Тут титульник

Оглавление

[Оглавление 2](#_Toc1)

[Почему необходимо следить за защитой веб-сервера  4](#_Toc2)

[Основные рекомендации к защите веб-серверов  5](#_Toc3)

[Сетевые ограничения  8](#_Toc4)

[Шифрование запросов  11](#_Toc5)

[Сжатие ответов сервера 12](#_Toc6)

[Базовая аутентификация  13](#_Toc7)

[Ограничения по геоданным  14](#_Toc8)

[Балансировка нагрузки  15](#_Toc9)

[Логирование  17](#_Toc10)

[ПРИЛОЖЕНИЕ А 29](#_Toc11)

[Разработанные настройки для веб-сервера nginx 29](#_Toc12)

Введение

В современном информационном обществе безопасность веб-серверов является одним из важнейших аспектов информационной безопасности. Веб-серверы играют ключевую роль в предоставлении доступа к веб-ресурсам и обработке пользовательских запросов. В этом контексте особое внимание уделяется защите веб-серверов от различных атак и уязвимостей.

Одним из самых популярных веб-серверов на сегодняшний день является Nginx. Nginx («Engine X») представляет собой легкий, высокопроизводительный сервер, который широко используется для обслуживания веб-сайтов, проксирования и балансировки нагрузки. Он также предлагает множество возможностей для обеспечения безопасности веб-сервера и защиты от различных видов атак.

Целью данной курсовой работы является исследование и анализ мер безопасности, которые могут быть применены для защиты веб-сервера Nginx. Мы рассмотрим основные уязвимости и атаки, с которыми может столкнуться веб-сервер, а также изучим различные методы и техники, которые могут быть использованы для повышения безопасности сервера Nginx.

В ходе исследования будут рассмотрены следующие аспекты защиты веб-сервера Nginx:

* Конфигурация сервера: Мы изучим важные параметры и настройки, которые могут повлиять на безопасность сервера, такие как ограничение доступа к файлам и директориям, использование SSL-сертификатов и применение правил файервола.
* Защита от DDoS-атак: Рассмотрим различные методы обнаружения и предотвращения распределенных атак отказа в обслуживании (DDoS), которые могут оказаться вредными для сервера и доступности веб-сайта.
* Фильтрация трафика: Изучим возможности фильтрации и обработки трафика, включая использование списка контроля доступа (ACL), блокировку IP-адресов, применение белых и черных списков.
* Мониторинг и регистрация: Рассмотрим важность непрерывного мониторинга и регистрации событий сервера для обнаружения подозрительной активности и быстрого реагирования на потенциальные угрозы.

В заключении работы будут представлены рекомендации и практические рекомендации по улучшению безопасности веб-сервера Nginx на основе полученных результатов и анализа. Кроме того будет предложена программа-конфигуратор для упрощения составления файла конфигурации веб-сервера. Безопасность веб-серверов является важной задачей для всех организаций, которые предоставляют свои услуги в сети интернет, и эта курсовая работа представляет собой ценный вклад в области информационной безопасности.

Почему необходимо следить за защитой веб-сервера

В последние годы количество атак на веб-серверы значительно выросло. С развитием технологий и все большей зависимости от онлайн-сервисов и электронной коммерции, веб-серверы стали привлекательной целью для злоумышленников. Некоторые факторы, способствующие увеличению количества атак на веб-серверы, включают:

* Распространение злоумышленников: С возрастанием числа злоумышленников, умеющих проводить атаки на веб-серверы, риск стал более значимым. Легко доступные инструменты и ресурсы в Интернете позволяют даже неопытным злоумышленникам осуществлять атаки на веб-серверы.
* Уязвимости веб-приложений: Часто веб-приложения содержат уязвимости, которые могут быть использованы злоумышленниками для атак на сервер. Отсутствие должной обработки и валидации входных данных, уязвимости в коде приложения или ошибки в конфигурации сервера могут предоставить злоумышленникам доступ к серверу.
* Развитие новых типов атак: Злоумышленники постоянно разрабатывают новые методы и техники для атак на веб-серверы. Это включает в себя более сложные DDoS-атаки, передовые методы SQL-инъекций, улучшенные техники фишинга и другие инновационные способы эксплуатации уязвимостей.
* Коммерческая ценность данных: Веб-серверы часто содержат ценную информацию, такую как финансовые данные, персональные данные пользователей, коммерческие секреты и другие конфиденциальные сведения. Захват или кража таких данных может принести значительную финансовую прибыль злоумышленникам, стимулируя их к проведению атак.
* Распространение ботнетов: Ботнеты, состоящие из множества зараженных компьютеров или устройств, используются для проведения массовых атак на веб-серверы. Эти ботнеты контролируются злоумышленниками и могут использоваться для запуска DDoS-атак или для эксплуатации уязвимостей на сервере.

Вот примеры нескольких атак, произведенных на крупные компании за последние годы:

* Атака на Sony PlayStation Network (2011): В 2011 году Sony PlayStation Network столкнулся с одной из наиболее серьезных атак на веб-инфраструктуру. Злоумышленники скомпрометировали сетевую инфраструктуру, что привело к утечке личных данных более 77 миллионов пользователей, включая имена, адреса электронной почты, пароли и финансовую информацию.
* Атака на Equifax (2017): В 2017 году компания Equifax, одна из трех крупнейших кредитных бюро в США, столкнулась с серьезной атакой. Злоумышленники эксплуатировали уязвимость веб-приложения, что позволило им получить доступ к личным данным около 147 миллионов человек, включая имена, социальные номера, даты рождения и номера кредитных карт.
* Атака на Yahoo (2013-2014): В 2013-2014 годах Yahoo столкнулся с масштабной атакой, в результате которой были скомпрометированы данные более 3 миллиардов пользователей. Злоумышленники получили доступ к учетным записям, включая имена, адреса электронной почты, хэшированные пароли и секретные вопросы безопасности.

Это всего лишь некоторые примеры успешных атак на веб-инфраструктуру, и с течением времени появляются новые методы и уязвимости, которые могут быть эксплуатированы злоумышленниками. Важно постоянно обновлять свои знания о безопасности и применять соответствующие меры для защиты веб-серверов и веб-приложений.

Основные рекомендации к защите веб-серверов

За стандартами защиты веб-серверов следят различные организации и стандартизационные органы, такие как OWASP (Open Web Application Security Project), NIST (National Institute of Standards and Technology) или ENISA (European Union Agency for Cybersecurity), а также сообщества экспертов в области информационной безопасности. Они играют важную роль в разработке и установлении лучших практик и рекомендаций по обеспечению безопасности веб-серверов.Вот основные из них:

* Обновляйте программное обеспечение: Регулярно обновляйте операционную систему, веб-сервер и все установленные компоненты и приложения до последних версий. Обновления часто содержат исправления уязвимостей, которые могут быть использованы злоумышленниками.
* Используйте сильные пароли: Установите сложные пароли для учетных записей администратора и других пользователей, а также для базы данных. Избегайте использования стандартных паролей и регулярно меняйте пароли.
* Применяйте фильтрацию трафика: Используйте межсетевые экраны (firewalls) и системы обнаружения вторжений (IDS/IPS) для фильтрации и контроля входящего и исходящего сетевого трафика. Это поможет блокировать подозрительные пакеты данных и защитить сервер от многих типов атак.
* Защитите от DDoS-атак: Реализуйте механизмы для обнаружения и смягчения DDoS-атак, такие как использование услуги облачной защиты от DDoS или настройка сетевых устройств для отсеивания вредоносного трафика.
* Применяйте принцип наименьших привилегий: Ограничьте привилегии учетных записей, чтобы минимизировать потенциальные последствия компрометации. Пользовательские учетные записи должны иметь только необходимые права доступа к ресурсам.
* Фильтруйте входящие данные: Валидируйте и санитизируйте входящие данные, особенно если они передаются в базу данных или выполняются на сервере. Это поможет предотвратить SQL-инъекции и XSS-атаки.
* Шифруйте соединение: Используйте протокол HTTPS с помощью сертификатов SSL/TLS для защиты передачи данных между клиентом и сервером. Это поможет предотвратить перехват информации и подделку данных.
* Регулярно создавайте резервные копии: Регулярное резервное копирование данных позволяет восстановить сервер в случае успешной атаки или сбоя. Убедитесь, что резервные копии хранятся в надежном месте, отдельно от основного сервера.
* Мониторинг и журналирование: Внедрите системы мониторинга, которые следят за активностью сервера и обнаруживают подозрительные или необычные события. Хороший журнал событий поможет вам исследовать инциденты и принять меры по предотвращению будущих атак.

Это лишь некоторые рекомендации, и полная защита веб-сервера требует комплексного подхода, учета специфических потребностей вашего сервера и постоянного обновления знаний о безопасности.

О выборе веб-сервера

Правильный выбор веб-сервера обеспечивает оптимальное функционирование веб-приложения, удовлетворяя требованиям бизнеса и ожиданиям пользователей. Он позволяет эффективно обрабатывать запросы, предоставлять контент быстро и без проблем, а также адаптироваться к растущим потребностям и нагрузке.

Кроме того, правильный выбор веб-сервера имеет прямое отношение к безопасности вашего веб-приложения. Надежный веб-сервер обеспечивает защиту от различных угроз и атак, минимизируя риски утечки данных или компрометации системы.

В своей работе я буду использовать Nginx (Engine-X) — это мощный веб-сервер и прокси-сервер, который выполняет ряд функций и может быть использован в различных сценариях. Вот некоторые основные области применения Nginx:

* Веб-сервер: Одной из основных функций Nginx является обслуживание веб-содержимого. Он способен обрабатывать статические файлы, такие как HTML, CSS, JavaScript и изображения. Благодаря своей высокой производительности и эффективному использованию ресурсов, Nginx позволяет эффективно обслуживать большое количество запросов, особенно в высоконагруженных средах.
* Обратный прокси: Nginx часто используется в качестве обратного прокси-сервера, который принимает запросы от клиентов и перенаправляет их на соответствующие веб-серверы. Это позволяет балансировать нагрузку между несколькими серверами и повышает отказоустойчивость, так как приложения на серверах могут быть легко масштабируемы.
* Балансировка нагрузки: Nginx предоставляет возможности балансировки нагрузки, которые позволяют распределять запросы равномерно между несколькими серверами. Это помогает оптимизировать использование ресурсов и обеспечивает более высокую доступность веб-приложений.
* Кэширование: Nginx поддерживает функцию кэширования, которая позволяет сохранять статические ресурсы, такие как изображения или файлы CSS/JavaScript, в оперативной памяти или на диске. Это сокращает нагрузку на сервер и ускоряет время загрузки страниц для повторных запросов.
* SSL/TLS терминирование: Nginx может выполнять функцию терминирования SSL/TLS, что позволяет осуществлять шифрование и расшифровку данных между клиентом и сервером. Это обеспечивает безопасную передачу данных и защиту от перехвата информации.
* Проксирование API: Nginx может быть использован для проксирования запросов к внутренним или внешним API. Это позволяет контролировать доступ, управлять авторизацией и маршрутизацией запросов к API.
* Управление статическими файлами и медиа-контентом: Nginx может использоваться для эффективного доставки статических файлов и медиа-контента, таких как видео или аудиофайлы. Это позволяет обеспечить быструю и надежную доставку контента конечным пользователям.

Сочетание высокой производительности, гибкости и богатого функционала делает Nginx популярным выбором для веб-серверов, обратных прокси и балансировщиков нагрузки во многих веб-приложениях и средах разработки.

Далее будут представлены основные настройки для nginx сервера, применяемые для защиты сервера.

Основные настройки Nginx

О структуре файла конфигурации

Файл конфигурации Nginx, известный как nginx.conf, определяет основные настройки и параметры работы веб-сервера Nginx. Вот общая структура файла конфигурации nginx.conf:

* Директивы глобального блока (http): В этом блоке определяются глобальные настройки для всего веб-сервера. Включает в себя директивы, такие как user, worker\_processes, events и другие, которые задают общие параметры работы сервера.
* Блок events: Здесь определяются параметры событийной модели, такие как количество рабочих процессов (worker\_processes), метод обработки событий и другие настройки, связанные с обработкой событий сервером.
* Блок http: В этом блоке определяются основные параметры и настройки HTTP-протокола. Он включает в себя блок server, который может быть повторен несколько раз для определения разных виртуальных хостов и их настроек.
* Блок server: Каждый блок server определяет настройки для конкретного виртуального хоста или сервера. В этом блоке определяются параметры, такие как listen, server\_name, location и другие, которые управляют поведением сервера для конкретного хоста.
* Блок location: Блок location определяет настройки для обработки запросов, соответствующих определенному пути URL. Здесь можно определить параметры, такие как root, proxy\_pass, rewrite и другие, чтобы настроить обработку запросов для конкретного пути.
* Другие блоки и директивы: В файле конфигурации nginx.conf могут быть определены и другие блоки и директивы для специфических настроек и модулей, таких как SSL/TLS, кэширование, сжатие и другие.

Структура файла конфигурации nginx.conf может различаться в зависимости от конкретных потребностей и настроек веб-сервера. Рекомендуется внимательно изучить документацию Nginx и следовать принятой структуре и синтаксису для корректной настройки сервера.

## Сетевые ограничения

Эти настройки позволяют вам определить различные параметры работы сервера Nginx, такие как порты, доступ по IP-адресам, размеры буферов, таймауты и другие параметры. Изменение этих настроек позволяет адаптировать сервер к конкретным требованиям приложения и повысить его производительность и безопасность.

http{

*#limit concurrency*

limit\_conn\_zone $server\_name zone=per\_vhost:5m;

limit\_conn\_zone $binary\_remote\_addr zone=per\_ip:5m;

server{

listen 81;

deny 192.168.0.170;

deny 192.168.0.171;

allow 192.168.0.174;

server\_name localhost;

*#buffer sizes*

client\_body\_buffer\_size 16k;

client\_header\_buffer\_size 1k;

client\_max\_body\_size 8m;

large\_client\_header\_buffers 2 1k;

*#timeouts*

client\_body\_timeout 12;

client\_header\_timeout 12;

*#keepalive*

keepalive\_timeout 65;

send\_timeout 10;

*#server version info off*

server\_tokens off;

...

location ~\* \.(css|js|jpg|png|gif)$ {

...

limit\_conn per\_ip 1;

...

