Домашнее задание 2

Настройка веб-серверов

Цель домашнего задания подготовить инфраструктуру для будущего проекта и научиться настраивать веб-серверы. Внимание! Вместо **pupkin** используйте **свою фамилию**.

1. Создание директории проекта ask_pupkin

С помощью утилиты django-admin (появляется после установки django) нужно создать следущую структуру:

```
ask_pupkin            - директория проекта
    |--- ask_pupkin   - библиотеки проекта (будут созданы django-admin.py)
    |--- manage.py    - скрипт управления (будет создан django-admin.py)
    |--- ask          - директория приложения (создается командой startproject)
    |--- templates    - шаблоны
    |--- static       - статические файлы (JS, CSS, картинки)
    |--- uploads      - файлы загруженные юзером
```

При этом директории templates, static, uploads — придется создать самостоятельно.

2. Настройка gunicorn для запуска WSGI скриптов

3. Создание простой WSGI скрипт

Создать скрипт, который:

- выводит “привет, мир”
- выводит список переданных GET и POST параметров
- выполняется при запросе localhost:8081

4. Реализовать пункт 3 с помощью django

Для этого необходимо сделать:

- создать новую вьюшку которая выводит GET и POST параметры в **views.py**
- подключить ее к нужному URL в **urls.py**
- прописать WSGI скрип, созданный Django (скорее всего wsgi.py) в gunicorn.

5. Настройка nginx для отдачи статического контента.

Необходимо настроить nginx следующим образом. Все файлы с URL начинающимся с **/uploads/** отдаются из **ask_pupkin/uploads**. Все файлы с расширением (.js .css .jpeg и т.д) — из директории **ask_pupkin/static**. Файлы должны отдаваться с заголовками, кеширующими файлы на стороне браузера. Файлы должны сжиматься на сервере для уменьшения размера передаваемых файлов. Размер конфига nginx не должен превышать 50 строк. Полученную конфигурацию необходимо запустить и проверить, для этого нужно разместить какой-либо файл (например **sample.html**) в директории **static** и загрузить его с помощью браузера **localhost/sample.html**.

6. Настройка проксирования в nginx

- Настроить nginx для проксирования всех нестатических запросов (URL без расширения, например **/** или **/login/**) на gunicorn.
- Настроить **upstream**.
- Настроить **proxy_cache** и проверить его работу.

7. Сравнение производительности.

С помощью утилиты Apache Benchmark (ab, идет в комплекте с Apache, для Ubuntu пакет **apache2-utils**) или wrk сравните производительность nginx (отдача статики) и gunicorn (запуск wsgi скриптов). Необходимо провести пять измерения:

- Отдача статического документа через nginx
- Отдача статического документа через gunicorn
- Отдача динамического документа напрямую через gunicorn
- Отдача динамического документа через проксирование запроса с nginx на gunicorn
- Отдача динамического документа через проксирование запроса с nginx на gunicorn, при кэшировние ответа на nginx (proxy cache)

Размеры всех документов должны быть примерно одинаковыми. По результатам измерений необходимо ответить на вопросы:

- Насколько быстрее отдается статика по сравнению с WSGI?
- Во сколько раз ускоряет работу proxy_cache?

8. Результат выполнения домашнего задания

Результат выполнения домашнего задания является:

- директория с созданным проектом
- конфиг nginx
- конфиг gunicorn
- результаты нагрузочного тестирования (вывод утилиты ab/wrk)

Проект нужно положить в систему контроля версии bitbucket.com (bitbucket.com) или github.com (github.com и) и выслать ссылку своему семинаристу письмом с пометкой **[TP]** в теме. Пометка именно такая, английскими буквами и обязательна.

9. Полезные ссылки.

- Nginx (http://nginx.org/ru/docs/)
- О проксирования в nginx (http://nginx.org/ru/docs/http/ngx_http_proxy_module.html#example)
- Gunicorn (http://docs.gunicorn.org/en/latest/configure.html)