Хочу рассказать об одной полезной возможности nginx, которую регулярно использую в своих делах. Речь пойдет о настройке проксирования запросов на удаленный сервер с помощью nginx и директивы proxy_pass. Я приведу примеры различных настроек и расскажу, где сам использую данный модуль популярного веб сервера.

Содержание:

Введение

Немного расскажу своими словами о том, как работает модуль **ngx_http_proxy_module**. Именно он реализует весь функционал, о котором пойдет речь. Допустим, у вас в локальной или виртуальной сети есть какие-то сервисы, не имеющие прямого доступа из интернета. А вы хотите таковой иметь. Можно пробрасывать нужные порты на шлюзе, можно что-то еще придумывать. А можно сделать проще всего — настроить единую точку входа на все свои сервисы в виде nginx сервера и с него проксировать различные запросы к нужным серверам.

Расскажу на конкретных примерах, где я это использую. Для наглядности и простоты буду прям по порядку перечислять эти варианты:

1. Ранее я рассказывал о настройке чат серверов — matrix и mattermost. В этих статьях я как раз рассказывал о том, как проксировать запросы в чат с

помощью nginx. Прошелся по теме вскользь, не останавливаясь подробно. Суть в том, что вы настраиваете на любом виртуальном сервере эти чаты, помещаете их в закрытые периметры сети без лишних доступов и просто проксируете запросы на эти сервера. Они идут через nginx, который у вас смотрит во внешний интернет и принимает все входящие соединения.

2. Допустим, у вас есть большой сервер с множеством контейнеров, например докера. На нем работает множество различных сервисов. Вы устанавливаете еще один контейнер с чистым nginx, на нем настраиваете проксирование запросов на эти контейнеры. Сами контейнеры мапите только к локальному интерфейсу сервера. Таким образом, они будут полностью закрыты извне, и при этом вы можете гибко управлять доступом.
3. Еще один популярный пример. Допустим, у вас есть сервер с гипервизором proxmox или любым другим. Вы настраиваете на одной из виртуальных машин шлюз, создаете локальную сеть только из виртуальных машин без доступа в нее извне. Делаете в этой локальной сети для всех виртуальных машин шлюз по-умолчанию в виде вашей виртуальной машины со шлюзом. На виртуальных серверах в локальной сети размещаете различные сервисы и не заморачиваетесь с настройками фаервола на них. Вся их сеть все равно не доступна из интернета. А доступ к сервисам проксируете с помощью nginx, установленным на шлюз или на отдельной виртуальной машине с проброшенными на нее портами.
4. Мой личный пример. У меня дома есть сервер synology. Я хочу организовать к нему простой доступ по https из браузера по доменному имени. Нет ничего проще. Настраиваю на сервере nginx получение бесплатного сертификата Let’s encrypt, настраиваю проксирование запросов на мой домашний ip, там на шлюзе делаю проброс внутрь локалки на synology сервер. При этом я могу фаерволом ограничить доступ к серверу только одним ip, на котором работает nginx. В итоге на самом synology вообще ничего не надо делать. Он и знать не знает, что к нему заходят по https, по стандартному порту 443.
5. Допустим, у вас большой проект, разбитый на составные части, которые живут на разных серверах. К примеру, на отдельном сервере живет форум, по пути /forum от основного домена. Вы просто берете и настраиваете проксирование всех запросов по адресу /forum на отдельный сервер. Точно так же можно без проблем все картинки перенести на другой сервер и проксировать к ним запросы. То есть вы можете создать любой location и переадресовывать запросы к нему на другие сервера.

