Установка Mazar

Описание нацелено на установку на системе CentOS 6 x64. В случае работы с другой OS необходима доводка на месте напильником. При желании ничто не мешает установить админку на любой системе, где поддерживаются все системные требования.

Системные требования (кроме OS)

Python 2.7 – текущая минорная версия. На Python 2.6 работать не будет (ограничения используемых библиотек). Как установить и использовать версию 2.7 без головной боли и ручной сборки на CentOS (где текущая версия 2.6) – описано ниже.

PostgreSQL версии 9.2 или более поздней. В теории возможна работа на 8.4, но лучше не рисковать производительностью. Также рекомендуется использование пула коннекций pgbouncer (включено в описание установки).

Redis 2.8 для кэширования данных в максимально нагруженных участках кода.

MongoDB 3.0 или выше – используется как хранилище неструктуризованных данных.

UWSGI как сервер приложений. Проверено с версией 2.0.10 LTS – которая и рекомендуется для использования.

Nginx как www-сервер. Практически любая версия будет работать, ибо ничего неземного от него не ожидается.

Также очень желателен logrotate – поскольку приложение ведет лог, который может вырасти до полного заполнения диска. Так что, если вы не используете logrotate на системе – озаботьтесь самостоятельно тем, чтобы /var/log/uwsgi/uwsgi_log не заполнил все свободное пространство.

Конфигурация системы и установка системного ПО

Провести апдейт системы и установить требуемые репозитории – epel, postgresql, RedHat software collections, remi. Для начала стоит проверить – некоторые хостеры могут иметь какие-то из этих репозиториев предустановленными.

```
$ yum update
```
$ yum install http://dl.fedoraproject.org/pub/epel/6/i386/epel-release-6-8.noarch.rpm
$ yum install centos-release-SCL
$ yum install http://yum.postgresql.org/9.4/redhat/rhel-6-x86_64/pgdg-centos94-9.4-1.noarch.rpm
$ yum install http://rpms.famillecollet.com/enterprise/remi-release-6.rpm

Remi по умолчанию отключен, так что отредактируйте /etc/yum.repos.d/remi.repo, заменив enabled=0 на enabled=1 в секции [remi]

Установите nginx

```
$ yum install nginx
```

PostgreSQL & pgbouncer

```
$ yum install postgresql94.x86_64 postgresql94-server.x86_64
```
$ yum install pgbouncer.x86_64

Redis

```
$ yum install redis.x86_64
```

Также нам понадобится то, что необходимо для сборки некоторых компонентов вручную и сервисных файлов

```
$ yum groupinstall "Development tools"
$ yum install libcurl-devel pcre-devel hatools
```

И – Python2.7 из Software Collection Library с модулями

```
$ yum install python27.x86_64 python27-python-devel.x86_64 python27-python-
psycopg2.x86_64 python27-python-pymongo.x86_64 python27-python-
virtualenv.noarch
```

Теперь остается создать пользователя, который будет использоваться для установки приложения. По умолчанию подразумевается, что имя пользователя – admin. Путь к его home-директорию указан в некоторых файлах конфигурации, так что если вы используете другое имя – вам надо будет править и конфигурационные файлы приложения.

Также необходимо дать возможность процессам, которые работают под другим uid, читать файлы внутри его директория (давать право на получение списка файлов вовсе не обязательно).

Для этого пользователя мы установим пароль, чтобы можно было пользоваться этим пользователем для разворачивания и конфигурации приложения.

```
$ useradd -m -s /bin/bash admin
```
$ chmod go+x /home/admin
$ passwd admin

Установка системных файлов конфигурации

Загрузите на сервер установочные файлы – etc.tgz и app.tgz. В первом архиве находятся файлы,

которые необходимо развернуть в /etc

```
$ tar xzvf etc.tgz -C /etc
```

Это перепишет конфигурационный файл и файл с пользователями для pgbouncer – больше ничего на чистой системе переписано не будет. Вы можете поменять данные для соединения с базой данных PostgreSQL, которую использует приложение – в этом случае вам надо будет поменять их также и в /etc/pgbouncer/users.txt

Кроме прочего, это добавит новый репозиторий – для MongoDB. Установите его сразу

```
$ yum install mongodb-org-server.x86_64 mongodb-org-shell.x86_64 mongodb-
org-tools.x86_64
```

Отредактируем конфигурацию nginx – так, чтобы выключить в ней сайты по умолчанию и добавить нашу конфигурацию. По умолчанию nginx ищет файлы сайтов в /etc/nginx/conf.d – мы изменим это на /etc/nginx/domains.d/

```
$ vim /etc/nginx/nginx.conf
```

меняем include /etc/nginx/conf.d/*.conf; на include /etc/nginx/domains.d/*.conf;

Теперь отредактируйте /etc/nginx/domains.d/slempo.conf – там необходимо сменить <YOUR ADDRESS> на адрес вашего сервера.

Создадим директории для статических файлов сервера, которые будут использоваться приложением, а также – для логов

```
$ mkdir -p /var/www/vhosts/smsg/content/static
$ mkdir -p /var/www/vhosts/smsg/content/media
$ chown -R admin:admin /var/www/vhosts
$ chmod 777 /var/www/vhosts/smsg/content/media
$ mkdir /var/log/uwsgi
$ chown nginx /var/log/uwsgi
```

Конфигурация PostgreSQL

```
$ service postgresql-9.4 initdb
$ vim /var/lib/pgsql/9.4/data/pg_hba.conf
```

...и меняем ident авторизацию на md5 для не-сокетных (то есть содержащих IP-адреса) соединений. Сохраняем файл, выходим из редактора.

```
$ service postgresql-9.4 start
$ su - postgres
$ createuser -l -P traff
$ createdb -O traff slempo
$ exit
```

Команда “createuser -l -P traff” запрашивает пароль – введите 12345 :) Это, мягко говоря, не

самый безопасный пароль, и такой хорош только для чемодана – но это пароль на базу, который может использовать только локальный пользователь. Если у вас на локальной машине нарушитель – то безопасность базы, наверное, уже не имеет никакого смысла. Этот же пароль указан еще в двух местах, в конфигурации приложения и pgbouncer – если хотите выбрать другой, то меняйте его и там.