}

}

}

Рассмотрим каждую из указанных настроек в контексте Nginx:

* **limit\_conn\_zone $server\_name zone=per\_vhost:5m** и **limit\_conn\_zone $binary\_remote\_addr zone=per\_ip:5m**: Эти директивы определяют ограничения на количество одновременных подключений для каждого виртуального хоста (per\_vhost) и для каждого IP-адреса клиента (per\_ip). Здесь указаны размеры зон памяти (5m), которые используются для отслеживания подключений.
* **listen 81**: Эта директива указывает, что сервер Nginx будет слушать входящие подключения на порту 81.
* **deny 192.168.0.170, deny 192.168.0.171** и **allow 192.168.0.174**: Эти директивы управляют доступом к серверу на основе IP-адреса клиента. Клиенты с IP-адресами 192.168.0.170 и 192.168.0.171 будут запрещены, а клиент с IP-адресом 192.168.0.174 будет разрешен.
* **server\_name localhost**: Эта директива определяет имя сервера, к которому применяются настройки в данном блоке конфигурации. В данном случае, сервер будет отвечать на запросы с хостом "localhost".
* **client\_body\_buffer\_size 16k**, **client\_header\_buffer\_size 1k, client\_max\_body\_size 8m, large\_client\_header\_buffers 2 1k**: Эти настройки связаны с размерами буферов, используемых для обработки тела запроса клиента (client\_body\_buffer\_size), заголовков клиента (client\_header\_buffer\_size), максимального размера тела запроса клиента (client\_max\_body\_size) и больших буферов для заголовков клиента (large\_client\_header\_buffers).
* **client\_body\_timeout 12**, **client\_header\_timeout 12**: Эти настройки определяют таймауты ожидания запроса от клиента для тела (client\_body\_timeout) и заголовков (client\_header\_timeout).
* **keepalive\_timeout 65**, **send\_timeout 10**: Эти настройки определяют таймауты соединения keep-alive (keepalive\_timeout) и отправки данных на сервер (send\_timeout).
* **server\_tokens off**: Эта директива отключает отправку информации о версии сервера в заголовках ответа, чтобы уменьшить возможность идентификации сервера в случае потенциальных атак.
* **location ~\* \.(css|js|jpg|png|gif)$ { ... }**: Это блок конфигурации для обработки запросов к статическим файлам с расширениями .css, .js, .jpg, .png и .gif. В данном случае, внутри блока могут быть указаны дополнительные настройки, например, limit\_conn per\_ip 1;, которая ограничивает количество одновременных подключений с одного IP-адреса до 1.

Кэширование запросов

 Кэширование запросов в Nginx - это процесс сохранения результатов запросов на сервере, чтобы при последующих запросах на тот же ресурс сервер мог возвращать результаты из кэша, без необходимости выполнения полной обработки запроса.

При использовании кэширования, Nginx может значительно сократить нагрузку на сервер, улучшить скорость ответа и снизить задержки для конечных пользователей. Когда клиент отправляет запрос, Nginx проверяет наличие соответствующей записи в кэше. Если запись присутствует и не устарела, сервер может немедленно вернуть результат клиенту без обращения к бэкенд-серверу.

http{

include fastcgi\_params;

include fastcgi.conf;

*#fastCGI*

fastcgi\_cache\_path /etc/nginx/cache levels=1:2 keys\_zone=microcache:10m max\_size=500m inactive=10m;

fastcgi\_cache\_key "$scheme$request\_method$host$request\_uri";

fastcgi\_ignore\_headers Cache-Control Expires **Set**-Cookie ;

add\_header caching $upstream\_cache\_status;

server{

...

location ~\* \.(css|js|jpg|png|gif)$ {

...

expires 1M;

...

}

...

location /testphp {

fastcgi\_cache microcache;

fastcgi\_cache\_valid 200 60m;

fastcgi\_pass 127.0.0.1:9000;

...

}

}

}

В приведенном примере конфигурации Nginx, рассмотрим несколько соответствующих настроек для кэширования:

* **fastcgi\_cache\_path /etc/nginx/cache levels=1:2 keys\_zone=microcache:10m max\_size=500m inactive=10m**: Эта настройка определяет путь к директории, где будут храниться кэшированные данные (/etc/nginx/cache), уровни директорий (levels=1:2), зону ключей (keys\_zone=microcache:10m), максимальный размер кэша (max\_size=500m) и время неактивности после которого записи в кэше считаются устаревшими (inactive=10m).
* **fastcgi\_cache\_key "$scheme$request\_method$host$request\_uri"**: Эта директива определяет ключ, по которому происходит кэширование запросов FastCGI. Ключ формируется на основе схемы ($scheme), метода запроса ($request\_method), хоста ($host) и URI запроса ($request\_uri).
* **fastcgi\_cache microcache**: Эта директива указывает, что запросы, соответствующие данной локации, должны быть кэшированы в зоне ключей microcache.
* **fastcgi\_cache\_valid 200 60m**: Эта директива определяет время жизни кэшированной записи с кодом ответа 200 (успешный ответ) в течение 60 минут.

Таким образом, с помощью этих настроек Nginx может кэшировать ответы FastCGI и возвращать их непосредственно из кэша, минуя обращение к бэкенд-серверу, если записи в кэше существуют и не устарели. Это позволяет снизить нагрузку на сервер и сократить время обработки запросов, повышая производительность и улучшая отзывчивость веб-приложения.

## Шифрование запросов

Шифрование и SSL (Secure Sockets Layer) являются важными аспектами безопасности веб-серверов. Nginx предоставляет возможность использовать SSL/TLS для шифрования соединения между клиентом и сервером.

SSL/TLS - это протоколы, обеспечивающие шифрование данных и аутентификацию для безопасной передачи информации по сети. Они используют криптографические алгоритмы для защиты конфиденциальности, целостности и подлинности данных.

http{

server{

listen 443 ssl;

*#ssl*

ssl\_certificate /etc/nginx/ssl/nginx.crt;

ssl\_certificate\_key /etc/nginx/ssl/nginx.key;

ssl\_session\_cache shared:SSL:1m;

ssl\_session\_timeout

ssl\_ciphers HIGH:!aNULL:!MD5;

ssl\_prefer\_server\_ciphers on;

}

}

В данном примере конфигурации Nginx демонстрируется использование шифрования запросов с помощью протокола SSL/TLS. Давайте рассмотрим каждую из указанных настроек:

* l**isten 443 ssl;**: Эта директива указывает на прослушивание порта 443 (стандартный порт для HTTPS) и использование протокола SSL/TLS для шифрования соединения между клиентом и сервером.
* **ssl\_certificate /etc/nginx/ssl/nginx.crt;** и **ssl\_certificate\_key /etc/nginx/ssl/nginx.key;**: Эти директивы указывают пути к сертификату и приватному ключу, необходимым для установки безопасного соединения. В данном случае, указываются пути к файлам сертификата (nginx.crt) и приватного ключа (nginx.key).
* **ssl\_session\_cache shared:SSL:1m;** и **ssl\_session\_timeout**: Эти директивы определяют настройки кэша сеансов SSL/TLS. shared:SSL:1m указывает, что кэш сеансов должен быть разделен между несколькими воркерами и иметь размер 1 мегабайт. ssl\_session\_timeout 5m; определяет время жизни сеансов в кэше.
* **ssl\_ciphers HIGH:!aNULL:!MD5;**: Эта директива определяет список шифров, которые могут быть использованы при установке SSL/TLS-соединения. В данном случае, используются шифры с высоким уровнем безопасности и отключены анонимные шифры и шифры, использующие алгоритм MD5.
* **ssl\_prefer\_server\_ciphers on;**: Эта директива указывает серверу предпочитать шифры, предложенные клиентом, при установке SSL/TLS-соединения.

Эти настройки позволяют использовать SSL/TLS для шифрования запросов между клиентом и сервером, обеспечивая безопасность и защиту передаваемых данных от перехвата или нежелательного доступа.

## Сжатие ответов сервера

Использование сжатия респонзов с помощью gzip позволяет уменьшить размер передаваемых данных между сервером и клиентом, что улучшает скорость загрузки страницы и экономит пропускную способность сети. Это особенно полезно при передаче больших текстовых файлов, стилей CSS и JavaScript, которые часто могут быть сжаты существенно без потери качества.

http{

server {

*#gzip*

gzip on;

gzip\_min\_length 100;

gzip\_comp\_level 3;

gzip\_types text/plain;

gzip\_types text/css;

gzip\_types text/javascript;

gzip\_disable "msie6";

}

}

В данном примере конфигурации Nginx демонстрируется использование сжатия респонзов с помощью gzip. Давайте рассмотрим каждую из указанных настроек:

* **gzip on;**: Эта директива включает сжатие респонзов с помощью gzip.
* **gzip\_min\_length 100;**: Эта директива указывает минимальный размер ответа, который будет сжиматься. В данном случае, ответы, размером менее 100 байт, не будут сжиматься.
* **gzip\_comp\_level 3;**: Эта директива задает уровень компрессии для сжатия gzip. Уровень 3 является стандартным и обеспечивает хороший баланс между скоростью и степенью сжатия.
* **gzip\_types text/plain; gzip\_types text/css; gzip\_types text/javascript;**: Эти директивы указывают типы контента, которые могут быть сжаты с помощью gzip. В данном случае, текстовые файлы (text/plain), CSS-файлы (text/css) и JavaScript-файлы (text/javascript) будут сжиматься.
* **gzip\_disable "msie6";**: Эта директива позволяет отключить сжатие для устаревших браузеров, в данном случае для Internet Explorer 6. Такие браузеры не всегда могут правильно обрабатывать сжатые респонзы, поэтому можно отключить сжатие для них.

## Базовая аутентификация

Базовая аутентификация (Basic Authentication) - это простой механизм аутентификации, который использует комбинацию имени пользователя и пароля для ограничения доступа к ресурсам сервера. В Nginx базовая аутентификация может быть легко настроена с использованием модуля ngx\_http\_auth\_basic.

http{

server{

location /profile {

auth\_basic "Restricted folder";

*#base auth*

auth\_basic\_user\_file /etc/nginx/creds/.htpasswd;

access\_log /var/log/nginx/new.log upstream\_time;

}

}

}

В данной конфигурации Nginx используется базовая аутентификация для ограничения доступа к пути /profile. Давайте рассмотрим каждую из указанных настроек:

* **location /profile { ... }**: Директива location определяет путь к ресурсу, для которого будет применяться базовая аутентификация. В данном случае, это путь /profile.
* a**uth\_basic "Restricted folder";**: Эта директива устанавливает сообщение, которое будет отображаться в диалоговом окне аутентификации браузера, при попытке доступа к ограниченной папке /profile. В данном случае, сообщение будет "Restricted folder".
* **auth\_basic\_user\_file /etc/nginx/creds/.htpasswd;**: Эта директива указывает путь к файлу .htpasswd, который содержит пары "имя пользователя:зашифрованный пароль". В данном случае, файл .htpasswd находится по пути /etc/nginx/creds/.htpasswd. Этот файл должен быть создан и содержать допустимые учетные данные для аутентификации пользователей.
* **access\_log /var/log/nginx/new.log upstream\_time;**: Эта директива настраивает запись журнала доступа в файл /var/log/nginx/new.log с указанием времени выполнения каждого запроса (upstream\_time). Журнал доступа содержит информацию о запросах к ресурсу /profile.

Таким образом, при доступе к пути /profile, браузер будет запрашивать у пользователя имя пользователя и пароль. Введенные учетные данные будут проверяться на соответствие с данными из файла .htpasswd. Если аутентификация прошла успешно, пользователю будет разрешен доступ к ограниченной папке /profile, а информация о запросах будет записываться в указанный журнал доступа.

Эта конфигурация позволяет ограничить доступ к конкретной папке на сервере с использованием базовой аутентификации, что обеспечивает дополнительный уровень защиты и контроля доступа к конфиденциальной информации.

## Ограничения по геоданным

Nginx GeoIP2 - это модуль, который позволяет использовать информацию о географическом расположении клиентов веб-сервера на основе их IP-адресов. Этот модуль использует базу данных GeoIP2, которая содержит информацию о стране, регионе, городе, координатах и других атрибутах, связанных с конкретными IP-адресами.

Для работы с данным модулем необходимо загрузить в конфигурацию сервера модуль ngx\_http\_geoip2, а так же предоставить серверу доступ к базе геоданных, в нашем случае использовалась GeoLite2-City.mmdb.

http{

*#geoip*

geoip2 /usr/share/GeoIP/GeoLite2-City.mmdb {

auto\_reload 60m;

$geoip2\_metadata\_city\_build metadata build\_epoch;

$geoip2\_data\_country\_name country names en;

$geoip2\_data\_country\_code country iso\_code;

$geoip2\_data\_city\_name city names en;

$geoip2\_data\_region\_name subdivisions 0 names en;

$geoip2\_data\_state\_code subdivisions 0 iso\_code;

}

geoip blocking

map "$geoip2\_data\_country\_code:$geoip2\_data\_state\_code" $allowed\_reg {

default no;

~^RU: yes; *#Россия*

~^BY: yes; *#Белоруссия*

~^AM: yes; *#Армения*

~^KZ: yes; *#Казахстан*

~^KG: yes; *#Кыргызстан*

UA:40 yes; *#Севастополь*

UA:43 yes; *#Крым*

UA:14 yes; *#Донецкая область*

UA:09 yes; *#Луганская область*

UA:23 yes; *#Запорожская область*

UA:65 yes; *#Херсонская область*

GE:AB yes; *#Абхазия*

}

server{

}

}

Давайте рассмотрим каждую из указанных настроек:

* **geoip2 /usr/share/GeoIP/GeoLite2-City.mmdb { ... }**: Директива geoip2 указывает путь к файлу базы данных GeoIP2, который содержит информацию о геолокации IP-адресов. В данном случае, файл находится по пути /usr/share/GeoIP/GeoLite2-City.mmdb. Дополнительно указаны переменные, которые будут содержать информацию о стране, регионе, городе и других атрибутах.
* **auto\_reload 60m;**: Эта настройка указывает интервал автоматической перезагрузки базы данных GeoIP2. В данном случае, база данных будет перезагружаться каждые 60 минут.
* **$geoip2\_metadata\_city\_build metadata build\_epoch;**: Эта переменная содержит информацию о времени последнего обновления базы данных GeoIP2.
* **$geoip2\_data\_country\_name country names en;**: Эта переменная содержит название страны на основе IP-адреса клиента.
* **$geoip2\_data\_country\_code country iso\_code;**: Эта переменная содержит код страны (двухбуквенный ISO-код) на основе IP-адреса клиента.
* **$geoip2\_data\_city\_name city names en;**: Эта переменная содержит название города на основе IP-адреса клиента.
* **$geoip2\_data\_region\_name subdivisions 0 names en;**: Эта переменная содержит название региона на основе IP-адреса клиента.
* **$geoip2\_data\_state\_code subdivisions 0 iso\_code;**: Эта переменная содержит код региона (двухбуквенный ISO-код) на основе IP-адреса клиента.
* **geoip blocking;**: Эта директива включает блокировку доступа на основе географической локации. Если клиент находится в запрещенном регионе, его доступ будет заблокирован.
* **map "$geoip2\_data\_country\_code:$geoip2\_data\_state\_code" $allowed\_reg { ... }**: Эта директива определяет переменную $allowed\_reg, которая будет содержать значение yes или no в зависимости от географической локации клиента. В данном случае, используется регулярное выражение для определения разрешенных регионов. Если соответствующий регион найден в базе данных GeoIP2, переменная будет иметь значение yes, в противном случае - no.
* **if ($allowed\_reg = no) { return 444; }**: Эта конструкция проверяет значение переменной $allowed\_reg. Если значение равно no, то выполняется директива return 444, которая прекращает обработку запроса и возвращает ошибку 444 ("No Response") клиенту. Это позволяет блокировать доступ к серверу для клиентов из запрещенных регионов.

