# БАРС.Муниципалитет – Электронная Похозяйственная Книга

**Инструкция по установке**

# Развертывание приложения с нуля на ОС Linux

# 1. Установка и настройка сервера баз данных

**Postgresql - 9.2**

**Для Ubuntu**

```
sudo apt-get install python-software-properties
sudo add-apt-repository ppa:pitti/postgresql
sudo apt-get update
sudo apt-get install postgresql-9.2 postgresql-server-dev-9.2
```

**Для CentOS**

```
sudo rpm -i http://yum.postgresql.org/9.2/redhat/rhel-6-x86_64/pgdg-centos92-9.2-
6.noarch.rpm
sudo yum install postgresql92 postgresql92-server postgresql92-libs postgresql92-
contrib
sudo service postgresql-9.2 initdb
sudo chkconfig postgresql-9.2 on
sudo /etc/init.d/postgresql-9.2 start
```

Далее необходимо создать пользователя и базы данных с которыми он будет работать.

```
sudo -u postgres psql
CREATE USER user PASSWORD 'password';
CREATE DATABASE project_db OWNER user;
CREATE DATABASE project_db_service OWNER user;
```

После этого рекомендуется сменить пароль пользователю postgres

В случае, если приложение и база данных находятся на одном сервере, то можно переходить к 2 пункту данной инструкции.
В случае, если приложение и база данных разнесены на несколько серверов, то необходимо провести дополнительную настройку.

- в файле postgresql.conf необходимо раскомментировать listen_addresses и указать
- `listen_addresses = '*'`
- `port = 5432`
- в файле pg_hba.conf
  добавить адреса с которых будет разрешено подключаться к базе данных. Например так
- `# TYPE  DATABASE        USER            ADDRESS                 METHOD`
- `host    all             all             192.168.0.0/24            md5`

Расположение файлов по умолчанию для Ubuntu
/etc/postgresql/9.2/main/

Расположение файлов по умолчанию для CentOS
/var/lib/pgsql/9.2/data/

**После внесенных изменений обязательно перезапустить Postgresql**

# 2. Настройка сервера приложения

## 2.0. Подготовительный этап

Приложения работают с использованием Python версией 2.6. и 2.7. Далее будет рассмотрена установка проекта под версию Python 2.7

Необходимо убедиться, что в систему установлены следующие пакеты. Рекомендуется запустить их установку, в случае если их не будет, они установятся, а если есть, то либо установится более новая версия, либо ничего не изменится.

**Для Ubuntu**

```
sudo apt-get update
sudo apt-get install python2.7 python2.7-dev libpq-dev libxmlsec1-dev openjdk-6-jre
openjdk-6-jre-lib libjpeg8 libjpeg8-dev libfreetype6-dev zlib1g-dev python-setuptools
sudo apt-get install default-jdk
sudo easy_install pip
```

После того, как установка завершится необходимо выполнить небольшую настройку, для того, чтобы впоследствии пакет PIL установился и приложения смогли нормально работать с изображениями
Необходимо выполнить с правами пользователя root

```
# ln -s /usr/lib/`uname -i`-linux-gnu/libfreetype.so /usr/lib/
# ln -s /usr/lib/`uname -i`-linux-gnu/libjpeg.so /usr/lib/
# ln -s /usr/lib/`uname -i`-linux-gnu/libz.so /usr/lib/
```

**Для CentOS**

```
sudo yum install postgresql-devel xmlsec1-openssl xmlsec1-openssl-devel java-1.6.0-
openjdk-devel gcc zlib-devel libjpeg-devel freetype-devel libxslt-devel libpng libpng-
devel libtool-ltdl-devel libmcrypt libmcrypt-devel python-devel wget bzip2-devel
```

Скачиваем и устанавливаем python 2.7

```
wget --no-check-certificate https://www.python.org/ftp/python/2.7.6/Python-2.7.6.tar.xz
tar xf Python-2.7.6.tar.xz
cd Python-2.7.6
./configure --prefix=/usr/local
make && sudo make altinstall
sudo ln -s /usr/local/bin/python2.7 /usr/bin/
```