Надеюсь в общем и целом понятно, о чем идет речь. Вариантов использования много. Я привел самые распространенные, которые пришли в голову и которые использую сам. Из плюсов, которые считаю наиболее полезными именно из своих кейсов, отмечу 2:

- Вы без проблем можете настроить https доступ к сервисам, при этом совершенно не трогая эти сервисы. Вы получаете и используете сертификаты на nginx сервере, используете https соединение с ним, а сам nginx уже передает информацию на сервера со службами, которые могут работать по обычному http и знать не знают о https.
- Вы очень легко можете менять адреса, куда проксируете запросы. Допустим у вас есть сайт, его запросы проксируются на отдельный сервер. Вы подготовили обновление или переезд сайта. Отладили все на новом сервере. Теперь вам достаточно на сервере nginx изменить адрес старого сервера на новый, куда будут перенаправляться запросы. И все. Если что-то пойдет не так, можете оперативно вернуть все обратно.

С теорией закончил. Перейдем теперь к примерам настройки. В своих примерах я буду использовать следующие обозначения:

blog.zeroxzed.ru	доменное имя тестового сайта
nginx_srv	имя внешнего сервера с установленным nginx
blog_srv	локальный сервер с сайтом, куда проксируем соединения
94.142.141.246	внешний ip nginx_srv
192.168.13.31	ip адрес blog_srv
77.37.224.139	ip адрес клиента, с которого я буду заходить на сайт

Настройка proxy_pass в nginx

Рассмотрим самый простой пример. Буду использовать свой технический домен zeroxzed.ru в этом и последующих примерах. Допустим, у нас есть сайт blog.zeroxzed.ru. В DNS создана A запись, указывающая на ip адрес сервера, где установлен nginx — nginx_srv. Мы будем проксировать все запросы с этого сервера на другой сервер в локальной сети blog_srv, где реально размещается сайт. Рисуем конфиг для секции server.

```
server {
    listen 80;
    server_name blog.zeroxzed.ru;
    access_log /var/log/nginx/blog.zeroxzed.ru-access.log;
    error_log /var/log/nginx/blog.zeroxzed.ru-error.log;

location / {
    proxy_pass http://192.168.13.31;
    proxy_set_header Host $host;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Real-IP $remote_addr;
    }
}
```

Заходим по адресу *http://blog.zeroxzed.ru*. Мы должны попасть на blog_srv, где тоже должен работать какой-то веб сервер. В моем случае это будет тоже nginx. У вас должно открыться содержимое, аналогичное тому, что вы увидите, набрав *http://192.168.13.31* в локальной сети. Если что-то не работает, то проверьте сначала, что по адресу директивы proxy_pass у вас все корректно работает.

Посмотрим логи на обоих сервера. На nginx_srv вижу свой запрос:

```
77.37.224.139 - - [19/Jan/2018:15:15:40 +0300] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; rv:11.0) like Gecko"
```

Проверяем blog_srv:

```
94.142.141.246 - - [19/Jan/2018:15:15:40 +0300] "GET / HTTP/1.0" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; rv:11.0) like Gecko" "77.37.224.139"
```

Как мы видим, запрос сначала пришел на nginx_srv, был переправлен на blog_srv, куда он пришел уже с адресом отправителя 94.142.141.246. Это адрес nginx_srv. Реальный же ip адрес клиента мы видим только в самом конце лога. Это неудобно, так как директива php **REMOTE_ADDR** не будет возвращать настоящий ip адрес клиента. А он очень часто бывает нужен. Мы это дальше исправим, а пока создадим в корне сайта на chat_srv тестовую страничку для проверки ip адреса клиента следующего содержания:

```
<?php
echo $_SERVER['REMOTE_ADDR']
?>
```

Назовем ее *myip.php*. Перейдем по адресу *http://blog.zeroxzed.ru/myip.php* и проверим, как сервер определит наш адрес. Никак не определит :) Он покажет адрес nginx_srv. Исправляем это и учим nginx передавать реальный ip адрес клиента на сервер.