Обратите внимание, что данная конфигурация может быть дополнена другими настройками сервера внутри блока server { ... }.

## Балансировка нагрузки

Nginx предоставляет несколько алгоритмов балансировки нагрузки, которые определяют способ распределения входящих запросов между серверами в группе. Каждый алгоритм имеет свои особенности и может быть выбран в зависимости от требований вашей системы. Ниже приведены основные алгоритмы балансировки нагрузки, поддерживаемые Nginx:

* **Round Robin** (Поочередный выбор):
  + Алгоритм по умолчанию.
  + Запросы распределяются по серверам в группе в порядке их указания.
  + При каждом новом запросе выбирается следующий сервер в порядке списка.
  + Простой и равномерный способ распределения нагрузки.
* **Least Connections** (Выбор сервера с наименьшим количеством активных соединений):
  + Запросы направляются на сервер с наименьшим количеством активных соединений.
  + Позволяет распределить нагрузку более равномерно, учитывая текущую нагруженность серверов.
* **IP Hash** (Хеширование по IP-адресу):
  + Каждый клиентский IP-адрес сопоставляется с конкретным сервером.
  + Позволяет обеспечить сохранение состояния сессии для клиента на протяжении всего времени взаимодействия с сервером.
  + Гарантирует, что все запросы от одного клиента будут направлены на один и тот же сервер.
* **Generic Hash** (Общее хеширование):
  + Запросы хешируются с использованием произвольного ключа, указанного в конфигурации.
  + Позволяет гибко настраивать способ хеширования для распределения нагрузки на основе определенных параметров запроса или других данных.

http{

upstream php\_servers {

*#hash $scheme$request\_uri;*

*#ip\_hash;*

least\_conn;

*#random two least\_conn;*

*#least\_time header;*

*#least\_time last\_byte;*

server localhost:10001;

server localhost:10002;

server localhost:10003;

}

server{

listen 81;

listen 443 ssl;

*# proxy the PHP scripts to Apache listening on 127.0.0.1:9000*

*# testing load balancers*

location /testphp {

fastcgi\_cache microcache;

fastcgi\_cache\_valid 200 60m;

fastcgi\_pass 127.0.0.1:9000;

proxy\_set\_header proxy\_header\_to\_server nginx;

add\_header proxy\_header nginx;

proxy\_pass http://php\_servers;

}

}

}

В данном примере конфигурации Nginx используется алгоритм балансировки нагрузки "Least Connections" для группы серверов php\_servers. Давайте рассмотрим, что делает каждая настройка:

* **least\_conn**: Этот алгоритм направляет запросы на сервер с наименьшим количеством активных соединений. Он распределяет нагрузку равномерно между серверами, учитывая их текущую нагруженность. Серверы localhost:10001, localhost:10002 и localhost:10003 указаны в качестве серверов для балансировки.
* **listen 81; и listen 443 ssl;**: Эти директивы указывают Nginx слушать соединения на портах 81 и 443 с использованием SSL/TLS.
* **location /testphp**: В этом блоке настраивается обработка запросов, которые соответствуют пути /testphp. Они проксируются на серверы из группы php\_servers с использованием балансировки нагрузки. Также присутствуют дополнительные настройки, такие как кеширование (fastcgi\_cache) и передача заголовков (proxy\_set\_header, add\_header).

В данной конфигурации алгоритм балансировки нагрузки "Least Connections" выбран для равномерного распределения запросов между серверами localhost:10001, localhost:10002 и localhost:10003. Это позволяет достичь более эффективного использования ресурсов и более высокой отказоустойчивости системы.

## Логирование

Nginx логирование играет важную роль, позволяя отслеживать и анализировать различные события, ошибки и активность сервера. Nginx предлагает различные типы логов, которые можно настроить в файле конфигурации nginx.conf. Давайте рассмотрим основные типы логов и их назначение:

* **Access logs** (логи доступа): Они записывают информацию о каждом запросе, поступающем на сервер Nginx. Access logs содержат информацию, такую как IP-адрес клиента, время запроса, HTTP-метод, запрошенный URL, код состояния ответа и объем переданных данных. Эти логи полезны для мониторинга активности сервера и анализа трафика.
* **Error logs** (логи ошибок): Они регистрируют различные ошибки, возникающие в процессе обработки запросов. Error logs включают сообщения об ошибках, критические события и предупреждения, связанные с работой сервера. Эти логи помогают в выявлении проблем и их диагностике для обеспечения стабильности и безопасности сервера.
* **Application logs** (логи приложений): Если вы используете Nginx в качестве прокси или обратного прокси для приложений, таких как веб-серверы или приложения на основе фреймворка, то вы можете настроить логирование событий, связанных с вашими приложениями. Это позволяет вам отслеживать действия и проблемы, возникающие в приложениях.

Для каждого из этих типов логов в Nginx можно настроить формат записей, место хранения файлов логов и уровень подробности записываемых сообщений. Это позволяет администраторам настроить логирование согласно своим требованиям и предпочтениям.

http{

*#log formatting*

log\_format upstream\_time '$remote\_addr - $remote\_user [$time\_local] '

'"$request" $status $body\_bytes\_sent '

'"$http\_referer" "$http\_user\_agent"'

'rt=$request\_time uct="$upstream\_connect\_time" uht="$upstream\_header\_time" urt="$upstream\_response\_time"';

log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

'$status $body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

*#access\_log logs/access.log main;*

server{

*#logs off*

error\_log off;

access\_log off;

*#access\_log logs/host.access.log main;*

location /profile {

error\_log /var/log/nginx/errors\_profile.log main

access\_log /var/log/nginx/access\_profile.log upstream\_time;

}

}

}

В представленной конфигурации Nginx определены два формата логов: upstream\_time и main. Давайте разберем, что делают эти настройки:

* **log\_format upstream\_time**: Этот формат логов определяет пользовательский формат записей для логов доступа. Он содержит следующие переменные и данные:
* **$remote\_addr**: IP-адрес клиента
* **$remote\_user**: имя пользователя (если используется базовая аутентификация)
* **$time\_local**: локальное время запроса
* **"$request"**: сам запрос
* **$status**: код состояния ответа сервера
* **$body\_bytes\_sent**: размер ответа в байтах
* **"$http\_referer"**: HTTP-заголовок Referer (если присутствует)
* **"$http\_user\_agent"**: HTTP-заголовок User-Agent
* **rt=$request\_time**: время обработки запроса
* **uct="$upstream\_connect\_time"**: время установки соединения с бэкенд-сервером
* **uht="$upstream\_header\_time"**: время получения заголовков ответа от бэкенд-сервера
* **urt="$upstream\_response\_time"**: время получения ответа от бэкенд-сервера
* **log\_format main**: Этот формат логов также определяет пользовательский формат записей для логов доступа. Он содержит переменные и данные, такие как IP-адрес клиента, имя пользователя (если есть), локальное время запроса, сам запрос, код состояния ответа сервера, размер ответа, HTTP-заголовки Referer и User-Agent, а также заголовок X-Forwarded-For (если присутствует).

Внутри блока server для пути /profile определены настройки логирования:

* **error\_log /var/log/nginx/errors\_profile.log main**: Указывает путь к файлу, в который будут записываться ошибки, связанные с обработкой запросов для данного пути.
* **access\_log /var/log/nginx/access\_profile.log upstream\_time;**: Указывает путь к файлу, в который будут записываться логи доступа для данного пути, используя формат upstream\_time для форматирования записей логов.

Обратите внимание, что логирование в блоке server может быть включено или выключено с помощью директив error\_log и access\_log. В представленной конфигурации логирование в целом выключено для данного сервера, но включено и настроено для пути /profile.

Настраивая логирование в Nginx, вы можете контролировать формат записей, выбирать, какие данные включать, и указывать файлы, в которые записывать логи. Это помогает в мониторинге и отладке сервера, а также в получении полезной информации о запросах и ошибках.

# Описание программы-конфигуратора formatter

Рассмотренный выше файл конфигурации веб-сервера nginx был прописан вручную. Однако данный подход занял довольно много времени, поскольку стандартный файл конфигурации, созданный при установке веб-сервера, содержал малое количество полезных настроек. Кроме того было необходимо периодически тестировать файл на соответствие синтаксису nginx и убеждаться в том, что настройки действительно применяются. Данные проблемы можно решить с помощью применения скрипта-конфигуратора. Написание подобного кода на языке c++ составляет практическую часть данной курсовой работы.

Программу-конфигуратор необходимо скомпиллировать и запустить в терминале с помощью следующей команды:

g++ formatter.cpp –o formatter &amp; ./formatter

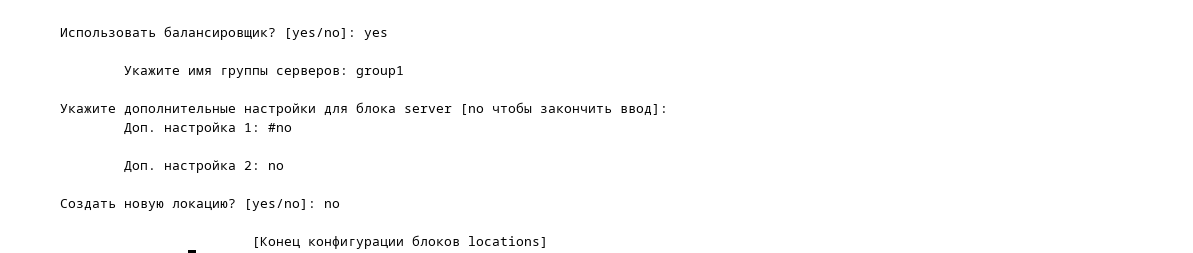
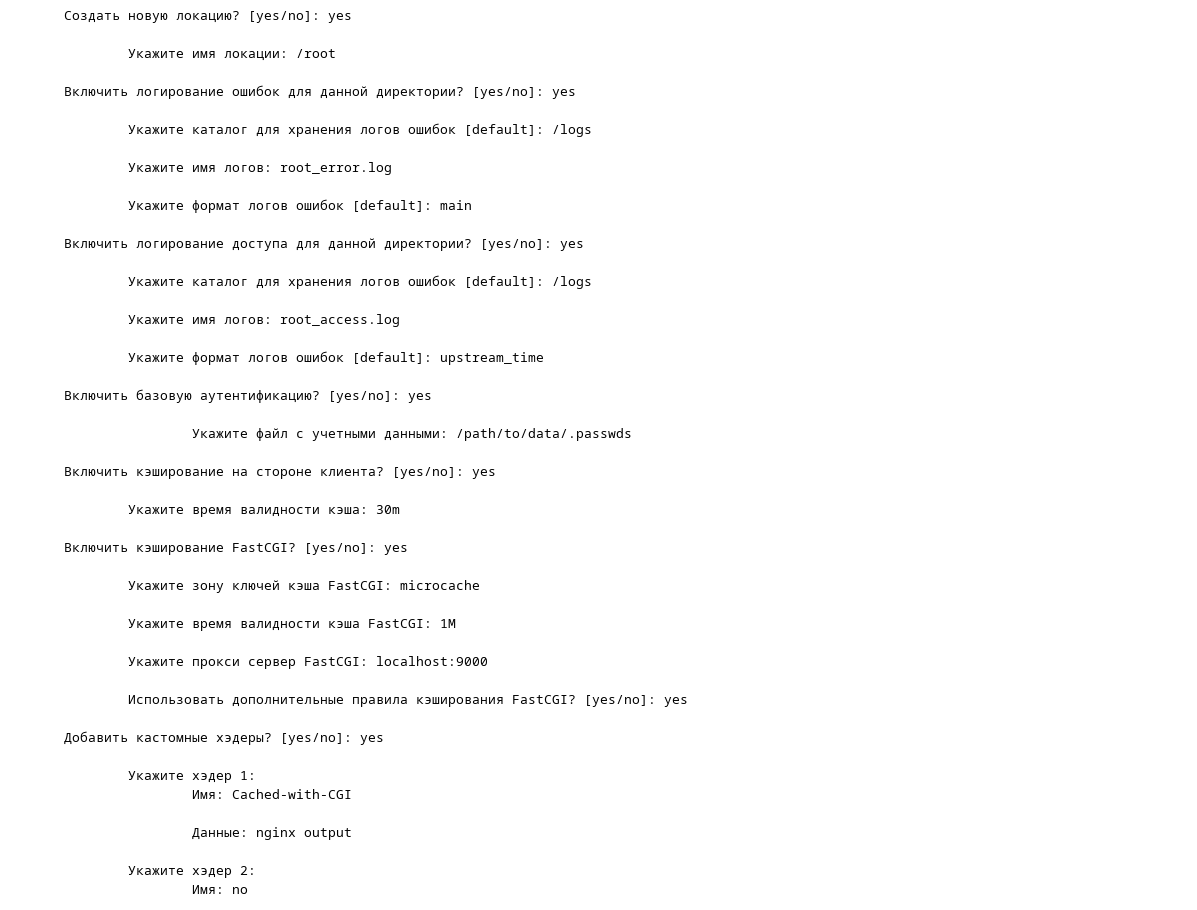
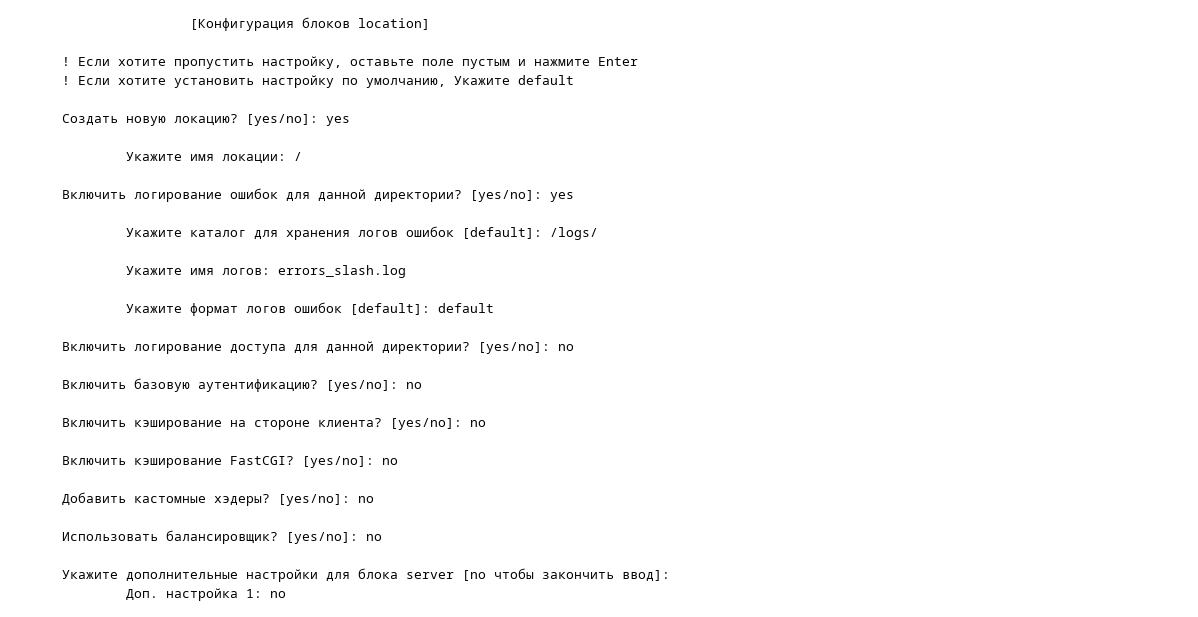
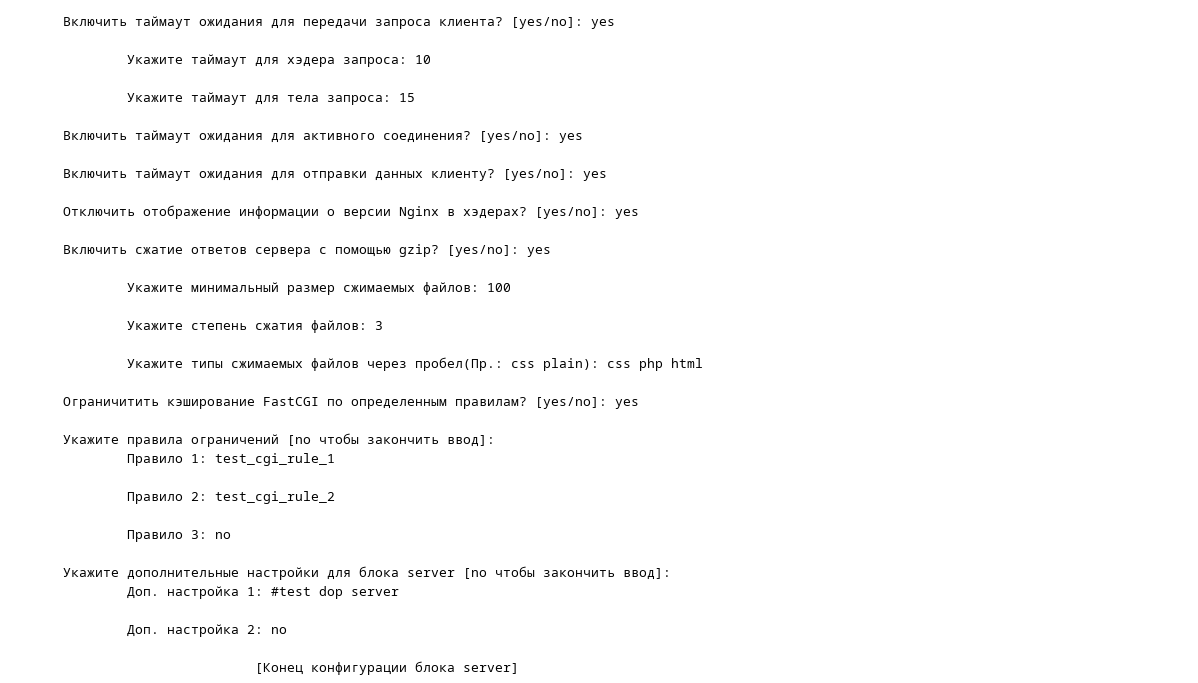
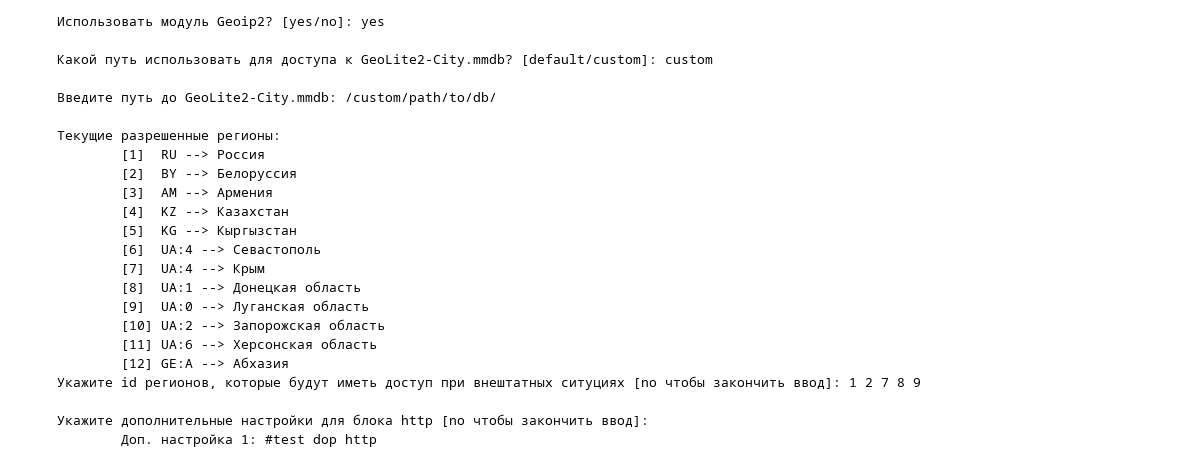
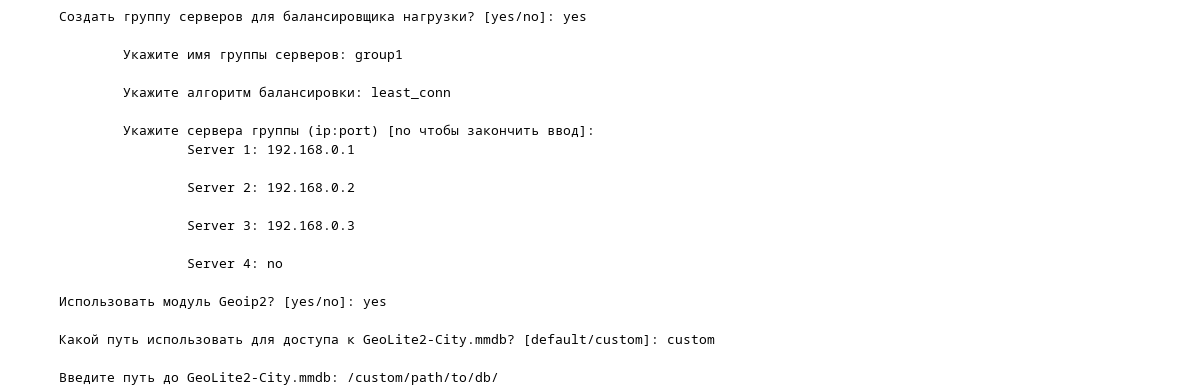
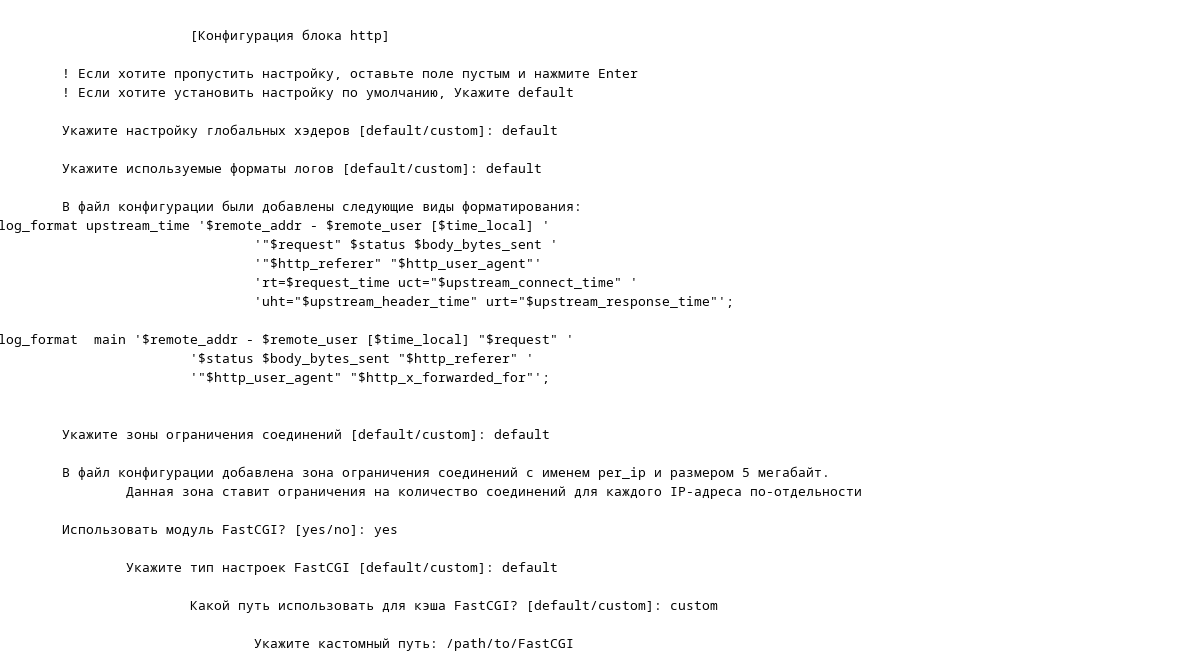
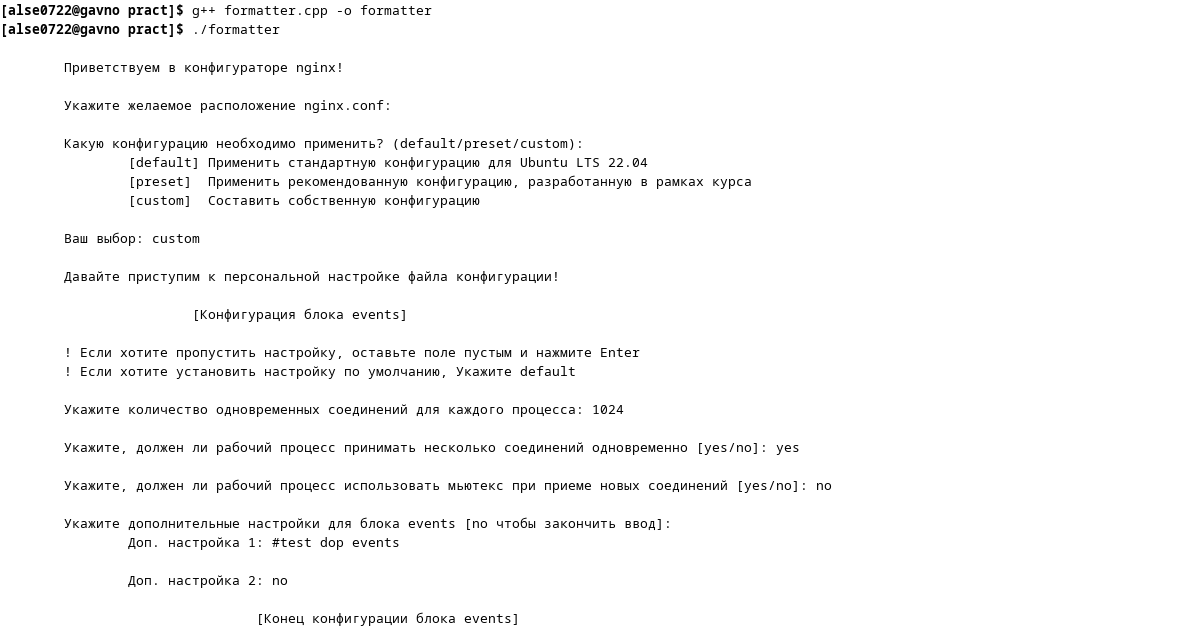
После ее выполнения в консоли появится приветствие, в ходе которого будет необходимо ввести адрес корневой папки веб-сервера и выбрать режим работы:

* default – по указанному адресу будет создан файл конфигурации nginx.conf, содержащий в себе стандартные настройки (версия для Ubuntu LTS 22.04)
* recommended – по указанному адресу будет создан файл конфигурации nginx.conf, содержащий в себе все настройки, описанные выше
* custom – режим конфигурации (заполнение пустого файла конфигурации nginx.conf)