Устанавливаем setuptools и pip

```
wget https://bitbucket.org/pypa/setuptools/raw/bootstrap/ez_setup.py
sudo /usr/local/bin/python2.7 ez_setup.py
sudo /usr/local/bin/easy_install-2.7 pip
sudo ln -s /usr/local/bin/pip /usr/bin/
```

# 2.1. Настройка окружения

Предлагается использовать следующую структуру папок

```
/opt
 └── bars
     └── epk
         ├── var
         |   ├── log
         |   └── run
         ├── venv
         ├── backup
         ├── tmp
         └── epk
```

Временно устанавливаем права на папку /opt/bars для текущего пользователя.

sudo chown -R $USER:$USER /opt/bars/

Копируем полученные файлы(дитсрибутив, файл зависимостей и ключ) в папку /opt/bars/epk/tmp

Приложения работают с использованием виртуального окружения, для его настройки необходимо в систему установить следующие пакеты:

```
sudo pip install virtualenv virtualenvwrapper
```

По умолчанию в Ubuntu и CentOS командная оболочка установлена bash. И для неё в файле ~/.bashrc необходимо установить следующие параметры

```
export WORKON_HOME=/opt/bars/epk/venv
source /usr/local/bin/virtualenvwrapper.sh
```

При следующем входе в систему эти параметры будут установлены автоматически. На данном этапе инструкции их необходимо выполнить вручную один раз.
Сейчас стали доступны команды для работы с виртуальным окружением.

Создаем виртуальное окружение

```
mkvirtualenv epk --python=/usr/bin/python2.7 --no-site-packages
workon epk
```

Активировав виртуальное окружение необходимо выполнить команду по установке зависимостей проекта.

```
workon epk
pip install -r /opt/bars/epk/tmp/REQUREMENTS -i http://pypi.bars-open.ru/simple/
```

После установки можно выполнить команду (в виртуальном окружении) pip freeze, которая выведет примерно такой список установленный библиотек

```
Django==1.3.1

PIL==1.1.7

SQLAlchemy==0.7.1
```

South==0.7.6

amqp==1.0.13

anyjson==0.3.3

billiard==2.7.3.34

celery==3.0.24

datalogging==1.2.3

dbfpy==2.2.5

django-celery==3.0.21

django-json-field==0.5.5

django-mptt==0.5.5

kombu==2.5.16

lxml==3.1beta1

m3==1.0.6.10

m3-audit==1.1

m3-autologin==1.0

m3-contragents==1.1.0.1

m3-dicts==1.0

m3-kladr==1.1.1

m3-mutex==1.0

m3-replica==1.0

m3-ssacc-client==1.0

m3-users==1.1.2

ordereddict==1.1

psycopg2==2.4.1

pyasn1==0.1.7

python-dateutil==2.1

pytz==2012j

requests==2.3.0

rsa==3.1.1-bars

six==1.2.0

wsgiref==0.1.2

xlrd==0.8.0

xlwt==0.7.4

# 2.2. Установка приложения.

Распаковываем и помещаем проект в назначенное местоТак же в папке conf должен быть лицензионный ключ (signed.xml), который выдается менеджерами.

```
workon epk
cd /opt/bars/epk/tmp
```
unzip epk.egg
mv `/opt/bars/epk/tmp /opt/bars/epk`
cp signed.xml `/opt/bars/epk/`epk
cd `/opt/bars/epk/`epk

В конфигурационном файле (/opt/bars/epk/epk/epk.conf) необходимо проверить правильность данных для связи приложения с базой данных.
В двух секциях [database]

Далее необходимо запустить команды, которые создадут необходимые таблицы в базе данных

```
python manage.pyc collectstatic // yes
python manage.pyc syncdb //no
python manage.pyc migrate
```

