Миграция на NOC Tower

Требования

Тестирование производилось для версии NOCa 0.7 и выше.

Подготовка

Если у вас уже есть установленный NOC Tower и развёрнутый NOC данный раздел можно пропустить и перейти к разделу Развёртывание бэкапов.

Установка NOC Tower

NOC Tower это средство для автоматизации развёртывания NOCa. Используется во всех новых версиях с его появлением установка NOCa без его использования не поддерживается.

Сам Tower - это веб-интерфейс для настроек. Рекомендумая процедура установки через Docker контейнер.

- FAQ по установке Tower (Башни)
- Установка NOC Tower (башня)

Архив:

- Установка NOC Tower на Debian 8.2
- Установка NOC Tower на Debian 8.4
- Установка NOC Tower на FreeBSD

Deploy

После установки NOC Tower необходимо настроить окружение. Данная процедура заключается в описании хостов, на которые будут установлены компоненты, необходимые для работы NOC'a.

Сама процедура описана в руководствах по установке. Замечу только, что в случае сомнений, выбирайте значение по умолчанию (Install Everything) - устанавливать всё. В последствии разнести какие-то компоненты на отдельные машины проблемы не составляет.

При первоначальной установке необходимо в опциях выбирать пункт Everything. После установки необходимо проверить работоспособность NOC'a. Внести пару хостов, посмотреть что всё нормально запускается. Все данные занесенные данные, во время миграции будет удалены🙂. Если они необходимы, можно либо сделать бэкап БД, либо сделать экспорт в csv.

Получения бэкапов

Если в вашем окружении БД NOC'a вынесены на отдельные хосты и вы не хотите менять данную традицию, то для PosgreSQL есть возможность сделать клон базы. Это делается командой:

CREATE DATABASE <pg_db_new> WITH TEMPLATE <pg_db_old> OWNER <pg_user>;

где <pg_db_new> - бд для нового НОК'а

<pg_db_old> - имя БД для старого НОК'а

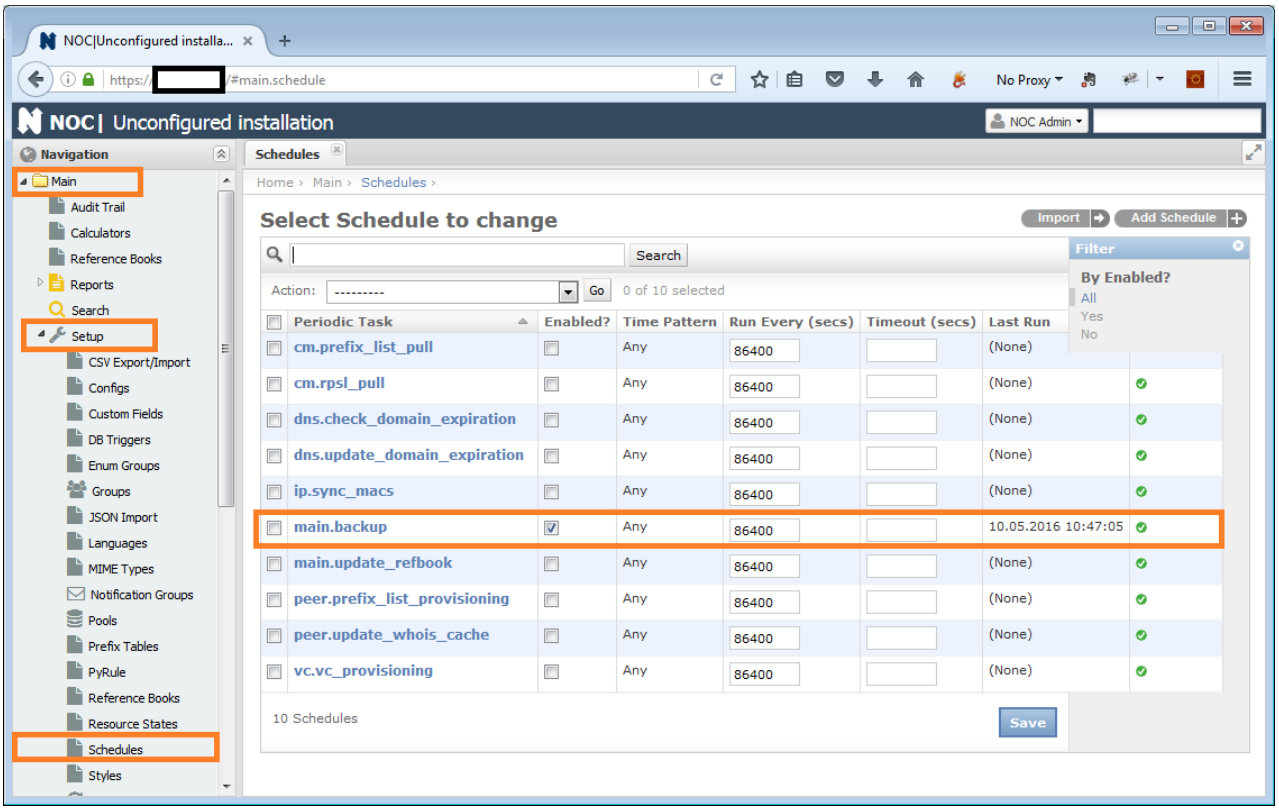
<pg_user> - пользователь, с которым будет подключиться новый НОК

