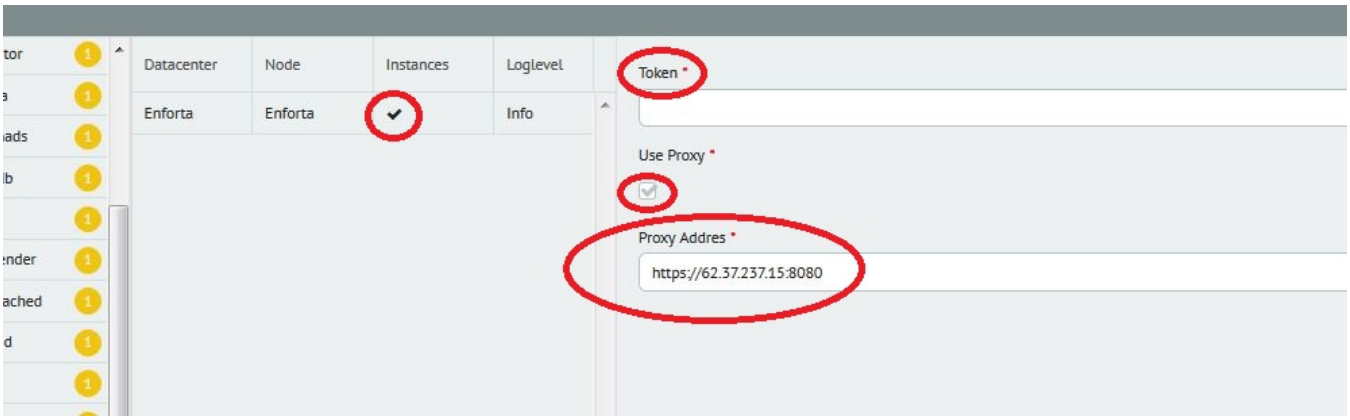
Отправка уведомлений в Telegram

Сделал через башню.

Сервис ставится автоматом из башни, достаточно сделать Pull и активировать сервис.

В сервисе реализовано использование proxy.



Как это было собрано.

В **/opt/noc/ansible/config/services.yml** нужно добавить в двух местах

Перед config

```
tgsender:

  description: Tgsender

  level: global
```

и после

```
tgsender:
  use_proxy:
    label: Use Proxy
    type: bool
    default: false
  token:
    label: Token
    type: str

  proxy_addres:
    label: Proxy Addres (https://ip:port) If use auth (https://us:pass@ip:port)
    type: str
```

В **/opt/noc/ansible/site.yml** добавил роль

```
  - role: tgsender

    become: yes

    when: "{{ has_svc_tgsender | default(False) }}"

    tags:

      - tgsender

      - node

      - noc
```

Копируем каталог **ansible/roles/mailsender** в **ansible/roles/tgsender** и правим файлы.

Так же создаем папку

mkdir ansible/roles/tgsender/handlers и файл в ней

touch ansible/roles/tgsender/handlers/main.yml

nano ansible/roles/tgsender/handlers/main.yml

```
---
- name: reload telegraf
  service:
        name: telegraf
        state: restarted
```