# 2.3. Настройка supervisor + gunicorn

Устанавливаем gunicorn 17.5

```
workon project
pip install gunicorn==17.5
```

В папке /opt/bars/epk/ необходимо создать файл gunicorn.conf.py со следующим содержанием

```
#! coding: utf-8

import os

import multiprocessing



current_dir = os.path.abspath(os.path.dirname(__file__))



bind      = "unix:/opt/bars/epk/var/run/epk.sock"

workers   = multiprocessing.cpu_count() * 2



user      = "www-data"

group     = "www-data"

errorlog  = "/opt/bars/epk/var/log/gunicorn.log"

loglevel  = "info"

pidfile   = "/opt/bars/epk/var/run/epk.pid"

timeout   = 3000
```

```
Создайте, если нужно, пользователя www-data.

sudo useradd www-data -M
```

**Установка и настройка supervisor**

Команды выполняются с правами root

```
pip install supervisor
```

**Для CentOS**

```
echo_supervisord_conf > /etc/supervisord.conf
ln -s /usr/local/bin/supervisor* /usr/bin/
```

**Для ubuntu**

```
/usr/local/bin/echo_supervisord_conf > /etc/supervisord.conf
```

в /etc/ появится файл конфигурации supervisord.conf, открываем и правим пару строк

```
chmod=0700                 ; socket file mode (default 0700)
chown=www-data:www-data       ; socket file uid:gid owner

; В самый низ добавляем

[include]
files = /etc/supervisor/conf.d/*.conf
```

Создадим папку supervisor и в ней conf.d, в ней будут храниться файлы конфигураций для запуска приложений

```
sudo mkdir -p /etc/supervisor/conf.d
```

В папке /etc/supervisor/conf.d/ создадим файлик project.conf(название приложения)

```
sudo touch /etc/supervisor/conf.d/epk.conf
```

открываем и пишем в него:

```
[program:epk]



command=/opt/bars/epk/venv/epk/bin/python manage.pyc run_gunicorn -c
/opt/bars/epk/gunicorn.conf.py



directory=/opt/bars/epk/epk

#пользователь и группа от чьего имени будет запускаться приложение

user=www-data

group=www-data



daemon=False
```

```
debug=False



autostart=true

autorestart=true



redirect_stderr=True

stdout_logfile=/opt/bars/epk/var/log/main.log
```

Перед запуском Supervisor'a необходимо убедиться, что для пользователя www-data есть необходимые права

```
sudo chown -R www-data:www-data /opt/bars/epk
sudo supervisord -c /etc/supervisord.conf
```

Проверить, что supervisor работает можно командой

```
sudo supervisorctl status
#которая выдаст примерно такое сообщение
epk                              RUNNING    pid 14894, uptime 0:04:21
```