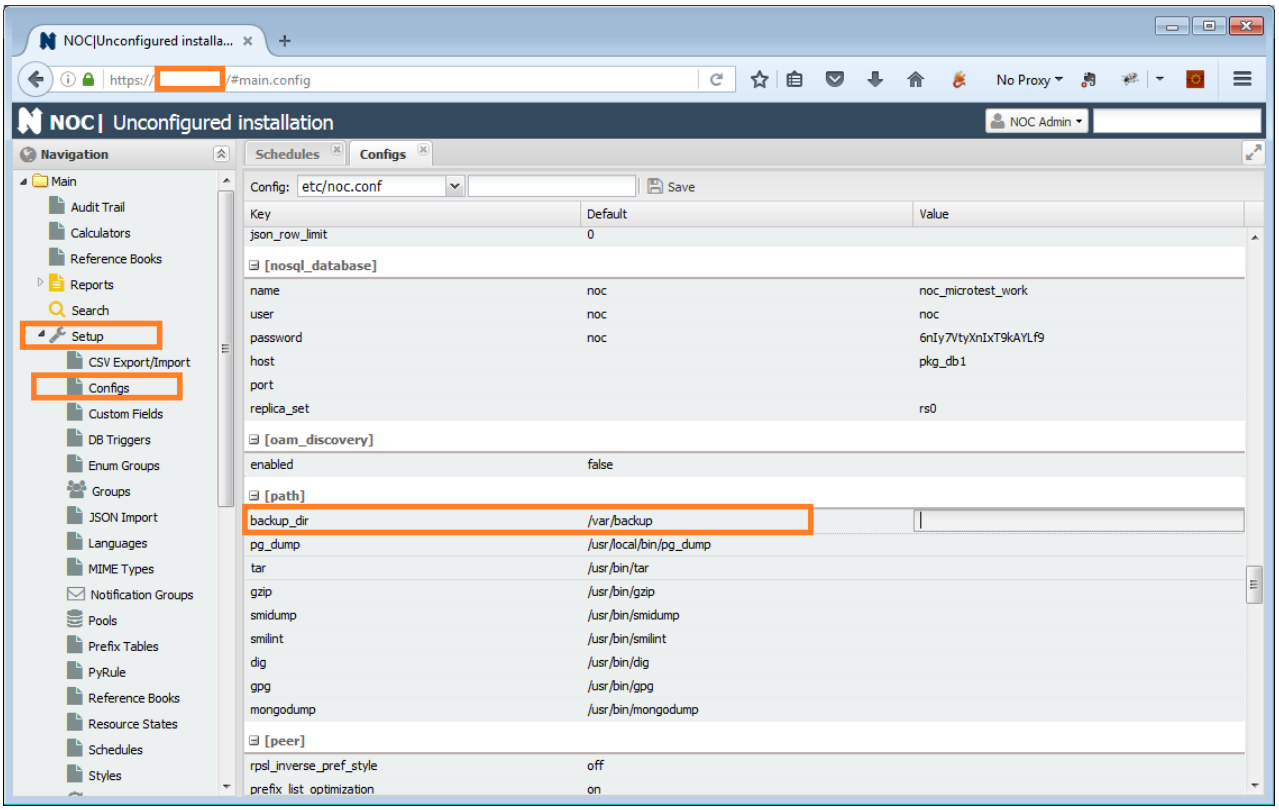
Для процедуры миграции понадобятся дампы баз данных со "старого" NOC'a: PostgreSQL и MongoDB. Миграция данных по производительности (Performance) не предусмотрено. По умолчанию бэкапы НОКа не включены. Проверить это можно зайдя в пункт меню Main -> Setup -> Schedulers. В столбце Enabled? должна стоять галочка. Зелёный значок справа показывает, что бэкап выполнен успешно. Если бэкапы не выполняются (галочка не установлена) - необходимо их включить. Если значок красного цвета - необходимо разобраться, что не работает.