Передача реального ip (real ip) адреса клиента в nginx при proxy_pass

В предыдущем примере мы на самом деле передаем реальный ip адрес клиента с помощью директивы **proxy_set_header**, которая добавляет в заголовок **X-Real-IP** настоящий ip адрес клиента. Теперь нам нужно на принимающей стороне, то есть blog_srv сделать обратную замену — заменить информацию об адресе отправителя на ту, что указана в заголовке X-Real-IP. Добавдяем в секцию server следующие параметры:

```
set_real_ip_from 94.142.141.246;
real_ip_header X-Real-IP;
```

Полностью секция server на blog_srv в самом простом варианте получается следующей:

```
    server {
        listen       80 default_server;
        server_name  blog.zeroxzed.ru;
        root         /usr/share/nginx/html;
        set_real_ip_from 94.142.141.246;
        real_ip_header X-Real-IP;
    location / {
        index index.php index.html index.htm;
        try_files        $uri $uri/        =404;
        }

    location ~ \.php$ {
        fastcgi_pass   127.0.0.1:9000;
        fastcgi_index  index.php;
        fastcgi_intercept_errors on;
        include fastcgi_params;
        fastcgi_param       SCRIPT_FILENAME  $document_root$fastcgi_script_name;
        fastcgi_ignore_client_abort     off;
        }

    error_page 404 /404.html;
        location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
        location = /50x.html {
    }
}
```

Сохраняем конфиг, перечитываем его и снова проверяем *http://blog.zeroxzed.ru/myip.php*. Вы должны увидеть свой реальный ip адрес. Его же вы увидите в

логе web сервера на blog_srv.

Дальше рассмотрим более сложные конфигурации.

Передача https через nginx с помощью proxy pass

Если у вас сайт работает по https, то достаточно настроить ssl только на nginx_srv, если вы не беспокоитесь за передачу информации от nginx_srv к blog_srv. Она может осуществляться по незащищенному протоколу. Рассмотрю пример с бесплатным сертификатом let's encrypt. Это как раз один из кейсов, когда я использую proxy_pass. Очень удобно настроить на одном сервере автоматическое получение всех необходимых сертификатов. Подробно настройку let's encrypt я рассматривал отдельно. Сейчас будем считать, что у вас стоит certbot и все готово для нового сертификата, который потом будет автоматически обновляться.

Для этого нам надо на nginx_srv добавить еще один location — /.well-known/acme-challenge/. Полная секция server нашего тестового сайта на момент получения сертификата будет выглядеть вот так:

```
server {
    listen 80;
    server_name blog.zeroxzed.ru;
    access_log /var/log/nginx/blog.zeroxzed.ru-access.log;
    error_log /var/log/nginx/blog.zeroxzed.ru-error.log;

    location /.well-known/acme-challenge/ {
        root /web/sites/blog.zeroxzed.ru/www/;
    }
```