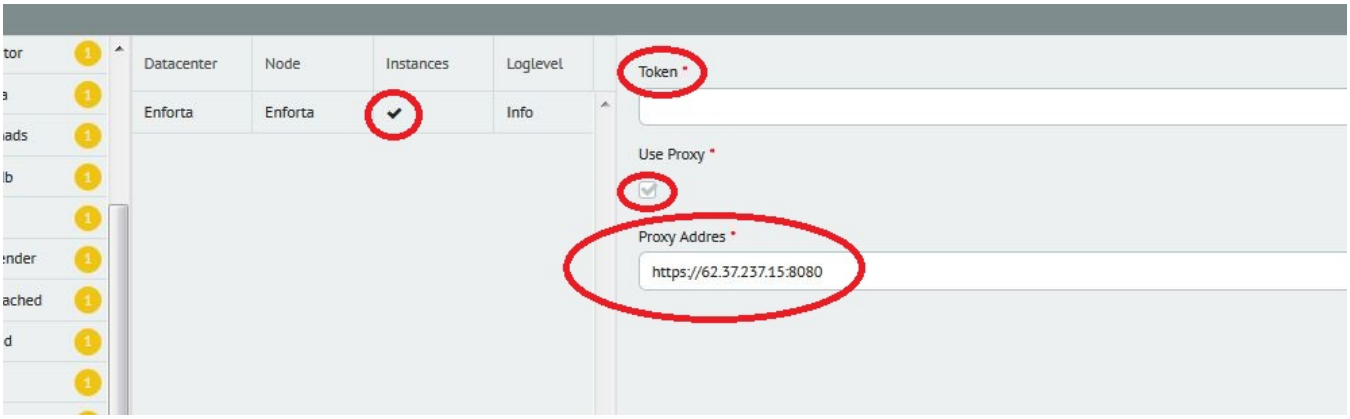
Это для отрисовки графиков в grafan как я понял.

Тоже самое делаем и в папке, но только после рестарта и Pull!!!!

/opt/tower/var/tower/playbooks/STAGE/ansible/config/services.yml

Если вы сделали верно, то у вас появится вот поля:

В сервисе реализована возможность отправки сообщений через прокси.



Сам сервис отправки сообщения в **Telegram**
Для начала нужно создать сам сервис

```
    /opt/noc# mkdir services/tgsender
    /opt/noc# touch services/tgsender/__init__.py
    /opt/noc# touch services/tgsender/service.py
    /opt/noc# chmod a+x services/tgsender/service.py
```

Сам сервис располагается в файле **services/tgsender/service.py**

```
#!./bin/python
# -*- coding: utf-8 -*-
##----------------------------------------------------------------------
## mailsender service
##----------------------------------------------------------------------
## Copyright (C) 2007-2016 The NOC Project
## See LICENSE for details
##----------------------------------------------------------------------
## Third-party modules
import datetime
import socket
import json
import urllib
import urllib2
import time
## NOC modules
from noc.core.service.base import Service
from noc.core.perf import metrics

class TgSenderService(Service):
    name = "tgsender"
    process_name = "noc-%(name).10s-%(instance).3s"
    def on_activate(self):
        self.subscribe(
            topic=self.name,
            channel="sender",
            handler=self.on_message
        )
    def on_message(self, message, address, subject, body, attachments=None, **kwargs):
        self.logger.info(
            "[%s] Receiving message: %s (%s) [%s, attempt %d]",
            message.id, subject, address,
            datetime.datetime.fromtimestamp(
                message.timestamp / 1000000000.0
            ),
            message.attempts
        )
        return self.send_tb(message.id, address, subject, body)
    def send_tb(self, messages, address, subject, body):
        RETRY_TIME = 2.0
        token = self.config.token
        proxy_addres = self.config.proxy_addres
        data = {'chat_id': address, 'text': body}
        time.sleep(RETRY_TIME)
        if self.config.use_proxy:
            try:
                proxy = urllib2.ProxyHandler({'https': proxy_addres})
                auth = urllib2.HTTPBasicAuthHandler()
                opener = urllib2.build_opener(proxy)
                urllib2.install_opener(opener)
                result = urllib2.urlopen("https://api.telegram.org/bot" + token + "/sendMessage",
                                         urllib.urlencode(data)).read()
                check = json.loads(result)
                self.logger.info("Proxy Send: %s\n" % check)
                metrics["telegram_proxy_sended_ok"] += 1
                return True
            except urllib2.HTTPError, e:
                self.logger.info("Proxy HTTPError: %s\n" % e.code)
                metrics["telegram_proxy_failed_httperror"] += 1
```

```
                return False
            except urllib2.URLError, e:
                self.logger.info("Proxy URLError: %s\n" % e.args)
                metrics["telegram_proxy_failed_urlerror"] += 1
                return False
            except urllib2.HTTPException, e:
                self.logger.info("Proxy HTTPException: %s\n" % e.err)
                metrics["telegram_proxy_failed_urlerror"] += 1
                return False
            except Exception, e:
                self.logger.info("Proxy Generic Exception: %s\n" % e.exp)
                metrics["telegram_proxy_failed_exceprion"] += 1
                return False
        else:
            try:
                result = urllib2.urlopen("https://api.telegram.org/bot" + token + "/sendMessage",
                                         urllib.urlencode(data)).read()
                check = json.loads(result)
                self.logger.info("Send: %s\n" % check)
                metrics["telegram_sended_ok"] += 1
                return True
            except urllib2.HTTPError, e:
                self.logger.info("HTTPError: %s\n" % e.code)
                metrics["telegram_failed_httperror"] += 1
                return False
            except urllib2.URLError, e:
                self.logger.info("URLError: %s\n" % e.args)
                metrics["telegram_failed_urlerror"] += 1
                return False
            except urllib2.HTTPException, e:
                self.logger.info("HTTPException: %s\n" % e.err)
                metrics["telegram_failed_urlerror"] += 1
                return False
            except Exception, e:
                self.logger.info("Generic Exception: %s\n" % e.exc)
                metrics["telegram_proxy_failed_exceprion"] += 1
                return False

if __name__ == "__main__":
    TgSenderService().start()
```