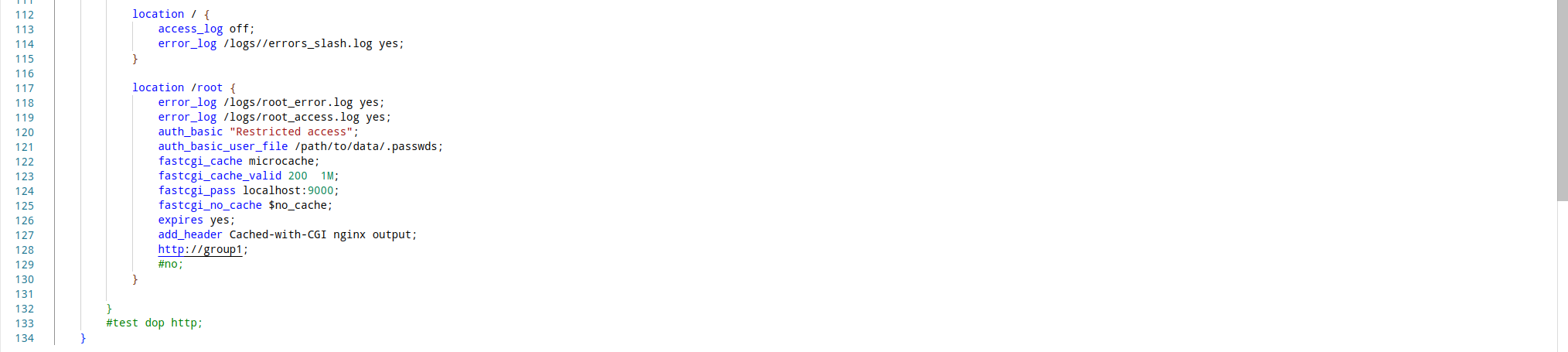
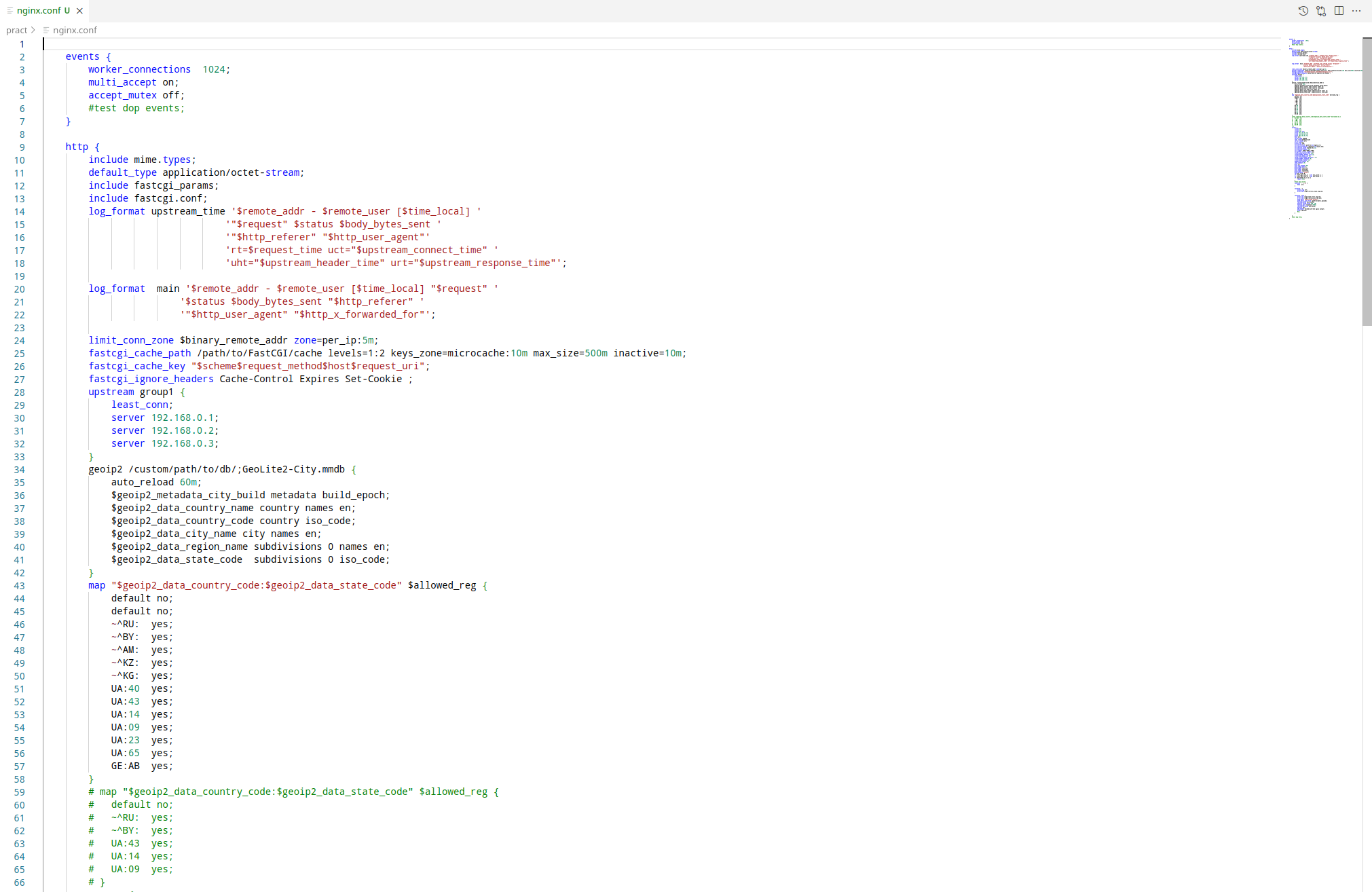
В режиме конфигурации пользователю предоставлена возможность самому прописывать необходимые для его приложения настройки или частично использовать рекоммендованные параметры.

В результате работы программы в режиме конфигурации, будет создан синтаксически верный и рабочий файл конфигурации nginx.conf по адресу, указанному пользователем во время приветствия. Давайте рассмотрим работу данной программы более детально.

# Тестирование программы-конфигуратора formatter



Проверим файл nginx.conf. Очевидно что файл был сгенерирован, можно просмотреть его содержимое в любом доступном текстовом редакторе (в данном случае использовался Visual Code с рисширением Nginx Configuration для наглядности).

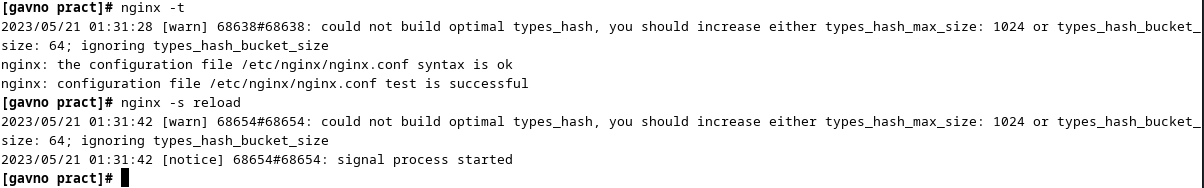


Для дого чтобы проверить правильность синтаксиса файла конфигурации необходимо ввести команду в терминале:

nginx –t

Можно использовать команду бесшовшой реконфигурации сервера, для того чтобы применить настройки файла nginx.conf:

nginx –s reload



На рисунке видно, что синтаксис конфигурационного файла в порядке. Кроме того веб-сервер успешно применил его настройки.

# Заключение

В заключение, данная курсовая работа по теме "Защита веб-сервера Nginx" была направлена на исследование и практическую реализацию методов и мер безопасности, связанных с конфигурацией веб-сервера Nginx.

В рамках практической части работы была разработана программа, способная конфигурировать файл nginx.conf, который является основным файлом конфигурации Nginx. Программа предоставляет пользователю интерактивный интерфейс для выбора типа конфигурации: стандартной, рекомендованной или кастомной. В зависимости от выбора пользователя, программа создает или изменяет файл nginx.conf соответствующим образом.

Полученные результаты позволили продемонстрировать возможности программы в автоматизации процесса конфигурирования Nginx и обеспечении необходимых настроек безопасности. Это может включать установку корректных значений для директив, таких как ограничение подключений, контроль времени ожидания, установка заголовков безопасности, скрытие информации о сервере и других аспектов, которые способствуют общей защите веб-сервера.

Важно отметить, что представленная программа является примером реализации и может быть доработана и расширена в соответствии с потребностями и требованиями конкретного сервера и его окружения. Также следует отметить, что конфигурация веб-сервера Nginx должна рассматриваться как одна из многих мер безопасности, а полная защита веб-сервера требует комплексного подхода, включая обновление программного обеспечения, управление доступом, мониторинг и другие стратегии и механизмы безопасности.

В целом, данная курсовая работа по защите веб-сервера Nginx и разработка программы для конфигурирования nginx.conf позволила изучить и применить практические методы обеспечения безопасности, связанные с одним из самых популярных веб-серверов, что является важной составляющей для обеспечения защиты веб-приложений и данных.

# Список использованных источников

https://habr.com/ru/articles/139931/ -- модуль nginx для защиты от ddos

https://amkolomna.ru/content/zashchita-nginx-ot-ddos-atak -- Защита nginx от DDoS атак

https://rudocs.ispmanager.com/ispmanager-lite/nastrojka-zashchity-ot-ddos-atak -- Настройка защиты от DDoS-атак

https://www.youtube.com/playlist?list=PLhgRAQ8BwWFa7ulOkX0qi5UfVizGD\_-Rc

https://www.youtube.com/watch?v=gSPOI6-ydU4&list=PLhgRAQ8BwWFa7ulOkX0qi5UfVizGD\_-Rc&index=6&ab\_channel=GeekCode

https://onedev.net/post/1025 - Установка Nginx 1.18 на Manjaro

https://www.youtube.com/playlist?list=PLhgRAQ8BwWFa7ulOkX0qi5UfVizGD\_-Rc - уроки по nginx

<https://habr.com/ru/companies/first/articles/683870/>

https://habr.com/ru/articles/428127/ -- Nginx cache: всё новое — хорошо забытое старое

https://nginx.org/ru/docs/http/ngx\_http\_limit\_req\_module.html

https://dev.maxmind.com/geoip/updating-databases?lang=en

https://itsecforu.ru/2018/08/29/как-установить-mod\_geoip-для-apache-в-rhel-и-centos/

https://ixnfo.com/nastroyka-logov-nginx.html

https://habr.com/ru/articles/351904/ -- Простая аутентификация на NGINX с помощью LUA

https://www.nic.ru/help/chto-takoe-nginx-i-kak-pravil6no-ego-nastroit6\_11046.html

ПРИЛОЖЕНИЕ А

## Разработанные настройки для веб-сервера nginx

events {

worker\_connections 1024;

}

http {

include mime.types;

include fastcgi\_params;

include fastcgi.conf;

default\_type application/octet-stream;

add\_header Access-Control-Allow-Origin;

add\_header Caching $upstream\_cache\_status;

add\_header Cache-Control public;

add\_header Vary Accept-Encoding;

add\_header X-Content-**Type**-Options nosniff;

add\_header X-XSS-Protection "1; mode=block";

log\_format upstream\_time '$remote\_addr - $remote\_user [$time\_local] '

'"$request" $status $body\_bytes\_sent '

'"$http\_referer" "$http\_user\_agent"'

'rt=$request\_time uct="$upstream\_connect\_time" uht="$upstream\_header\_time" urt="$upstream\_response\_time"';

log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

'$status $body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

limit\_conn\_zone $server\_name zone=per\_vhost:5m;

limit\_conn\_zone $binary\_remote\_addr zone=per\_ip:5m;

fastcgi\_cache\_path /etc/nginx/cache levels=1:2 keys\_zone=microcache:10m max\_size=500m inactive=10m;

fastcgi\_cache\_key "$scheme$request\_method$host$request\_uri";

fastcgi\_ignore\_headers Cache-Control Expires **Set**-Cookie ;

upstream php\_servers {

least\_conn;

server localhost:10001;

server localhost:10002;

server localhost:10003;

}

geoip2 /usr/share/GeoIP/GeoLite2-City.mmdb {

auto\_reload 60m;

$geoip2\_metadata\_city\_build metadata build\_epoch;

$geoip2\_data\_country\_name country names en;

$geoip2\_data\_country\_code country iso\_code;

$geoip2\_data\_city\_name city names en;

$geoip2\_data\_region\_name subdivisions 0 names en;

$geoip2\_data\_state\_code subdivisions 0 iso\_code;

}

map "$geoip2\_data\_country\_code:$geoip2\_data\_state\_code" $allowed\_reg {

default no;

~^RU: yes;

~^BY: yes;

~^AM: yes;

~^KZ: yes;

~^KG: yes;

UA:40 yes;

UA:43 yes;

UA:14 yes;

UA:09 yes;

UA:23 yes;

UA:65 yes;

GE:AB yes;

}

server {

listen 81;

listen 443 ssl;

allow 192.168.0.174;

deny all;

server\_name localhost;

root /etc/nginx/site/gaming/;

error\_log off;

access\_log off;

*#access\_log logs/host.access\_base.log main;*

*#error\_log logs/host.error\_base.log main;*

ssl\_certificate /etc/nginx/ssl/nginx.crt;

ssl\_certificate\_key /etc/nginx/ssl/nginx.key;

ssl\_session\_cache shared:SSL:1m;

ssl\_session\_timeout 5m;

ssl\_ciphers HIGH:!aNULL:!MD5;

ssl\_prefer\_server\_ciphers on;

client\_body\_buffer\_size 16k;

client\_header\_buffer\_size 1k;

client\_max\_body\_size 8m;

large\_client\_header\_buffers 2 1k;

client\_body\_timeout 12;

client\_header\_timeout 12;

keepalive\_timeout 65;

send\_timeout 10;

server\_tokens off;

gzip on;

gzip\_min\_length 100;

gzip\_comp\_level 3;

gzip\_types text/plain;

gzip\_types text/css;

gzip\_types text/javascript;

gzip\_disable "msie6";

set $no\_cache 0;

**if** ($request\_method = POST) { set $no\_cache 1; }

**if** ($query\_string != "") { set $no\_cache 1; }

**if** ($request\_uri ~\* "/profile") { set $no\_cache 1; }

**if** ($allowed\_reg = no) {

**return** 444;

}

location /test\_auth {

auth\_basic "Restricted access";

auth\_basic\_user\_file /etc/nginx/creds/.htpasswd;

access\_log /var/log/nginx/new.log upstream\_time;

}

location ~\* \.(css|js|jpg|png|gif)$ {

fastcgi\_cache microcache;

fastcgi\_cache\_valid 200 60m;

expires 1M;

access\_log off;

error\_log off;

limit\_conn per\_ip 1;

}

location /test\_balancer {

fastcgi\_cache microcache;

fastcgi\_cache\_valid 200 60m;

fastcgi\_pass 127.0.0.1:9000;

fastcgi\_no\_cache $no\_cache;

proxy\_set\_header proxy\_header\_to\_server nginx;

add\_header proxy\_header nginx;

proxy\_pass http://php\_servers;

}

location ~ /\.ht {

deny all;

}

}

}

ПРИЛОЖЕНИЕ Б

Код программы-конфигуратора formatter

*#include <iostream>*

*#include <cstdio>*

*#include <fstream>*

*#include <istream>*

*#include <vector>*

*#include <regex>*

*#include <string>*

*#include <iterator>*

using namespace std;

type**def** **vector**<string> vs;

type**def** **string** ss;

struct events\_block

{

ss start;

ss tab;

vs workers;

vs multi;

vs mutex;

vs custom;

ss end;

};

struct http\_block

{

ss start;

ss tab;

vs include;

vs global\_headers;

vs log\_formatting;

vs limit\_concurrency;

vs fast\_cgi;

vs load\_balancer;

vs geoip;

vs geoip\_blocks\_allow;

vs geoip\_blocks\_deny;

vs custom;

ss end;