## Автозапуск supervisor

### Пример для Ubuntu

Для того, чтобы supervisor запускался автоматически при запуске ОС необходимо добавить файл со следующим содержанием в /etc/init.d/supervisord

```
#! /bin/sh
### BEGIN INIT INFO
# Provides:          supervisord
# Required-Start:    $remote_fs
# Required-Stop:     $remote_fs
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Example initscript
# Description:       This file should be used to construct scripts to be
#                    placed in /etc/init.d.
### END INIT INFO

# Author: Dan MacKinlay <danielm@phm.gov.au>
# Based on instructions by Bertrand Mathieu
# http://zebert.blogspot.com/2009/05/installing-django-solr-varnish-and.html

# Do NOT "set -e"

# PATH should only include /usr/* if it runs after the mountnfs.sh script
PATH=/sbin:/usr/sbin:/bin:/usr/bin
DESC="Description of the service"
NAME=supervisord
DAEMON=/usr/local/bin/supervisord
DAEMON_ARGS=""
PIDFILE=/tmp/$NAME.pid
SCRIPTNAME=/etc/init.d/$NAME

# Exit if the package is not installed
[ -x "$DAEMON" ] || exit 0

# Read configuration variable file if it is present
```

```
[ -r /etc/default/$NAME ] && . /etc/default/$NAME

# Load the VERBOSE setting and other rcS variables
. /lib/init/vars.sh

# Define LSB log_* functions.
# Depend on lsb-base (>= 3.0-6) to ensure that this file is present.
. /lib/lsb/init-functions

#
# Function that starts the daemon/service
#
do_start()
{
    # Return
    #   0 if daemon has been started
    #   1 if daemon was already running
    #   2 if daemon could not be started
    start-stop-daemon --start --quiet --pidfile $PIDFILE --exec $DAEMON --test >
/dev/null \
        || return 1
    start-stop-daemon --start --quiet --pidfile $PIDFILE --exec $DAEMON -- \
        $DAEMON_ARGS \
        || return 2
    # Add code here, if necessary, that waits for the process to be ready
    # to handle requests from services started subsequently which depend
    # on this one.  As a last resort, sleep for some time.
}

#
# Function that stops the daemon/service
#
do_stop()
{
    # Return
    #   0 if daemon has been stopped
    #   1 if daemon was already stopped
    #   2 if daemon could not be stopped
    #   other if a failure occurred
    start-stop-daemon --stop --quiet --retry=TERM/30/KILL/5 --pidfile $PIDFILE --name
$NAME
    RETVAL="$?"
    [ "$RETVAL" = 2 ] && return 2
    # Wait for children to finish too if this is a daemon that forks
    # and if the daemon is only ever run from this initscript.
    # If the above conditions are not satisfied then add some other code
    # that waits for the process to drop all resources that could be
    # needed by services started subsequently.  A last resort is to
    # sleep for some time.
    start-stop-daemon --stop --quiet --oknodo --retry=0/30/KILL/5 --exec $DAEMON
    [ "$?" = 2 ] && return 2
    # Many daemons don't delete their pidfiles when they exit.
    rm -f $PIDFILE
    return "$RETVAL"
}

#
# Function that sends a SIGHUP to the daemon/service
#
do_reload() {
    #
    # If the daemon can reload its configuration without
    # restarting (for example, when it is sent a SIGHUP),
    # then implement that here.
    #
    start-stop-daemon --stop --signal 1 --quiet --pidfile $PIDFILE --name $NAME
    return 0
```

```
}

case "$1" in
  start)
    [ "$VERBOSE" != no ] && log_daemon_msg "Starting $DESC" "$NAME"
    do_start
    case "$?" in
        0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
        2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
    esac
    ;;
  stop)
    [ "$VERBOSE" != no ] && log_daemon_msg "Stopping $DESC" "$NAME"
    do_stop
    case "$?" in
        0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
        2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
    esac
    ;;
  #reload|force-reload)
    #
    # If do_reload() is not implemented then leave this commented out
    # and leave 'force-reload' as an alias for 'restart'.
    #
    #log_daemon_msg "Reloading $DESC" "$NAME"
    #do_reload
    #log_end_msg $?
    #;;
  restart|force-reload)
    #
    # If the "reload" option is implemented then remove the
    # 'force-reload' alias
    #
    log_daemon_msg "Restarting $DESC" "$NAME"
    do_stop
    case "$?" in
      0|1)
        do_start
        case "$?" in
            0) log_end_msg 0 ;;
            1) log_end_msg 1 ;; # Old process is still running
            *) log_end_msg 1 ;; # Failed to start
        esac
        ;;
      *)
          # Failed to stop
        log_end_msg 1
        ;;
    esac
    ;;
  *)
    #echo "Usage: $SCRIPTNAME {start|stop|restart|reload|force-reload}" >&2
    echo "Usage: $SCRIPTNAME {start|stop|restart|force-reload}" >&2
    exit 3
    ;;
esac

:
```

и выполнить следующие команды

```
sudo chmod +x /etc/init.d/supervisord
sudo update-rc.d supervisord defaults
```