Для того, чтобы NOC узнал про новый метод отправки, необходимо поправить файлик **main/models/notificationgroup.py** и заменить

```
    NOTIFICATION_TOPICS = {
        "mail": "mailsender"
    }
```

на

```
    NOTIFICATION_TOPICS = {
        "mail": "mailsender",
        "tg": "tgsender"
    }
```

Файлы supervisord
`/opt/noc# sed 's/mailsender/tgsender/' < etc/defaults/services/mailsender.conf > etc/defaults/services/tgsender.conf`

После всего этого выбираем в окружении способ обновления **Evaluation** и делаем Deploy.

Для тех кто хочет запустить в ручном режиме.
Правим **/opt/noc/etc/noc.yml**

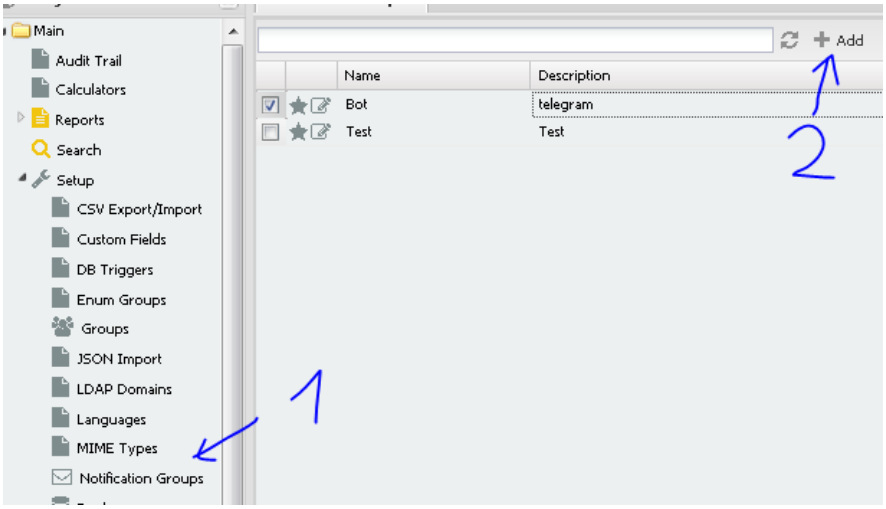
```
tgsender-global-STAGE:
    token:
    global_n_instances: 1
    global_offset: 0
    listen:  ip: ,     190
    loglevel: info
    n_instances: 1
```

