Установка NOC в RHEL6

Думаю, в CentOS6.x тоже прокатит, плюс-минус какие-то нюансы

Кратенько, в режиме HOWTO

1. Ставим репо (mongod, postgre, epel) и необходимые пакеты:

```
#yum localinstall http://yum.postgresql.org/9.4/redhat/rhel-6-x86_64/pgdg-redhat94-9.4-2.noarch.rpm
#yum localinstall http://mirror.yandex.ru/epel/6Server/x86_64/epel-release-6-8.noarch.rpm
```

   Репо для mongo прописываем руками

```
[root@srv-noc-dev etc]# cat /etc/yum.repos.d/mongod.repo
[mongodb-org-2.6]
name=MongoDB 2.6 Repository
baseurl=http://downloads-distro.mongodb.org/repo/redhat/os/x86_64/
gpgcheck=0
enabled=1
```

   Включаем actual repo (у меня репозиторий корпоративный, но суть в том, что желательно включить наиболее актуальные репозитории)

```
#yum update
#yum install mongodb-org-server.x86_64 mongodb-org-mongos.x86_64 mongodb-org-shell.x86_64 postgresql94.
x86_64 postgresql94-contrib.x86_64 postgresql94-python.x86_64 postgresql94-server.x86_64 libsmi.x86_64
httpd python-devel.x86_64 quilt.x86_64 gmpy.x86_64 python-psycopg2.x86_64 libpqxx-devel.x86_64 gmp-devel.
x86_64 python-pip.noarch
```

2. Ставим hg, virtualenv, скачиваем репо NOC и добавляем группу и пользователя. Python у нас будет 2.6. Это не рекомендованный 2.7, конечно, но уж что было.

```
#pip install virtualenv
#pip install mercurial
#cd /opt && hg clone http://bitbucket.org/nocproject/noc noc
#virtualenv  /opt/noc
#groupadd noc
#useradd -g noc -s /bin/sh -d /home/noc noc
```

3. Настраиваем демон PostgreSQL

```
#service postgresql-9.4 initdb
#chkconfig postgresql-9.4 on && service postgresql-9.4 start
```

   Настраиваем правила доступа к базе

```
[root@localhost etc]# cat /var/lib/pgsql/9.4/data/pg_hba.conf | grep -v ^# | grep -v ^$
local   all             all                                     peer
host    noc     noc     127.0.0.1/32    trust
host    noc     noc     ::1/128 trust
host    all             all             127.0.0.1/32            trust
host    all             all             ::1/128                 trust
local   replication     postgres                                peer
host    replication     postgres        127.0.0.1/32            ident
host    replication     postgres        ::1/128                 ident
```

   Создаем роль и базу

```
su - postgres
postgres@/$ psqlpostgres=# CREATE USER noc SUPERUSER ENCRYPTED PASSWORD 'thenocproject';
postgres=# CREATE DATABASE noc ENCODING 'UTF8' OWNER noc;
postgres=# \q
```

4. Настраиваем Mongo (версия монги должна быть не ниже 2.6)

```
#chkconfig mongod on && service mongod start

#mongo
MogoDB shell version: 2.0.1connecting to: test
> use noc
switched to db noc
> db.createUser({ "user" : "noc", "pwd" : "noc", "roles" : [ { "role" : "dbOwner", "db" : "noc" }] }
```

5. Копируем upgrade.conf, заодно прописываем в PATH путь к бинарикам psql, т.к. путь отличается от дефолта

```
#cp /opt/noc/etc/upgrade.defaults /opt/noc/etc/upgrade.conf
#PATH=/usr/pgsql-9.4/bin:${PATH}
```

6. Конфиги (с сокращениями).

```
[root@localhost etc]# cat /opt/noc/etc/noc.conf
...
[database]
engine = postgresql_psycopg2
name = noc
user = noc
password = thenocproject
host = 127.0.0.1
port = 5432

[nosql_database]
name = noc
user = noc
password = noc
host = 127.0.0.1:27017
port =
replica_set =
```

```
[root@localhost etc]# cat /opt/noc/etc/upgrade.conf
...
# ENABLED:
# Enable upgrade system.
# Read and configure following file
# and change to ENABLED=yes
#
ENABLED=yes

...
# BRANCH:
# Branch to follow
# Possible values are:
# default - production releases
# develop - stable development branch
# release/<version> - pre-release testing branch
# feature/<name> - feature development branch
# hotfix/<version> - hotfix development branch
# none - do not pull and apply repo updates
#
BRANCH=develop
```

```
[root@localhost etc]# cat noc-launcher.conf
...
[noc-activator]
enabled = true
user = root
group =
config.0 = etc/noc-activator.conf
```

7. Выставляем права на папку с логами и папки самого noc

```
#cd /var/noc && chmod a+wr -R .
#cd /opt/noc && chown noc -R .
#cp -r /opt/noc/django/contrib/admin /opt/noc/lib/python2.6/site-packages/django/contrib/admin
```

настраиваем virtual host для apache (можно использовать и nginx, разумеется)

```
[root@localhost etc]# cat /etc/httpd/conf.d/noc.conf
LoadModule authz_host_module modules/mod_authz_host.so
LoadModule alias_module moduXles/mod_alias.so
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so

<VirtualHost *:80>
    DocumentRoot /opt/noc
    ServerName srv-noc
    Alias /media /opt/noc/lib/python2.6/site-packages/django/contrib/admin/static/
    Alias /static /opt/noc/static
    Alias /log /var/noc/log
    <Directory /opt/noc>
        Order allow,deny
        Allow from all
    </Directory>
    <Location /media>
        Order allow,deny
        Allow from all
    </Location>
    ProxyRequests On
    <Proxy *>
        Order deny,allow
        Allow from all
    </Proxy>
    ProxyPass / http://localhost:8000/
    ProxyPassReverse / http://localhost:8000/
</VirtualHost>
```

8. Стартуем noc, проверяем. Если активатор ругается на "Activator pool 'default' is not available", проверяем, что noc-activator стартует от рута.