Вероятные причины не выполнения бэкапов:

- отсутствие директории на диске или неправильные права на неё (владельцем должен быть пользователь noc), куда настроены выполняться бэкапы. Настройки бэкапов находятся в файле **noc.conf**. Посмотреть их можно в пункте Main -> Setup -> Configs.
- не правильный путь до исполняемых файлов pg_dump и mongodump (проверить noc.conf раздел path)
- бэкап выполняется, но нет файлов бэкапа баз postgres (pg_dump*), необходимо дать пользователю noc права superuser на базу (или выполнить бэкап вручную, командой "pg_dump -Fc -f /tmp/noc-db-bkp.dump -U noc noc", из-под пользователя с правами superuser на базу).

Бэкап пишет логи в файл **noc-scheduler.log** под процессом main.backup. Там можно посмотреть причину не выполнения бэкапа.





Развёртывание бэкапов

Предполагается что у вас всё готово для миграции:

1. Есть развёрнутый и рабочий NOC
2. Есть дампы баз PostgreSQL и MongoDB (см. пункт Получение бэкапов выше)

3. Необходимо остановить NOC командой service noc stop

Развёртывание бэкапа PostgreSQL

1. Подключаемся к PostgreSQL с базой нового NOC'а под пользователем postgres (pgsql для FreeBSD).

Linux

```
# su - postgres
# psql -d template1
```

2. Пересоздаём базу нового НОКа. Вместо <pg_db> необходимо подставить имя базы, которое задавали при настройки Environments в Башне (по умолчанию noc).

```
postgres=# DROP DATABASE <pg_db>;
postgres=# CREATE DATABASE <pg_db> ENCODING 'UTF8' OWNER <pg_user>;
postgres=# \q
```

3. Заливаем бэкап от старого НОК'а в созданную БД. Делать это необходимо из-под пользователя postgres (pgsql для FreeBSD).

Будьте осторожны если старый NOC установлен на версии PostgreSQL ниже 9 (н-р 8.4). По умолчанию новый NOC устанавливает версию PostgreSQL 9.4. Прямое развёртывание дампов версий ниже 9 может приводить к ошибкам. Также, в новом НОКе не используется плагин PostGIS, поэтому при импорте из бэкапа данных ГИС будут выдаваться ошибки.

Можно провести миграцию всех данных Postgres в версию 9.4. Для этого можно воспользоваться официальным руководством Миграция данных между версиями PostgreSQL.