};

struct server\_block

{

ss start;

ss tab;

vs listen;

vs allow;

vs deny;

ss server\_name;

ss root\_folder;

vs global\_logs\_status;

vs global\_logs\_settings;

vs ssl;

vs buffers;

vs timeouts;

vs keepalive;

ss server\_token;

vs gzip;

vs caching;

vs custom;

ss end;

};

struct location\_block

{

ss start;

ss tab;

ss location\_name;

vs auth;

vs fast\_cgi;

ss expires;

vs custom\_logs\_status;

vs custom\_logs\_settings;

vs custom\_headers;

ss proxy\_pass;

vs custom;

ss end;

};

struct nginx\_conf

{

events\_block events;

http\_block http;

server\_block server;

vector<location\_block> locations;

ss location;

};

nginx\_conf config;

vs readInputValues()

{

vs values;

string input;

getline(cin, input);

istringstream iss(input);

string value;

**while** (iss >> value)

{

values.push\_back(value);

}

**return** values;

}

void setEventsBlock()

{

ss workers, multi, mutex, custom;

int cnt(1);

config.events.start = "\n\tevents {";

config.events.tab = "\n\t\t";

config.events.end = "\n\t}\n";

cout << "\n\t\n\t\t\t[Конфигурация блока events]\n";

cout << "\n\t! Если хотите пропустить настройку, оставьте поле пустым и нажмите Enter";

cout << "\n\t! Если хотите установить настройку по умолчанию, Укажите default\n";

cout << "\n\tУкажите количество одновременных соединений для каждого процесса: ";

getline(cin, workers);

**if** (workers != "")

**if** (workers != "default")

config.events.workers.push\_back("worker\_connections\t" + workers + ";");

**else**

config.events.workers.push\_back("worker\_connections 1024;");

cout << "\n\tУкажите, должен ли рабочий процесс принимать несколько соединений одновременно [yes/no]: ";

getline(cin, multi);

**if** (multi != "")

{

**if** (multi == "default")

config.events.multi.push\_back("multi\_accept on;");

**if** (multi == "yes")

config.events.multi.push\_back("multi\_accept on;");

**if** (multi == "no")

config.events.multi.push\_back("multi\_accept off;");

}

cout << "\n\tУкажите, должен ли рабочий процесс использовать мьютекс при приеме новых соединений [yes/no]: ";

getline(cin, mutex);

**if** (mutex != "")

{

**if** (mutex == "default")

config.events.mutex.push\_back("accept\_mutex on;");

**if** (mutex == "yes")

config.events.mutex.push\_back("accept\_mutex on;");

**if** (mutex == "no")

config.events.mutex.push\_back("accept\_mutex off;");

}

cout << "\n\tУкажите дополнительные настройки для блока events [no чтобы закончить ввод]: ";

cout << "\n\t\tДоп. настройка " << cnt << ": ";

getline(cin, custom);

**while** (custom != "no")

{

config.events.custom.push\_back(custom + (custom.back() == ';' ? "" : ";"));

cnt++;

cout << "\n\t\tДоп. настройка " << cnt << ": ";

getline(cin, custom);

}

cout << "\n\t\t\t\t[Конец конфигурации блока events]\n";

}

void setHttpBlock()

{

config.http.start = "\n\thttp {";

config.http.tab = "\n\t\t";

config.http.end = "\n\t}";

config.http.include.push\_back("include mime.types;");

config.http.include.push\_back("default\_type application/octet-stream;");

vs servers, regions, selected\_regions;

ss header\_st, gl\_header\_name, gl\_header\_data,

log\_st, log\_format\_name, log\_format\_data,

limc\_st, lim\_conn\_param, lim\_conn\_name, lim\_conn\_size,

fcgi\_st, fcgi\_path, fcgi\_custom,

ups\_st, ups\_name, ups\_type, ups\_server,

geo\_st, geo\_path, geobl\_regs,

custom;

ss upstream\_time\_format = R"(log\_format upstream\_time '$remote\_addr - $remote\_user [$time\_local] '

'"$request" $status $body\_bytes\_sent '

'"$http\_referer" "$http\_user\_agent"'

'rt=$request\_time uct="$upstream\_connect\_time" '

'uht="$upstream\_header\_time" urt="$upstream\_response\_time"';

)";

ss main\_format = R"(log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

'$status $body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

)";

regions = {

"default no;",

"~^RU: yes;",

"~^BY: yes;",

"~^AM: yes;",

"~^KZ: yes;",

"~^KG: yes;",

"UA:40 yes;",

"UA:43 yes;",

"UA:14 yes;",

"UA:09 yes;",

"UA:23 yes;",

"UA:65 yes;",

"GE:AB yes;"};

int cnt(1);

cout << "\n\t\n\t\t\t[Конфигурация блока http]\n";

cout << "\n\t! Если хотите пропустить настройку, оставьте поле пустым и нажмите Enter";

cout << "\n\t! Если хотите установить настройку по умолчанию, Укажите default\n";

cout << "\n\tУкажите настройку глобальных хэдеров [default/custom]: ";

getline(cin, gl\_header\_name);

**if** (header\_st != "no")

{

**if** (header\_st == "default")

{

config.http.global\_headers.push\_back(R"(add\_header Access-Control-Allow-Origin;)");

config.http.global\_headers.push\_back(R"(add\_header Caching $upstream\_cache\_status;)");

config.http.global\_headers.push\_back(R"(add\_header Cache-Control public;)");

config.http.global\_headers.push\_back(R"(add\_header Vary Accept-Encoding;)");

config.http.global\_headers.push\_back(R"(add\_header X-Content-Type-Options nosniff;)");

config.http.global\_headers.push\_back(R"(add\_header X-XSS-Protection "1; mode=block";)");

}

**if** (header\_st == "custom")

{

cout << "\n\tУкажите хэдер " << cnt << ": ";

cout << "\n\t\tИмя: ";

getline(cin, gl\_header\_name);

**while** (gl\_header\_name != "no")

{

cout << "\n\t\tДанные: ";

getline(cin, gl\_header\_data);

config.http.global\_headers.push\_back(

"add\_header " + gl\_header\_name + " " + gl\_header\_data + ";");

cnt++;

cout << "\n\tУкажите хэдер " << cnt << ": ";

cout << "\n\t\tИмя: ";

getline(cin, gl\_header\_name);

}

}

}

cout << "\n\tУкажите используемые форматы логов [default/custom]: ";

getline(cin, log\_st);

**if** (log\_st != "")

{

**if** (log\_st == "default")

{

config.http.log\_formatting.push\_back(upstream\_time\_format);

config.http.log\_formatting.push\_back(main\_format);

cout << "\n\tВ файл конфигурации были добавлены следующие виды форматирования:\n";

cout << upstream\_time\_format << endl;

cout << main\_format << endl;

}

**if** (log\_st == "custom")

{

cnt = 1;

config.http.log\_formatting.push\_back(upstream\_time\_format);

config.http.log\_formatting.push\_back(main\_format);

cout << "\n\tВ файл конфигурации были добавлены следующие виды форматирования:\n";

cout << upstream\_time\_format << endl;

cout << main\_format << endl;

cout << "\n\tНовый формат " << cnt << ": ";

cout << "\n\t\tИмя: ";

getline(cin, log\_format\_name);

**while** (gl\_header\_name != "no")

{

cout << "\n\t\tДанные: ";

getline(cin, log\_format\_data);

config.http.log\_formatting.push\_back("log\_format " + log\_format\_name + " " + log\_format\_data + ";");

cnt++;

cout << "\n\tНовый формат" << cnt << ": ";

cout << "\n\t\tИмя: ";

getline(cin, gl\_header\_name);

}

}

}

cout << "\n\tУкажите зоны ограничения соединений [default/custom]: ";

getline(cin, limc\_st);

**if** (limc\_st != "")

{

**if** (limc\_st == "default")

{

config.http.limit\_concurrency.push\_back(

"limit\_conn\_zone $binary\_remote\_addr zone=per\_ip:5m;");

cout << "\n\tВ файл конфигурации добавлена зона ограничения соединений с именем per\_ip и размером 5 мегабайт.";

cout << "\n\t\tДанная зона ставит ограничения на количество соединений для каждого IP-адреса по-отдельности\n";

}

**if** (limc\_st == "custom")

{

cnt = 1;

cout << "\n\tУкажите зону ограничения соединений " << cnt << " [no чтобы закончить ввод]: ";

cout << "\n\t\tИмя: ";

getline(cin, lim\_conn\_name);

cout << "\n\t\tПеременная ограничения: ";

getline(cin, lim\_conn\_param);

cout << "\n\t\tРазмер зоны: ";

getline(cin, lim\_conn\_size);

**while** (gl\_header\_name != "no")

{

config.http.limit\_concurrency.push\_back(

"limit\_conn\_zone" + lim\_conn\_param + " zone=" + lim\_conn\_name + ":" + lim\_conn\_size + ";");

cnt++;

cout << "\n\tУкажите зону ограничения соединений " << cnt << " [no чтобы закончить ввод]: ";

cout << "\n\t\tИмя: ";

getline(cin, lim\_conn\_name);

cout << "\n\t\tПеременная ограничения: ";

getline(cin, lim\_conn\_param);

cout << "\n\t\tРазмер зоны: ";

getline(cin, lim\_conn\_size);

}

}

}

cout << "\n\tИспользовать модуль FastCGI? [yes/no]: ";

getline(cin, fcgi\_st);

**if** (fcgi\_st == "yes")

{

config.http.include.push\_back("include fastcgi\_params;");

config.http.include.push\_back("include fastcgi.conf;");

cout << "\n\t\tУкажите тип настроек FastCGI [default/custom]: ";

getline(cin, fcgi\_st);

**if** (fcgi\_st == "default")

{

cout << "\n\t\t\tКакой путь использовать для кэша FastCGI? [default/custom]: ";

getline(cin, fcgi\_path);

**if** (fcgi\_path == "default" || fcgi\_path == "")

{

config.http.fast\_cgi.push\_back(

"fastcgi\_cache\_path " +

config.location + "/cache levels=1:2 keys\_zone=microcache:10m max\_size=500m inactive=10m;");

config.http.fast\_cgi.push\_back(

R"(fastcgi\_cache\_key "$scheme$request\_method$host$request\_uri";)");

config.http.fast\_cgi.push\_back(

"fastcgi\_ignore\_headers Cache-Control Expires Set-Cookie ;");

}

**else**

{

cout << "\n\t\t\t\tУкажите кастомный путь: ";

getline(cin, fcgi\_path);

config.http.fast\_cgi.push\_back(

"fastcgi\_cache\_path " +

fcgi\_path + "/cache levels=1:2 keys\_zone=microcache:10m max\_size=500m inactive=10m;");

config.http.fast\_cgi.push\_back(

R"(fastcgi\_cache\_key "$scheme$request\_method$host$request\_uri";)");

config.http.fast\_cgi.push\_back(

"fastcgi\_ignore\_headers Cache-Control Expires Set-Cookie ;");

}

}

**if** (fcgi\_st == "custom")

{

cnt = 1;

cout << "\n\tУкажите настройку FastCGI " << cnt << " [no чтобы закончить ввод]: ";

getline(cin, fcgi\_custom);

**while** (fcgi\_custom != "no")

{

config.http.fast\_cgi.push\_back(

fcgi\_custom + (fcgi\_custom.back() == ';' ? "" : ";"));

cnt++;

cout << "\n\tУкажите настройку FastCGI " << cnt << " [no чтобы закончить ввод]: ";

getline(cin, fcgi\_custom);

}

}

}

cout << "\n\tСоздать группу серверов для балансировщика нагрузки? [yes/no]: ";

getline(cin, ups\_st);

**if** (ups\_st == "yes")

{

cout << "\n\t\tУкажите имя группы серверов: ";

getline(cin, ups\_name);

cout << "\n\t\tУкажите алгоритм балансировки: ";

getline(cin, ups\_type);

cout << "\n\t\tУкажите сервера группы (ip:port) [no чтобы закончить ввод]: ";

cnt = 1;

cout << "\n\t\t\tServer " << cnt << ": ";

getline(cin, ups\_server);

**while** (ups\_server != "no" &amp;&amp; ups\_server != "")

{

cnt++;

servers.push\_back(ups\_server);

cout << "\n\t\t\tServer " << cnt << ": ";

getline(cin, ups\_server);

}

config.http.load\_balancer.push\_back(

"upstream " + ups\_name + " {");

config.http.load\_balancer.push\_back(

"\t" + ups\_type + ";");

**for** (auto server : servers)

config.http.load\_balancer.push\_back(

"\tserver " + server + ";");

config.http.load\_balancer.push\_back(

"}");

}

cout << "\n\tИспользовать модуль Geoip2? [yes/no]: ";

getline(cin, geo\_st);

**if** (geo\_st == "yes")

{

cout << "\n\tКакой путь использовать для доступа к GeoLite2-City.mmdb? [default/custom]: ";

getline(cin, geo\_path);

**if** (geo\_path != "")

{

**if** (geo\_path == "default")

config.http.geoip.push\_back(

"geoip2 /usr/share/GeoIP/GeoLite2-City.mmdb {");

**if** (geo\_path != "default")

{

cout << "\n\tВведите путь до GeoLite2-City.mmdb: ";

getline(cin, geo\_path);

config.http.geoip.push\_back(

"geoip2 " + geo\_path + (geo\_path.back() == ';' ? "" : ";") + "GeoLite2-City.mmdb {");

}

config.http.geoip.push\_back("\tauto\_reload 60m;");

config.http.geoip.push\_back("\t$geoip2\_metadata\_city\_build metadata build\_epoch;");

config.http.geoip.push\_back("\t$geoip2\_data\_country\_name country names en;");

config.http.geoip.push\_back("\t$geoip2\_data\_country\_code country iso\_code;");

config.http.geoip.push\_back("\t$geoip2\_data\_city\_name city names en;");

config.http.geoip.push\_back("\t$geoip2\_data\_region\_name subdivisions 0 names en;");

config.http.geoip.push\_back("\t$geoip2\_data\_state\_code subdivisions 0 iso\_code;");

config.http.geoip.push\_back("}");

