**Практическая работа №4.**

**Теория**

1. **NGINX**

Nginx — это сервер веб-сервер и прокси-сервер, который обеспечивает высокую производительность и эффективную обработку запросов. Nginx изначально был создан для обеспечения высокой производительности в условиях высоких нагрузок на сервер, где он используется для обслуживания динамических веб-сайтов и приложений.

Одним из ключевых преимуществ Nginx является его способность обрабатывать тысячи одновременных соединений, что делает его идеальным выбором для высоконагруженных веб-сайтов и приложений. В качестве прокси-сервера Nginx используется для балансировки нагрузки и распределения запросов на различные серверы, что помогает увеличить скорость и устойчивость приложения.

Важным преимуществом Nginx является его способность к эффективной обработке статических файлов, которые обслуживаются непосредственно из кэш-памяти, что значительно ускоряет процесс обработки запросов. Кроме того, Nginx является открытым программным обеспечением, которое позволяет пользователям использовать его бесплатно, уставлять свои доработки и расширения.

В современной веб-разработке Nginx широко используется для обслуживания динамических веб-сайтов и приложений. Он также может быть использован в качестве прокси-сервера для передачи запросов на серверы приложений, таких как Apache, Tomcat и другие.

В общем, Nginx является мощным инструментом для обеспечения высокой производительности и эффективной обработки запросов, что делает его идеальным выбором для высоконагруженных веб-сайтов и приложений.

Теперь поговорим о том, как работает обратный прокси сервер. Когда клиент делает запрос к сайту, то запрос поступает на внешний IP-адрес, который установлен на nginx. Затем nginx выполняет функцию обратного проксирования и передает запрос на приложение-сервер, который может обрабатывать запросы от клиентов. После этого, первично обработанный запрос отправляется обратно на Nginx-сервер.

При использовании обратного прокси-сервера Nginx, дополнительно можно указать настройки для SSL-сертификатов и кэширования. Конфигурация сервера происходит в файле nginx.conf.

1. **Настройка NGINX**

В данном примере мы рассмотрим настройку nginx.conf в качестве обратного прокси-сервера с пояснениями и комментариями. Рассмотрим базовую конфигурацию для настройки Nginx в качестве обратного прокси-сервера.

|  |
| --- |
| events {}  http{  server {  listen 80; #порт который необходимо слушать  location / { #указание конфигурации для статических страниц  root /usr/share/nginx/html; #указания директории для статических страниц  }  location ~ \.php$ { #указание конфигурации для php страниц  proxy\_pass http://php;  proxy\_redirect http://php /;  proxy\_set\_header Host $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-Host $server\_name;  }  } |

**Задание**

Вам необходимо создать конфигурацию docker-compose для вашего веб-сервера.

Docker контейнер веб сервера должен включать:

* ОС Linux
* Веб-сервер Apache

Docker контейнер базы данных должен включать:

* ОС Linux
* Базу данных Mysql

Docker контейнер NGINX должен включать:

* ОС Linux
* NGINX

Docker-compose:

* Docker-контейнеры, созданные ранее
* Примонтированные тома
* Настройки портов
* Связь между контейнерами
* Переменные внешнего окружения
* Настройки базы данных

В рамках данного курса вам предлагается создать один большой проект маркетплейса, некоторые из практик, например, как эта, содержат кусок того проекта.

1. Создать 2 статичные html страницы для маркетплейса, где на одной странице будет информация по доставке, а на другой будут контакты и реквизиты вашего маркетплейса. (Эти страницы будут в финальном проекте)

2. Нужно сделать таким образом, чтобы статические страницы хранились в контейнере вместе с сервисом обратного прокси-сервера, а также настроить NGINX так, чтобы он мог их выдавать по запросу.

3. Создать 2 статичные html страницы. Динамические страницы могут быть взяты из предыдущих практических заданий или же написаны с нуля. Однако, необходимо сделать так, чтобы запросы и ответы на динамические страницы проходили через NGINX.

**Полезные ссылки**

1. Официальная документация NGINX: https://nginx.org/ru/
2. Статья по установке NGINX: https://www.digitalocean.com/community/tutorials/how-to-install-nginx-on-ubuntu-20-04-ru
3. Статья по настройке NGINX в качестве обратного прокси сервера: https://help.reg.ru/support/servery-vps/oblachnyye-servery/ustanovka-programmnogo-obespecheniya/kak-nastroit-nginx-v-kachestve-obratnogo-proksi-dlya-apache
4. Сбор мини статьей по настройке :https://www.dmosk.ru/categories.php?object=nginx
5. Шпаргалка по NGINX: http://idoka.ru/nginx-tips-and-tricks/

**Вопросы**

* Что такое обратный прокси сервер?
* Как можно использовать NGINX?
* Как работает кеширование в NGINX?
* Как в конфигурации NGINX добавить несколько веб-серверов динамических страниц?