Процедура миграции PostgreSQL

```
####      PostgreSQL
# su - postgres
# cd /tmp/
# pg_dumpall > pg_backup.dump
# xit
####  pg_backup.dump     PostgreSQL (- scp /tmp/pg_backup.dump root@<new_nodeIP>:/tmp/)
####
# su - postgres
# cd /tmp/
# psql -d template1
postgres=# DROP DATABASE <pg_db>;
postgres=# \q
# psql -f /tmp/pg_backup.dump
```

```
    ,    NOC     PostgreSQL ,        .       Environments,       PostgreSQL :
ALTER USER <pg_user> WITH PASSWORD <pg_user_password>;
```

```
# su - postgres
# pg_restore -d noc <dump_file_name>
```

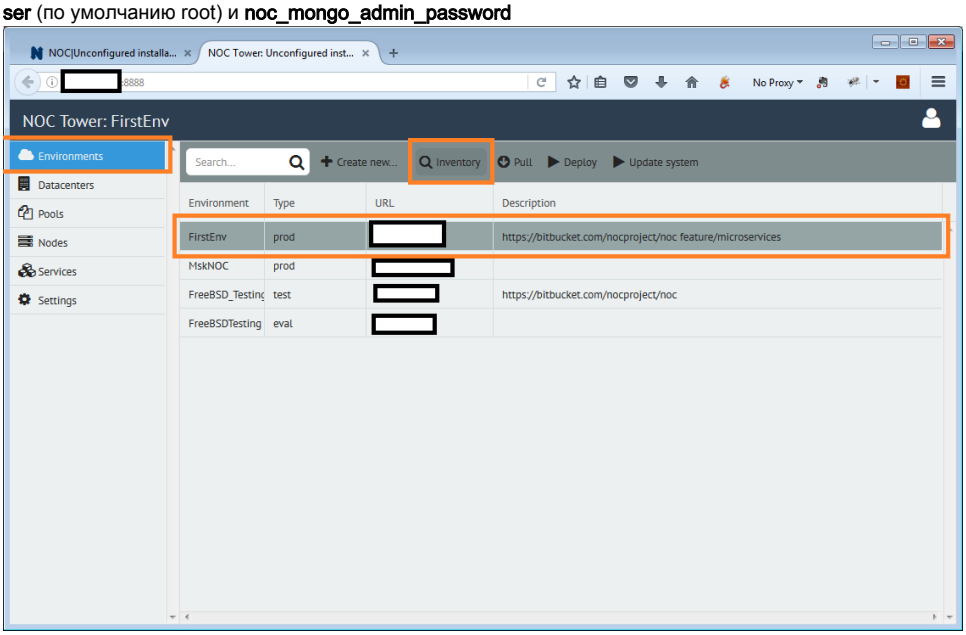
4. Убеждаемся что во время импорта критических ошибок не появлялось (при критичных ошибках восстановление прервётся).