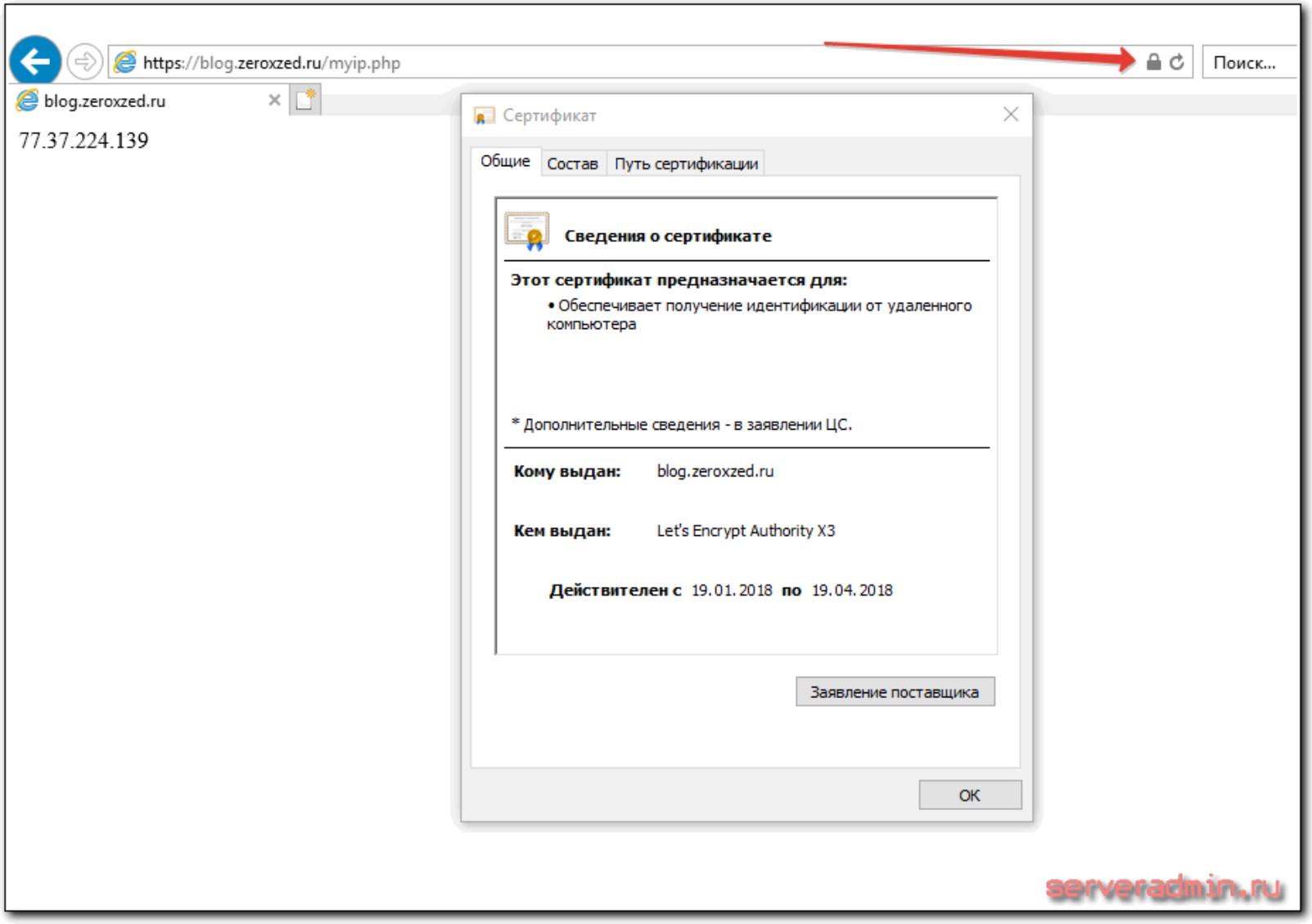
```
    location / {
        proxy_pass http://192.168.13.31;
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```

Перечитывайте конфиг nginx и получайте сертификат. После этого конфиг меняется на следующий:

```
server {
    listen 80;
    server_name blog.zeroxzed.ru;
    access_log /var/log/nginx/blog.zeroxzed.ru-access.log;
    error_log /var/log/nginx/blog.zeroxzed.ru-error.log;
    return 301 https://$server_name$request_uri; # редирект обычных запросов на https
    }

server {
    listen 443 ssl http2;
    server_name blog.zeroxzed.ru;
    access_log /var/log/nginx/blog.zeroxzed.ru-ssl-access.log;
    error_log /var/log/nginx/blog.zeroxzed.ru-ssl-error.log;

    ssl on;
    ssl_certificate /etc/letsencrypt/live/blog.zeroxzed.ru/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/blog.zeroxzed.ru/privkey.pem;
    ssl_session_timeout 5m;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_dhparam /etc/ssl/certs/dhparam.pem;
    ssl_ciphers 'EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH';
    ssl_prefer_server_ciphers on;
```

```
    ssl_session_cache shared:SSL:10m;

    location /.well-known/acme-challenge/ {
        root /web/sites/blog.zeroxzed.ru/www/;
    }
    location / {
        proxy_pass http://192.168.13.31;
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```