Затем правим **opt/noc/etc/supervisord.conf**

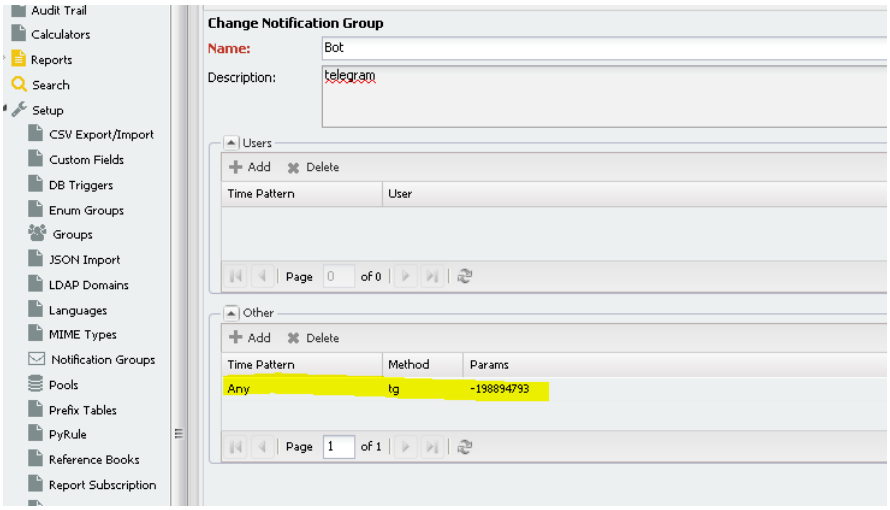
```
 [program:tgsender]
    command = ./services/tgsender/service.py --instance=%(process_num)d
    process_name = %(program_name)s-%(process_num)03d
    numprocs = 1
    umask = 022
    priority = 610
    autostart = true
    autorestart = unexpected
    startsecs = 2
    startretries = 999
    exitcodes = 0
    stopsignal = TERM
    stopwaitsecs = 10
    stopasgroup = false
    killasgroup = true
    user = noc
    redirect_stderr = true
    stdout_logfile = var/log/%(program_name)s-%(process_num)03d.log
    stdout_logfile_maxbytes = 10MB
    stdout_logfile_backups = 3
    stdout_events_enabled = false
    stderr_logfile = var/log/%(program_name)s-%(process_num)03d.err
    stderr_logfile_maxbytes = 1MB
    stderr_logfile_backups = 3
    stderr_capture_maxbytes = 1MB
    stderr_events_enabled = false
    environment = LD_PRELOAD="/usr/lib64/libjemalloc.so.1",NOC_NUMPROCS="1",NOC_USER="noc",NOC_ROOT="/opt/noc",
NOC_ENV="STAGE",NOC_LOGLEVEL="info",NOC_DC="STAGE",NOC_NODE="STAGE",NOC_LISTEN=" IP:190"
```

В консоли запускаем ./noc ctl и вводим "update". новый сервис должен подгрузиться, затем его стартуем.

В самом NOCe создаем Bota в Notification Groups



И заполняем поля



В поле Params указываем chat_id, обязательно с "-", получить его можно либо через запрос бота, либо через вебверсию телеграмма

Если вы сделали все правильно, то у вас должен появится сервис в ./noc ctl

```
syslogcollector-default                RUNNING   pid 20787, uptime 2 days, 7:24:38
tgsender:tgsender-000                  RUNNING   pid 30709, uptime 7:33:58
trapcollector-default                  RUNNING   pid 20785, uptime 2 days, 7:24:38
```

И в NSQ https://ваш IP::4171 появится очередь

Streams / tgsender

Topic: **tgsender**

Empty Queue | Delete Topic | Pause Topic

Topic Message Queue

NSQd Host	Depth	Memory + Disk
✗ [redacted]	0	0 + 0
Total:	0	0 + 0

Channel Message Queues

Channel	Depth	Memory + Disk	In-Flight	Deferred	Requeued
sender	0	0 + 0	0	0	239

Для проверки бота через ./noc shell

```
import logging
  ...: logging.basicConfig(level=logging.DEBUG)
  ...: from noc.main.models.notificationgroup import NotificationGroup
  ...: g = NotificationGroup.get_by_id(X) -  X -  ,   Notification Groups.
  ...: print g.notify(" ", "")
```

Отображение результата работы сервиса.