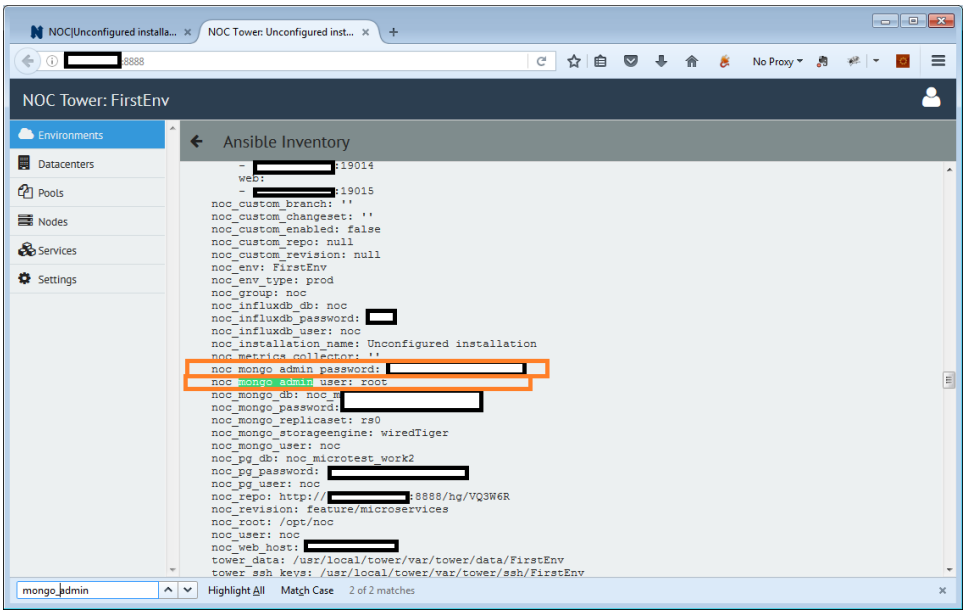
Развёртывание бэкапа MongoDB

Сокращения

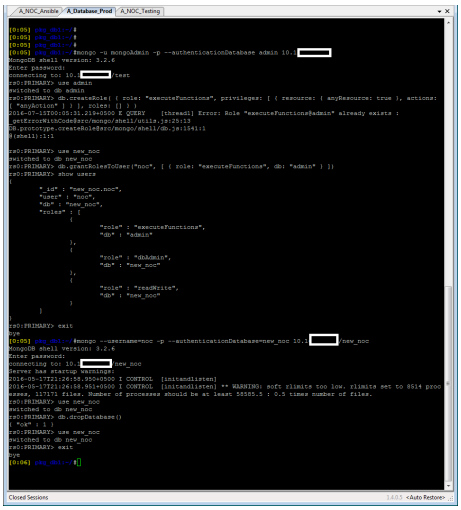
- <noc_mongo_admin_user> - MongoDB . , userAdminAnyDatabase (root)
- <noc_mongo_admin_password> - <noc_mongo_admin_user>
- <noc_mongo_database> - , Environments
- <noc_mongo_user> - , , Environments

1. Подключаемся к MongoDB из под пользователя с административными правами. В случае, если монго разворачивалась из Башни, узнать пользователя можно выбрав созданный Environment и нажав кнопку Inventory сверху. Там необходимо найти пункты **noc_mongo_admin_user** (по умолчанию root) и **noc_mongo_admin_password**






```
# mongo -u <noc_mongo_admin_user> -p --authenticationDatabase admin 127.0.0.1
```



2. Создаём роль "executeFunctions" и выдаём её пользователю <noc_mongo_user>

```
rs0:PRIMARY> use admin
rs0:PRIMARY> db.createRole( { role: "executeFunctions", privileges: [ { resource: { anyResource: true },
actions: [ "anyAction" ] } ], roles: [] } )
rs0:PRIMARY> use <noc_mongo_database>
rs0:PRIMARY> db.grantRolesToUser("noc", [ { role: "executeFunctions", db: "admin" } ])
rs0:PRIMARY> exit
```

3. Пересоздаём базу

> При выполнении команды db.dropDatabase(), начиная с версии MongoDB 2.6, пользователь не удаляется. Если в этом есть необходимость, можно воспользоваться командой db.runCommand({ dropAllUsersFromDatabase: 1 }).
>
> Для создания пользователя применяется команда: db.createUser({user:*<noc_mongo_user>*, pwd:*<noc_mongo_user_password>*, roles: [{ role: "dbAdmin", db:*<noc_mongo_database>*},{ role: "readWrite", db:*<noc_mongo_database>*},{ role: "executeFunctions", db: "admin" }]})

```
# mongo -u <noc_mongo_user> -p --authenticationDatabase <noc_mongo_database> 127.0.0.1
rs0:PRIMARY> use <noc_mongo_database>
rs0:PRIMARY> db.dropDatabase()
rs0:PRIMARY> use <noc_mongo_database>
rs0:PRIMARY> exit
```