Также – добавляем содержимое /etc/sysctl.conf.add в /etc/sysctl.conf

```
$ cat /etc/sysctl.conf.add >> /etc/sysctl.conf
$ sysctl -p
```

Теперь стартуем необходимые сервисы, которые уже можно стартовать:

```
$ service nginx restart
$ service mongod start
$ service redis start
$ service pgbouncer start
```

Установка сервера приложений

Единственный надежный способ установки UWSGI – это установка из исходников. Проводить ее мы будем под пользователем admin, которого создали раньше. Зайдем на сервер созданным пользователем. Первым делом – включим Python 2.7 в шелле, который используем для дальнейшей сборки

```
$ scl enable python27 bash
```

Качаем отсюда

http://projects.unbit.it/downloads/uwsgi-2.0.10.tar.gz

прямо на сервер

```
$ wget http://projects.unbit.it/downloads/uwsgi-2.0.10.tar.gz
```

разворачиваем и собираем

```
$ tar xzvf uwsgi-2.0.10.tar.gz
$ cd uwsgi-2.0.10
$ make
```

Копируем получившийся файл uwsgi в /usr/local/sbin/ - там его по умолчанию ищет инициализационный файл /etc/init/uwsgi.conf

Установка приложения

Разворачиваем app.tgz прямо в хомяк admin'a (подразумевается, что app.tgz лежит там, и мы

находимся там же в юзерском шелле)

```
$ tar xzvf app.tgz
```

Начинаем править конфигурацию

```
$ vim apps/smsg_r/smsg/local_settings.py
```

И там правьте все, что написано большими буквами в угловых скобках. В частности:

<YOUR SERVER ADDRESS> - меняется на тот адрес сервера, по которому вы будете к нему обращаться.

<YOUR SECRET KEY> - это уникальный ключ, который используется для криптографических целей. Его вы можете создать самостоятельно, к примеру – на онлайновом генераторе тут: http://www.miniwebtool.com/django-secret-key-generator/

Далее, очень важный момент. По умолчанию в приложении включен отладочный режим. Это помогает увидеть, что не так во время установки – но должно быть отключено как только вы покончили с установкой и все работает. Для этого строку
DEBUG = True
надо заменить на
DEBUG = False

Еще раз – это ВАЖНО. Без этого вся конфигурация вашего сервера будет вылезать наружу при любой ошибке. С отключенным отладочным режимом вы будете получать уведомления об ошибках на почтовый адрес, указанный в local_settings.py, а пользователи – видеть стандартную страницу с Error 500

Создаем virtualenv и конфигурируем базу

Под тем же пользователем

```
$ cd
$ mkdir virtualenv
$ cd virtualenv
$ virtualenv --system-site-packages django
$ source ./django/bin/activate
$ cd ~/apps/smsg_r
$ pip install -r requirements.txt
```

После последней команды начинается сборка всех необходимых модулей. Если она закончилась проблемой – решаем, задаем вопросы. В итоге – таки решаем.

Теперь покончим с базой:

```
$ ./manage.py migrate
$ ./manage.py collectstatic --noinput
```

```
$ ./manage.py createsuperuser
```

Последняя команда запросит у нас имя пользователя, email и пароль. Email можно вводить от фонаря, имя – любое на выбор, пароль – чем безопаснее, тем лучше. Уж точно не 12345 – ибо этот вход *будет* виден всем.

Ключ на старт

Добавляем crontab для нашего пользователя. Изначально лежит в ~/apps/crontab

```
$ crontab ~/apps/crontab
```

Снова – рутовый шелл.

Стартуем сервер приложений

```
$ initctl start uwsgi
```

Теперь убедимся в том, что на следующем рестарте сервера все будет работать, как надо

```
$ chkconfig nginx on
$ chkconfig mongod on
$ chkconfig pgbouncer on
$ chkconfig postgresql-9.4 on
$ chkconfig redis on
```

Все, можно заходить по указанному адресу в админку. Первым делом – создайте хотя бы одного инсталлера в разделе Installers с User id 1. Пароль можете выбрать случайный и забыть.

Где находится адрес и порт бота?

Они – в /etc/nginx/domains.d/slempo.conf
Тот коротенький прокси-конфиг, который почти в конце его. По умолчанию – порт 2080. Меняйте там на свое усмотрение.

Не открывается админка или боты не могут достучаться до порта 2080

Скорее всего – поднят файрволл. Проверить можно командой:

```
# iptables -L -n
```

Если список правил в трех policy не пуст – ваши порты прикрыты по умолчанию. Откройте доступ к портам 80 и 2080:

```
# lokkit --service=http --update
# lokkit --port=2080:tcp –update
```

Как увеличить производительность системы?

1. Отключите журнал запросов от ботов, добавив в smsg/local_settings.py строку:

```
LOG_CLIENT_REQUESTS = True
```

2. Отключите fsync в PostgreSQL, изменив в файле

/var/lib/pgsql/9.4/data/postgresql.conf

следующее:

Уберите комментарий со строки

fsync = on

Измените значение на off