}

cout << "\n\tТекущие разрешенные регионы: ";

cout << "\n\t\t[1] RU --> Россия";

cout << "\n\t\t[2] BY --> Белоруссия";

cout << "\n\t\t[3] AM --> Армения";

cout << "\n\t\t[4] KZ --> Казахстан";

cout << "\n\t\t[5] KG --> Кыргызстан";

cout << "\n\t\t[6] UA:4 --> Севастополь";

cout << "\n\t\t[7] UA:4 --> Крым";

cout << "\n\t\t[8] UA:1 --> Донецкая область";

cout << "\n\t\t[9] UA:0 --> Луганская область";

cout << "\n\t\t[10] UA:2 --> Запорожская область";

cout << "\n\t\t[11] UA:6 --> Херсонская область";

cout << "\n\t\t[12] GE:A --> Абхазия";

cout << "\n\tУкажите id регионoв, которые будут иметь доступ при внештатных ситуциях [no чтобы закончить ввод]: ";

getline(cin, geobl\_regs);

string delimiter = " ";

size\_t pos = 0;

string token;

**while** ((pos = geobl\_regs.find(delimiter)) != string::npos)

{

token = geobl\_regs.substr(0, pos);

int index = stoi(token);

**if** (index >= 0 &amp;&amp; index < regions.size())

selected\_regions.push\_back(regions[index]);

geobl\_regs.erase(0, pos + delimiter.length());

}

**if** (!geobl\_regs.empty())

{

int index = stoi(geobl\_regs);

**if** (index >= 0 &amp;&amp; index < regions.size())

selected\_regions.push\_back(regions[index]);

}

config.http.geoip\_blocks\_allow.push\_back(

R"(map "$geoip2\_data\_country\_code:$geoip2\_data\_state\_code" $allowed\_reg {)");

config.http.geoip\_blocks\_allow.push\_back(

"\tdefault no;");

**for** (auto reg : regions)

config.http.geoip\_blocks\_allow.push\_back(

"\t" + reg);

config.http.geoip\_blocks\_allow.push\_back(

"}");

config.http.geoip\_blocks\_deny.push\_back(

R"(map "$geoip2\_data\_country\_code:$geoip2\_data\_state\_code" $allowed\_reg {)");

config.http.geoip\_blocks\_deny.push\_back(

"\tdefault no;");

**for** (auto reg : selected\_regions)

config.http.geoip\_blocks\_deny.push\_back(

"\t" + reg);

config.http.geoip\_blocks\_deny.push\_back(

"}");

}

cnt = 1;

cout << "\n\tУкажите дополнительные настройки для блока http [no чтобы закончить ввод]: ";

cout << "\n\t\tДоп. настройка " << cnt << ": ";

getline(cin, custom);

**while** (custom != "no")

{

config.http.custom.push\_back(custom + (custom.back() == ';' ? "" : ";"));

cnt++;

cout << "\n\t\tДоп. настройка " << cnt << ": ";

getline(cin, custom);

}

cout << "\n\t\t\t\t[Конец конфигурации блока http]\n";

}

void setServerBlock()

{

config.server.start = "\n\t\tserver {";

config.server.tab = "\n\t\t\t";

config.server.end = "\n\t\t}";

int cnt(1);

vs listening\_ports, allow\_hosts, deny\_hosts, gzip\_files, gzip\_disable;

ss custom, server\_name, root\_dir,

alog\_st, elog\_st, alog\_path, elog\_path, alog\_type, elog\_type,

ssl\_st, ssl\_crt, ssl\_key, ssl\_listen,

buf\_st, tmout\_st, kpalive\_st, ver\_st,

gzip\_st, cache\_st;

cout << "\n\t\n\t\t\t[Конфигурация блока server]\n";

cout << "\n\t! Если хотите пропустить настройку, оставьте поле пустым и нажмите Enter";

cout << "\n\t! Если хотите установить настройку по умолчанию, Укажите default\n";

cout << "\n\tУкажите прослушиваемые порты [через пробел]: ";

listening\_ports = readInputValues();

**for** (auto port : listening\_ports)

config.server.listen.push\_back("listen " + port + ";");

cout << "\n\tУкажите разрешенные хосты [через пробел]: ";

allow\_hosts = readInputValues();

**for** (auto host : allow\_hosts)

config.server.allow.push\_back("allow " + host + ";");

cout << "\n\tУкажите запрещенные хосты [через пробел]: ";

deny\_hosts = readInputValues();

**for** (auto host : deny\_hosts)

config.server.deny.push\_back("deny " + host + ";");

cout << "\n\tУкажите имя сервера [default]: ";

getline(cin, server\_name);

**if** (server\_name == "default")

cout << "\n\tИмя сервера установлено как: localhost";

config.server.server\_name = config.server.tab + "server\_name ";

config.server.server\_name += server\_name == "" ? "localhost" : server\_name;

cout << "\n\tУкажите корневой каталог сайта [default]: ";

**if** (root\_dir == "default")

cout << "\n\tКорневой каталог сайта установлен как: /etc/nginx/site/";

getline(cin, root\_dir);

config.server.root\_folder = config.server.tab + "root ";

config.server.root\_folder += root\_dir == "" ? "/etc/nginx/site/" : root\_dir;

cout << "\n\tВключить логирование ошибок для всех директорий? [yes/no]: ";

getline(cin, elog\_st);

**if** (elog\_st != "")

{

**if** (elog\_st == "yes")

{

config.server.global\_logs\_status.push\_back("error\_log on;");

cout << "\n\t\tУкажите каталог для хранения логов ошибок [default]: ";

getline(cin, elog\_path);

elog\_path = elog\_path == "default" ? "/etc/nginx/logs/" : elog\_path;

cout << "\n\t\tУкажите формат логов ошибок [default]: ";

getline(cin, elog\_type);

elog\_type = elog\_type == "default" ? "main" : elog\_type;

config.server.global\_logs\_settings.push\_back(

"error\_log " + elog\_path + (elog\_path.back() == '/' ? "" : "/") +

"error\_base.log " + elog\_type + ";");

}

**if** (elog\_st == "no")

config.server.global\_logs\_status.push\_back("error\_log off;");

}

cout << "\n\tВключить логирование доступа для всех директорий? [yes/no]: ";

getline(cin, alog\_st);

**if** (alog\_st != "")

{

**if** (alog\_st == "yes")

{

config.server.global\_logs\_status.push\_back("access\_log on;");

cout << "\n\t\tУкажите каталог для хранения логов ошибок [default]: ";

getline(cin, alog\_path);

alog\_path = alog\_path == "default" ? "/etc/nginx/logs/" : alog\_path;

cout << "\n\t\tУкажите формат логов ошибок [default]: ";

getline(cin, alog\_type);

alog\_type = alog\_type == "default" ? "main" : alog\_type;

config.server.global\_logs\_settings.push\_back(

"access\_log " + alog\_path + (alog\_path.back() == '/' ? "" : "/") +

"access\_base.log " + alog\_type + ";");

}

**if** (alog\_st == "no")

config.server.global\_logs\_status.push\_back("access\_log off;");

}

cout << "\n\tВключить шифрование SSL/TLS? [yes/no]: ";

getline(cin, ssl\_st);

**if** (ssl\_st == "yes")

{

cout << "\n\t\tУкажите прослушивающий порт: ";

getline(cin, ssl\_listen);

cout << "\n\t\tУкажите файл ssl-сертификата (.crt): ";

getline(cin, ssl\_crt);

cout << "\n\t\tУкажите файл ssl-ключа (.key): ";

getline(cin, ssl\_key);

config.server.listen.push\_back("listen " + ssl\_listen + " ssl;");

config.server.ssl.push\_back("ssl\_certificate " + ssl\_crt + ";");

config.server.ssl.push\_back("ssl\_certificate\_key " + ssl\_key + ";");

config.server.ssl.push\_back("ssl\_session\_cache shared:SSL:1m;");

config.server.ssl.push\_back("ssl\_session\_timeout 5m;");

config.server.ssl.push\_back("ssl\_ciphers HIGH:!aNULL:!MD5;");

config.server.ssl.push\_back("ssl\_prefer\_server\_ciphers on;");

}

cout << "\n\tВключить ограничения буферизации сообщений? [yes/no]: ";

getline(cin, buf\_st);

**if** (ssl\_st == "yes")

{

config.server.buffers.push\_back("client\_body\_buffer\_size 16k;");

config.server.buffers.push\_back("client\_header\_buffer\_size 1k;");

config.server.buffers.push\_back("client\_max\_body\_size 8m;");

config.server.buffers.push\_back("large\_client\_header\_buffers 2 1k;");

}

cout << "\n\tВключить таймаут ожидания для передачи запроса клиента? [yes/no]: ";

getline(cin, tmout\_st);

**if** (ssl\_st == "yes")

{

cout << "\n\t\tУкажите таймаут для хэдера запроса: ";

getline(cin, tmout\_st);

config.server.timeouts.push\_back("client\_header\_timeout " + tmout\_st + ";");

cout << "\n\t\tУкажите таймаут для тела запроса: ";

getline(cin, tmout\_st);

config.server.timeouts.push\_back("client\_body\_timeout " + tmout\_st + ";");

}

cout << "\n\tВключить таймаут ожидания для активного соединения? [yes/no]: ";

getline(cin, kpalive\_st);

**if** (kpalive\_st == "yes")

config.server.keepalive.push\_back("keepalive\_timeout 65;");

cout << "\n\tВключить таймаут ожидания для отправки данных клиенту? [yes/no]: ";

getline(cin, kpalive\_st);

**if** (kpalive\_st == "yes")

config.server.keepalive.push\_back("send\_timeout 10;");

cout << "\n\tОтключить отображение информации о версии Nginx в хэдерах? [yes/no]: ";

getline(cin, ver\_st);

**if** (ver\_st == "yes")

config.server.server\_token = config.server.tab + "server\_tokens off;";

cout << "\n\tВключить сжатие ответов сервера с помощью gzip? [yes/no]: ";

getline(cin, gzip\_st);

**if** (gzip\_st == "yes")

{

config.server.gzip.push\_back("gzip on;");

cout << "\n\t\tУкажите минимальный размер сжимаемых файлов: ";

getline(cin, gzip\_st);

config.server.gzip.push\_back("gzip\_min\_length " + gzip\_st + ";");

cout << "\n\t\tУкажите степень сжатия файлов: ";

getline(cin, gzip\_st);

config.server.gzip.push\_back("gzip\_comp\_level " + gzip\_st + ";");

cout << "\n\t\tУкажите типы сжимаемых файлов через пробел(Пр.: css plain): ";

gzip\_files = readInputValues();

**for** (auto file : gzip\_files)

config.server.gzip.push\_back("gzip\_types text/" + file + ";");

config.server.gzip.push\_back(R"(gzip\_disable "msie6";)");

}

cout << "\n\tОграничитить кэширование FastCGI по определенным правилам? [yes/no]: ";

getline(cin, cache\_st);

**if** (cache\_st == "yes")

{

config.server.caching.push\_back("set $no\_cache 0;");

cout << "\n\tУкажите правила ограничений [no чтобы закончить ввод]: ";

cnt = 1;

cout << "\n\t\tПравило " << cnt << ": ";

getline(cin, custom);

**while** (custom != "no")

{

config.server.caching.push\_back("if (" + custom + ") { set $no\_cache 1; }");

cnt++;

cout << "\n\t\tПравило " << cnt << ": ";

getline(cin, custom);

}

}

**if** (config.http.geoip.size() != 0)

{

config.server.custom.push\_back("if ($allowed\_reg = no) {");

config.server.custom.push\_back("\treturn 444;");

config.server.custom.push\_back("}");

}

cnt = 1;

cout << "\n\tУкажите дополнительные настройки для блока server [no чтобы закончить ввод]: ";

cout << "\n\t\tДоп. настройка " << cnt << ": ";

getline(cin, custom);

**while** (custom != "no")

{

config.server.custom.push\_back(custom + (custom.back() == ';' ? "" : ";"));

cnt++;

cout << "\n\t\tДоп. настройка " << cnt << ": ";

getline(cin, custom);

}

cout << "\n\t\t\t\t[Конец конфигурации блока server]\n";

}

void setLocations()

{

int cnt(1);

ss loc\_st, exp\_st, prx\_st, prx\_name, bal\_st,

auth\_st, auth\_path, auth\_log, cgi\_st,

celog\_st, celog\_path, celog\_type, celog\_name,

prxhc\_st, prxhs\_st, custom, hd\_name, hd\_status;

location\_block default\_loc;

cout << "\n\t\n\t\t\t[Конфигурация блоков location]\n";

cout << "\n\t! Если хотите пропустить настройку, оставьте поле пустым и нажмите Enter";

cout << "\n\t! Если хотите установить настройку по умолчанию, Укажите default\n";

default\_loc.location\_name = R"( ~ /\.ht )";

default\_loc.start = "\n\t\t\tlocation " + default\_loc.location\_name + "{";

default\_loc.tab = "\n\t\t\t\t";

default\_loc.end = "\n\t\t\t}\n";

default\_loc.custom.push\_back("deny all;");

config.locations.push\_back(default\_loc);

cout << "\n\tСоздать новую локацию? [yes/no]: ";

getline(cin, loc\_st);

**while** (loc\_st == "yes")

{

location\_block nloc;

cout << "\n\t\tУкажите имя локации: ";

getline(cin, nloc.location\_name);

nloc.start = "\n\t\t\tlocation " + nloc.location\_name + " {";

nloc.tab = "\n\t\t\t\t";

nloc.end = "\n\t\t\t}\n";

cout << "\n\tВключить логирование ошибок для данной директории? [yes/no]: ";

getline(cin, celog\_st);

**if** (celog\_st != "")

{

**if** (celog\_st == "yes")

{

// nloc.custom\_logs\_status.push\_back("error\_log on;");

cout << "\n\t\tУкажите каталог для хранения логов ошибок [default]: ";

getline(cin, celog\_path);

celog\_path = celog\_path == "default" ? "/etc/nginx/logs/" : celog\_path;

cout << "\n\t\tУкажите имя логов: ";

getline(cin, celog\_name);

cout << "\n\t\tУкажите формат логов ошибок [default]: ";

getline(cin, celog\_type);

celog\_type = celog\_type == "default" ? "main" : celog\_type;

nloc.custom\_logs\_settings.push\_back(

"error\_log " + celog\_path + (celog\_st.back() == '/' ? "" : "/") +

celog\_name + " " + celog\_st + ";");

}

**if** (celog\_st == "no")

nloc.custom\_logs\_status.push\_back("error\_log off;");

}

cout << "\n\tВключить логирование доступа для данной директории? [yes/no]: ";

getline(cin, celog\_st);