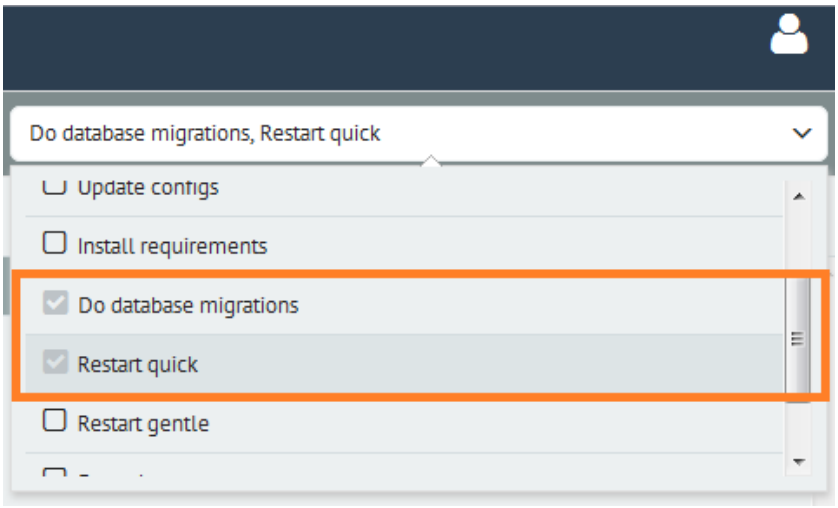
4. Распаковываем бэкап и импортируем данные старого НОКа в пересозданную базу.

```
# tar -xzf noc-mongo-2016-05-05-10-47.tar.gz
# mongorestore --host=localhost --username=<noc_mongo_user> --
authenticationDatabase=<noc_mongo_database> --db=<noc_mongo_database> ./<old_mongo_db_name>
```

Миграция.

> Если, при восстановление бэкапов, вы изменили имя базы или пользователя, или заменили пароль то необходимо актуализировать настройки в Environments.

После выполнения всех действий, отмечаем в башне опции "Do database migration" и "Restart quick". Затем нажимаем Deploy, в процессе произойдёт миграция данных.



Если возникли какие-то проблемы с миграцией через башню. Можно выполнить миграцию вручную. Команды для этого (выполняются из папки НОКа):

1. ./noc migrate
2. ./noc ensure-indexes
3. ./noc collection sync
4. ./noc sync-perm
5. ./noc sync-mibs
6. Если миграция выполняется в первый раз - необходимо выполнить: ./scripts/deploy/apply-pools

Проверка

Если у вас после миграции не запускается ВЕБ - проверьте что НОК запущен!

После успешной миграции необходимо зайти в НОК и проверить что все данные переехали нормально. Если возникли проблемы, необходимо поискать ответы в пункте Возможные проблемы и затем спросить в общем чате, возможно, кто-то уже с ними сталкивался🙂

Возможные проблемы

В: После миграции не открывается Веб.

О: Проверьте что НОК запущен.

Проверка состояние процессов.

```
# cd /opt/noc
# ./noc ctl
###     RUNNING
```

В: После миграции не отображаются МО.

О: Обычно, такое может произойти при ошибках миграции - например, если не была очищена база Монги, перед заливкой туда бэкапа. Необходимо повторить шаги миграции.

В: Всем устройствам назначем странный пул P0001

О: Это пул "затычка" создался при миграции. Необходимо вручную пройтись по устройствам и назначить рабочий пул или воспользоваться noc shell (вместо <pool_name> необходимо указать имя своего пула (по умолчанию - default)):

```
# ./noc shell
from noc.sa.models.managedobject import ManagedObject
from noc.main.models.pool import Pool
p = Pool.objects.get(name=<pool_name>)
for m in ManagedObject.objects.filter(is_managed=True):
    m.pool = p
    m.save()
```

В: После миграции с устройств не собираются данные.

О: Необходимо проверить Managed Object Profiles (Service Activation -> Setup -> Managed Object Profiles). По умолчанию все методы сбора отключены. Также необходимо просмотреть логи discovery и активаторов на предмет ошибок и трейсов.