### Пример для CentOS

```
#!/bin/sh
#
# /etc/rc.d/init.d/supervisord
#
# Supervisor is a client/server system that
# allows its users to monitor and control a
# number of processes on UNIX-like operating
# systems.
#
# chkconfig: - 64 36
# description: Supervisor Server
# processname: supervisord

# Source init functions
. /etc/rc.d/init.d/functions

prog="supervisord"

prefix="/usr/"
exec_prefix="${prefix}"
prog_bin="${exec_prefix}/bin/supervisord"
PIDFILE="/tmp/$prog.pid"
OPTS="-c /etc/supervisord.conf"

start()
{
        echo -n $"Starting $prog: "
        daemon $prog_bin --pidfile $PIDFILE $OPTS
        [ -f $PIDFILE ] && success $"$prog startup" || failure $"$prog startup"
        echo
}

stop()
{
        echo -n $"Shutting down $prog: "
        [ -f $PIDFILE ] && killproc $prog || success $"$prog shutdown"
        echo
}

case "$1" in

  start)
    start
  ;;

  stop)
    stop
  ;;

  status)
        status $prog
  ;;

  restart)
    stop
    start
  ;;

  *)
    echo "Usage: $0 {start|stop|restart|status}"
  ;;

esac
```

и выполнить следующие команды

```
sudo chkconfig --add supervisord
```

```
sudo chkconfig supervisord on
```

# 2.4. Установка и настройка Nginx

Чтобы установить самую последнюю версию nginx необходимо подключить официальный репозитарий и ставить уже из него.

## Установка

### Для Ubuntu

Для Debian/Ubuntu, для проверки подлинности подписи репозитория nginx, и чтобы избавиться от предупреждений об отсутствующем PGP-ключе во время установки пакета nginx, необходимо добавить ключ, которым были подписаны пакеты и репозиторий nginx, в связку ключей программы apt. Загрузите этот ключ с веб-сайта nginx и добавьте его в связку ключей программы apt, выполнив команду:

```
sudo wget http://nginx.org/keys/nginx_signing.key
sudo apt-key add nginx_signing.key
```

Для Ubuntu замените codename на кодовое имя дистрибутива, и добавьте в конец файла /etc/apt/sources.list следующее:

```
deb http://nginx.org/packages/ubuntu/ codename nginx
deb-src http://nginx.org/packages/ubuntu/ codename nginx
```

Для Ubuntu затем выполните команды:

```
apt-get update
apt-get install nginx
```

nginx автоматически добавится в автозагрузку

### Для CentOS

Создайте файл с именем /etc/yum.repos.d/nginx.repo и таким содержимым:

```
[nginx]
name=nginx repo
baseurl=http://nginx.org/packages/centos/6/$basearch/
gpgcheck=0
enabled=1
```

И последующая установка с добавлением nginx в автозагрузку.

```
sudo yum install nginx
sudo chkconfig nginx on
```

## Настройка

Необходимо создать файл /etc/nginx/proxy_params с содержимым

```
proxy_redirect off;
proxy_set_header Host $http_host;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
#proxy_set_header X-Forwarded-Proto https; # используется в случае с https
client_max_body_size 20m;
client_body_buffer_size 1024k;
proxy_buffering off;
```

```
proxy_connect_timeout 360;
proxy_send_timeout 360;
proxy_read_timeout 360;
proxy_buffer_size 4k;
proxy_buffers 32 32k;
proxy_busy_buffers_size 64k;
proxy_temp_file_write_size 1024k;
```

Что в папке /etc/nginx/conf.d и /etc/nginx/site-enabled есть уже примеры виртуальных хостов. Их лучше удалить или перенести в другие папки.
Создадим файл /etc/nginx/conf.d/epk.conf

```
upstream epk {

        server  unix:/opt/bars/epk/var/run/epk.sock fail_timeout=0;

}



server {

    listen      80;

    #server_name  project.com; #Если есть доменное имя, то прописывается здесь

    access_log  /opt/bars/epk/var/log/nginx_access.log;

    error_log   /opt/bars/epk/var/log/nginx_error.log;


    keepalive_timeout 5;


    location / {

    proxy_pass http://epk;

    include proxy_params;

    }


    location /static/ {

    alias /opt/bars/epk/epk/static/;

       expires 3d;

    }


    location /media/ {

    alias /epk/bars/epk/epk/media/;

       expires 3d;

    }
```

```
}
```

Перезапускаем nginx

```
sudo /etc/init.d/nginx restart
```