**if** (celog\_st != "")

{

**if** (celog\_st == "yes")

{

// nloc.custom\_logs\_status.push\_back("access\_log on;");

cout << "\n\t\tУкажите каталог для хранения логов ошибок [default]: ";

getline(cin, celog\_path);

celog\_path = celog\_path == "default" ? "/etc/nginx/logs/" : celog\_path;

cout << "\n\t\tУкажите имя логов: ";

getline(cin, celog\_name);

cout << "\n\t\tУкажите формат логов ошибок [default]: ";

getline(cin, celog\_type);

celog\_type = celog\_type == "default" ? "main" : celog\_type;

nloc.custom\_logs\_settings.push\_back(

"error\_log " + celog\_path + (celog\_st.back() == '/' ? "" : "/") +

celog\_name + " " + celog\_st + ";");

}

**if** (celog\_st == "no")

nloc.custom\_logs\_status.push\_back("access\_log off;");

}

cout << "\n\tВключить базовую аутентификацию? [yes/no]: ";

getline(cin, auth\_st);

**if** (auth\_st == "yes")

{

cout << "\n\t\t\tУкажите файл с учетными данными: ";

getline(cin, auth\_path);

nloc.auth.push\_back(R"(auth\_basic "Restricted access";)");

nloc.auth.push\_back("auth\_basic\_user\_file " + auth\_path + ";");

}

cout << "\n\tВключить кэширование на стороне клиента? [yes/no]: ";

getline(cin, exp\_st);

**if** (auth\_st == "yes")

{

cout << "\n\t\tУкажите время валидности кэша: ";

getline(cin, exp\_st);

nloc.expires = "expires " + auth\_st + ";";

}

**if** (config.http.fast\_cgi.size() > 0)

{

cout << "\n\tВключить кэширование FastCGI? [yes/no]: ";

getline(cin, cgi\_st);

**if** (auth\_st == "yes")

{

cout << "\n\t\tУкажите зону ключей кэша FastCGI: ";

getline(cin, cgi\_st);

nloc.fast\_cgi.push\_back("fastcgi\_cache " + cgi\_st + ";");

cout << "\n\t\tУкажите время валидности кэша FastCGI: ";

getline(cin, cgi\_st);

nloc.fast\_cgi.push\_back("fastcgi\_cache\_valid 200 " + cgi\_st + ";");

cout << "\n\t\tУкажите прокси сервер FastCGI: ";

getline(cin, cgi\_st);

**if** (cgi\_st != "" &amp;&amp; cgi\_st != "no")

nloc.fast\_cgi.push\_back("fastcgi\_pass " + cgi\_st + ";");

cout << "\n\t\tИспользовать дополнительные правила кэширования FastCGI? [yes/no]: ";

getline(cin, cgi\_st);

**if** (cgi\_st == "yes")

nloc.fast\_cgi.push\_back("fastcgi\_no\_cache $no\_cache;");

}

}

cout << "\n\tДобавить кастомные хэдеры? [yes/no]: ";

getline(cin, prxhc\_st);

**if** (prxhc\_st == "yes")

{

cnt = 1;

cout << "\n\t\tУкажите хэдер " << cnt << ": ";

cout << "\n\t\t\tИмя: ";

getline(cin, hd\_name);

**while** (hd\_name != "no")

{

cout << "\n\t\t\tДанные: ";

getline(cin, hd\_status);

nloc.custom\_headers.push\_back(

"add\_header " + hd\_name + " " + hd\_status + ";");

cnt++;

cout << "\n\t\tУкажите хэдер " << cnt << ": ";

cout << "\n\t\t\tИмя: ";

getline(cin, hd\_name);

}

}

**if** (config.http.load\_balancer.size() > 0)

{

cout << "\n\tИспользовать балансировщик? [yes/no]: ";

getline(cin, bal\_st);

**if** (bal\_st == "yes")

{

cout << "\n\t\tУкажите имя группы серверов: ";

getline(cin, bal\_st);

nloc.proxy\_pass = "http://" + bal\_st + ";";

}

}

cnt = 1;

cout << "\n\tУкажите дополнительные настройки для блока server [no чтобы закончить ввод]: ";

cout << "\n\t\tДоп. настройка " << cnt << ": ";

getline(cin, custom);

**while** (custom != "no")

{

nloc.custom.push\_back(custom + (custom.back() == ';' ? "" : ";"));

cnt++;

cout << "\n\t\tДоп. настройка " << cnt << ": ";

getline(cin, custom);

}

config.locations.push\_back(nloc);

cout << "\n\tСоздать новую локацию? [yes/no]: ";

getline(cin, loc\_st);

}

cout << "\n\t\t\t\t[Конец конфигурации блоков locations]\n";

}

void pushAllBlocks()

{

ofstream config\_file(config.location + (config.location.back() == '/' ? "" : "/") + "nginx.conf");

config\_file << config.events.start;

**for** (auto params : config.events.workers)

**if** (params != "")

config\_file << config.events.tab + params;

**for** (auto params : config.events.multi)

**if** (params != "")

config\_file << config.events.tab + params;

**for** (auto params : config.events.mutex)

**if** (params != "")

config\_file << config.events.tab + params;

**for** (auto params : config.events.custom)

**if** (params != "")

config\_file << config.events.tab + params;

config\_file << config.events.end;

config\_file << config.http.start;

**for** (auto params : config.http.include)

**if** (params != "")

config\_file << config.http.tab + params;

**for** (auto params : config.http.global\_headers)

**if** (params != "")

config\_file << config.http.tab + params;

**for** (auto params : config.http.log\_formatting)

**if** (params != "")

config\_file << config.http.tab + params;

**for** (auto params : config.http.limit\_concurrency)

**if** (params != "")

config\_file << config.http.tab + params;

**for** (auto params : config.http.fast\_cgi)

**if** (params != "")

config\_file << config.http.tab + params;

**for** (auto params : config.http.load\_balancer)

**if** (params != "")

config\_file << config.http.tab + params;

**for** (auto params : config.http.geoip)

**if** (params != "")

config\_file << config.http.tab + params;

**for** (auto params : config.http.geoip\_blocks\_allow)

**if** (params != "")

config\_file << config.http.tab + params;

**for** (auto params : config.http.geoip\_blocks\_deny)

**if** (params != "")

config\_file << config.http.tab + "# " + params;

config\_file << config.server.start;

**for** (auto params : config.server.listen)

**if** (params != "")

config\_file << config.server.tab + params;

**for** (auto params : config.server.allow)

**if** (params != "")

config\_file << config.server.tab + params;

**for** (auto params : config.server.deny)

**if** (params != "")

config\_file << config.server.tab + params;

config\_file << config.server.server\_name;

config\_file << config.server.root\_folder;

config\_file << config.server.server\_token;

**for** (auto params : config.server.global\_logs\_status)

**if** (params != "")

config\_file << config.server.tab + params;

**for** (auto params : config.server.global\_logs\_settings)

**if** (params != "")

config\_file << config.server.tab + params;

**for** (auto params : config.server.ssl)

**if** (params != "")

config\_file << config.server.tab + params;

**for** (auto params : config.server.buffers)

**if** (params != "")

config\_file << config.server.tab + params;

**for** (auto params : config.server.timeouts)

**if** (params != "")

config\_file << config.server.tab + params;

**for** (auto params : config.server.keepalive)

**if** (params != "")

config\_file << config.server.tab + params;

**for** (auto params : config.server.gzip)

**if** (params != "")

config\_file << config.server.tab + params;

**for** (auto params : config.server.caching)

**if** (params != "")

config\_file << config.server.tab + params;

**for** (auto params : config.server.custom)

**if** (params != "")

config\_file << config.server.tab + params;

**for** (auto location : config.locations)

{

config\_file << location.start;

**for** (auto params : location.custom\_logs\_status)

**if** (params != "")

config\_file << location.tab + params;

**for** (auto params : location.custom\_logs\_settings)

**if** (params != "")

config\_file << location.tab + params;

**for** (auto params : location.auth)

**if** (params != "")

config\_file << location.tab + params;

**for** (auto params : location.fast\_cgi)

**if** (params != "")

config\_file << location.tab + params;

**if** (location.expires != "")

config\_file << location.tab + location.expires;

**for** (auto params : location.custom\_headers)

**if** (params != "")

config\_file << location.tab + params;

**if** (location.proxy\_pass != "")

config\_file << location.tab + location.proxy\_pass;

**for** (auto params : location.custom)

**if** (params != "")

config\_file << location.tab + params;

config\_file << location.end;

}

config\_file << config.server.end;

**for** (auto params : config.http.custom)

**if** (params != "")

config\_file << config.http.tab + params;

config\_file << config.http.end;

config\_file.close();

}

void applyConfiguration()

{

string userInput;

cout << "\n\tКакую конфигурацию необходимо применить? (default/preset/custom): ";

cout << "\n\t\t[default] Применить стандартную конфигурацию для Ubuntu LTS 22.04";

cout << "\n\t\t[preset] Применить рекомендованную конфигурацию, разработанную в рамках курса";

cout << "\n\t\t[custom] Составить собственную конфигурацию";

cout << "\n\n\tВаш выбор: ";

getline(cin, userInput);

**if** (userInput == "default")

{

ifstream srcFile("nginx.conf.ubuntu\_template", ios::binary);

ofstream dstFile(config.location + (config.location.back() == '/' ? "" : "/") + "nginx.conf", ios::binary);

**if** (srcFile &amp;&amp; dstFile)

{

dstFile << srcFile.rdbuf();

cout << "\n\tФайл конфигурации успешно создан.";

}

**else**

cout << "\n\tНе удалось создать файл конфигурации.";

}

**else** **if** (userInput == "preset")

{

ifstream srcFile("nginx.conf.mine", ios::binary);

ofstream dstFile("nginx.conf", ios::binary);

**if** (srcFile &amp;&amp; dstFile)

{

dstFile << srcFile.rdbuf();

cout << "\n\tФайл конфигурации успешно создан.";

}

**else**

cout << "\n\tНе удалось создать файл конфигурации.";

}

**else** **if** (userInput == "custom")

{

ofstream customFile("nginx.conf");

**if** (customFile)

{

cout << "\n\tДавайте приступим к персональной настройке файла конфигурации!";

setEventsBlock();

setHttpBlock();

setServerBlock();

setLocations();

pushAllBlocks();

}

**else**

cout << "\n\tНе удалось создать файл конфигурации.";

}

**else**

cout << "\n\tНекорректный ввод.";

}

int main()

{

cout << "\n\tПриветствуем в конфигураторе nginx!\n\n\tУкажите желаемое расположение nginx.conf:";

// cin >> config.location;

cout << "\n\t";

config.location = "/home/alse0722/Desktop/kurs4/pract";

applyConfiguration();

**return** 0;

}

ПРИЛОЖЕНИЕ С

Пример результата работы программы-конфигуратора formatter

events {

worker\_connections 1024;

multi\_accept on;

accept\_mutex off;

*#test dop events;*

}

http {

include mime.types;

default\_type application/octet-stream;

include fastcgi\_params;

include fastcgi.conf;

log\_format upstream\_time '$remote\_addr - $remote\_user [$time\_local] '

'"$request" $status $body\_bytes\_sent '

'"$http\_referer" "$http\_user\_agent"'

'rt=$request\_time uct="$upstream\_connect\_time" '

'uht="$upstream\_header\_time" urt="$upstream\_response\_time"';

log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

'$status $body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

limit\_conn\_zone $binary\_remote\_addr zone=per\_ip:5m;

fastcgi\_cache\_path /path/to/FastCGI/cache levels=1:2 keys\_zone=microcache:10m max\_size=500m inactive=10m;

fastcgi\_cache\_key "$scheme$request\_method$host$request\_uri";

fastcgi\_ignore\_headers Cache-Control Expires **Set**-Cookie ;

upstream group1 {

least\_conn;

server 192.168.0.1;

server 192.168.0.2;

server 192.168.0.3;

}

geoip2 /custom/path/to/db/;GeoLite2-City.mmdb {

auto\_reload 60m;

$geoip2\_metadata\_city\_build metadata build\_epoch;

$geoip2\_data\_country\_name country names en;

$geoip2\_data\_country\_code country iso\_code;

$geoip2\_data\_city\_name city names en;

$geoip2\_data\_region\_name subdivisions 0 names en;

$geoip2\_data\_state\_code subdivisions 0 iso\_code;

}

map "$geoip2\_data\_country\_code:$geoip2\_data\_state\_code" $allowed\_reg {

default no;

default no;

~^RU: yes;

~^BY: yes;

~^AM: yes;

~^KZ: yes;

~^KG: yes;

UA:40 yes;

UA:43 yes;

UA:14 yes;

UA:09 yes;

UA:23 yes;

UA:65 yes;

GE:AB yes;

}

*# map "$geoip2\_data\_country\_code:$geoip2\_data\_state\_code" $allowed\_reg {*

*# default no;*

*# ~^RU: yes;*

*# ~^BY: yes;*

*# UA:43 yes;*

*# UA:14 yes;*

*# UA:09 yes;*

*# }*

server {

listen 80;

listen 81;

listen 443 ssl;

allow 192.168.0.170;

allow 192.168.0.171;

allow 192.168.0.173;

deny all;

server\_name SERVER

root /root/path/to/site

server\_tokens off;

error\_log off;

access\_log off;

ssl\_certificate /path/to/ssl/nginx.crt;

ssl\_certificate\_key /path/to/ssl/nginx.key;

ssl\_session\_cache shared:SSL:1m;

ssl\_session\_timeout 5m;

ssl\_ciphers HIGH:!aNULL:!MD5;

ssl\_prefer\_server\_ciphers on;

client\_body\_buffer\_size 16k;

client\_header\_buffer\_size 1k;

client\_max\_body\_size 8m;

large\_client\_header\_buffers 2 1k;

client\_header\_timeout 10;

client\_body\_timeout 15;

keepalive\_timeout 65;

send\_timeout 10;

gzip on;

gzip\_min\_length 100;

gzip\_comp\_level 3;

gzip\_types text/css;

gzip\_types text/php;

gzip\_types text/html;

gzip\_disable "msie6";

set $no\_cache 0;

**if** (test\_cgi\_rule\_1) { set $no\_cache 1; }

**if** (test\_cgi\_rule\_2) { set $no\_cache 1; }

**if** ($allowed\_reg = no) {

**return** 444;

}

*#test dop server;*

location ~ /\.ht {

deny all;

}

location / {

access\_log off;

error\_log /logs//errors\_slash.log yes;

}

location /root {

error\_log /logs/root\_error.log yes;

error\_log /logs/root\_access.log yes;

auth\_basic "Restricted access";

auth\_basic\_user\_file /path/to/data/.passwds;

fastcgi\_cache microcache;

fastcgi\_cache\_valid 200 1M;

fastcgi\_pass localhost:9000;

fastcgi\_no\_cache $no\_cache;

expires yes;

add\_header Cached-**with**-CGI nginx output;

http://group1;

*#no;*

}

}

*#test dop http;*

}