Проверяем, что получилось.


https://blog.zeroxzed.ru/myip.php
Поиск...
blog.zeroxzed.ru
77.37.224.139
Сертификат
Общие
Состав
Путь сертификации
Сведения о сертификате
Этот сертификат предназначается для:
• Обеспечивает получение идентификации от удаленного компьютера
* Дополнительные сведения - в заявлении ЦС.
Кому выдан: blog.zeroxzed.ru
Кем выдан: Let's Encrypt Authority X3
Действителен с 19.01.2018 по 19.04.2018
Заявление поставщика
OK
serveradmin.ru

Наш сайт работает по https, при том, что мы вообще не трогали сервер, где этот сайт располагается. Конкретно с web сайтом это может быть не так актуально, но если вы проксируете запросы не на обычный сайт, а на какой-то нестандартный сервис, который трудно перевести на https, это может быть хорошим решением.

Проксирование определенной директории или файлов

Рассмотрим еще один пример. Допустим, у вас форум живет в директории *http://blog.zeroxzed.ru/forum/*, вы хотите вынести форум на отдельный web сервер для увеличения быстродействия. Для этого к предыдущему конфигу добавьте еще один location.

```
location /forum/ {
        proxy_pass http://192.168.13.31;
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_redirect default;
        }
```

Еще одно популярное решение. Вы можете отдавать картинки с одного сервера, а все остальное с другого. В моем примере, картинки будут жить на том же сервере, где nginx, а остальной сайт на другом сервере. Тогда у нас должна быть примерно такая конфигурация локаций.

```
location / {
        proxy_pass http://192.168.13.31;
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Real-IP $remote_addr;
        }

location ~ \.(gif|jpg|png)$ {
        root /web/sites/blog.zeroxzed.ru/www/images;
        }
```

Чтобы все это работало корректно, необходимо, чтобы сам сайт умел правильно размещать свои изображения. Вариантов это организовать множество.

Можно как на сервере монтировать сетевые паки различными способами, так и программистам изменять код для управления размещением изображений. В любом случае, это комплексный подход к работе с сайтом.

Существует очень много директив для управления прокси-соединениями. Все они описаны в соответствующей документации nginx. Я не большой специалист по настройке nginx. В основном использую свои готовые конфиги, зачастую даже не вникая в суть, если получается сразу решить задачу. Подсматриваю что-то у других, записываю к себе, стараюсь разобраться.

Особое внимание следует уделить директивам кэширования proxy_cache, если в этом есть потребность. Можно существенно увеличить отклик веб сайта, если подходящим образом настроить отдачу кэша. Но это тонкий момент и нужно настраивать в каждом конкретном случае отдельно. Готовых рецептов тут не бывает.

Более подробно о комплексной настройке nginx читайте в отдельной большой статье с моими личными примерами.

Заключение

> Не понравилась статья и хочешь научить меня администрировать? Пожалуйста, я люблю учиться. Комментарии в твоем распоряжении. Расскажи, как сделать правильно!

На этом у меня все. Не рассмотрел еще один возможный вариант, когда вы проксируете https сайт и передаете инфу на бэкенд тоже по https. Нет под рукой готового примера, чтобы проверить, а заочно не стал рисовать конфиг. По идее, ничего сложного в этом нет, настраиваете nginx на обоих серверах с одним и тем же сертификатом. Но наверняка не скажу, что все заработает при этом. Возможно, есть какие-то нюансы с таким проксированием. Мне обычно не приходится так делать.

Как я уже писал в начале, в основном проксирую запросы с одного внешнего веб сервера на закрытый периметр сети, куда нет никому доступа. В этом случае у меня нет необходимости использовать https при передаче запросов на бэкенд. В качестве бэкенда не обязательно будет отдельный сервер. Это запросто может быть контейнер на этом же сервере.