**NGINX as a file server**

NGINX is good at serviing static files such as images and html files. It means it's already a great server for downloading. In this article I expand it by adding features of authentication, uploading and deleting files using lua. I will also talk about the community's favorite nginx-upload-module at the end of this article.

**CONTENTS**

* [Download](https://www.yanxurui.cc/posts/server/2017-03-21-NGINX-as-a-file-server/#download)
* [Authentication](https://www.yanxurui.cc/posts/server/2017-03-21-NGINX-as-a-file-server/#authentication)
  + [auth\_basic](https://www.yanxurui.cc/posts/server/2017-03-21-NGINX-as-a-file-server/#auth_basic)
  + [auth\_basic\_user\_file](https://www.yanxurui.cc/posts/server/2017-03-21-NGINX-as-a-file-server/#auth_basic_user_file)
* [upload](https://www.yanxurui.cc/posts/server/2017-03-21-NGINX-as-a-file-server/#upload)
  + [lua-resty-upload](https://www.yanxurui.cc/posts/server/2017-03-21-NGINX-as-a-file-server/#lua-resty-upload)
  + [nginx\_upload.conf](https://www.yanxurui.cc/posts/server/2017-03-21-NGINX-as-a-file-server/#nginx_uploadconf)
  + [my\_upload.lua](https://www.yanxurui.cc/posts/server/2017-03-21-NGINX-as-a-file-server/#my_uploadlua)
  + [my\_delete.lua](https://www.yanxurui.cc/posts/server/2017-03-21-NGINX-as-a-file-server/#my_deletelua)
  + [usage](https://www.yanxurui.cc/posts/server/2017-03-21-NGINX-as-a-file-server/#usage)
* [nginx-upload-module](https://www.yanxurui.cc/posts/server/2017-03-21-NGINX-as-a-file-server/#nginx-upload-module)
  + [common usage](https://www.yanxurui.cc/posts/server/2017-03-21-NGINX-as-a-file-server/#common-usage)
  + [Hack it as a normal file server](https://www.yanxurui.cc/posts/server/2017-03-21-NGINX-as-a-file-server/#hack-it-as-a-normal-file-server)
* [reference](https://www.yanxurui.cc/posts/server/2017-03-21-NGINX-as-a-file-server/#reference)

## Download

When we send a get request, NGINX searches for a file by appending URI to the path specified by root. If the URI ends with a slash, NGINX treats it as a directory and tries to find an index file which is index.html by default in it. If such a file can not be found, NGINX returns HTTP code **403(Forbidden)**. The [ngx\_http\_autoindex\_module](http://nginx.org/en/docs/http/ngx_http_autoindex_module.html) can return an automatically generated directory listing in this case.

server {

listen 8001; # a customed port

# download

autoindex on; # enable directory listing output

autoindex\_exact\_size off; # output file sizes rounded to kilobytes, megabytes, and gigabytes

autoindex\_localtime on; # output local times in the directory

location / {

root upload;

}

}

upload is a folder under NGINX's prefix which is /opt/nginx in my case:

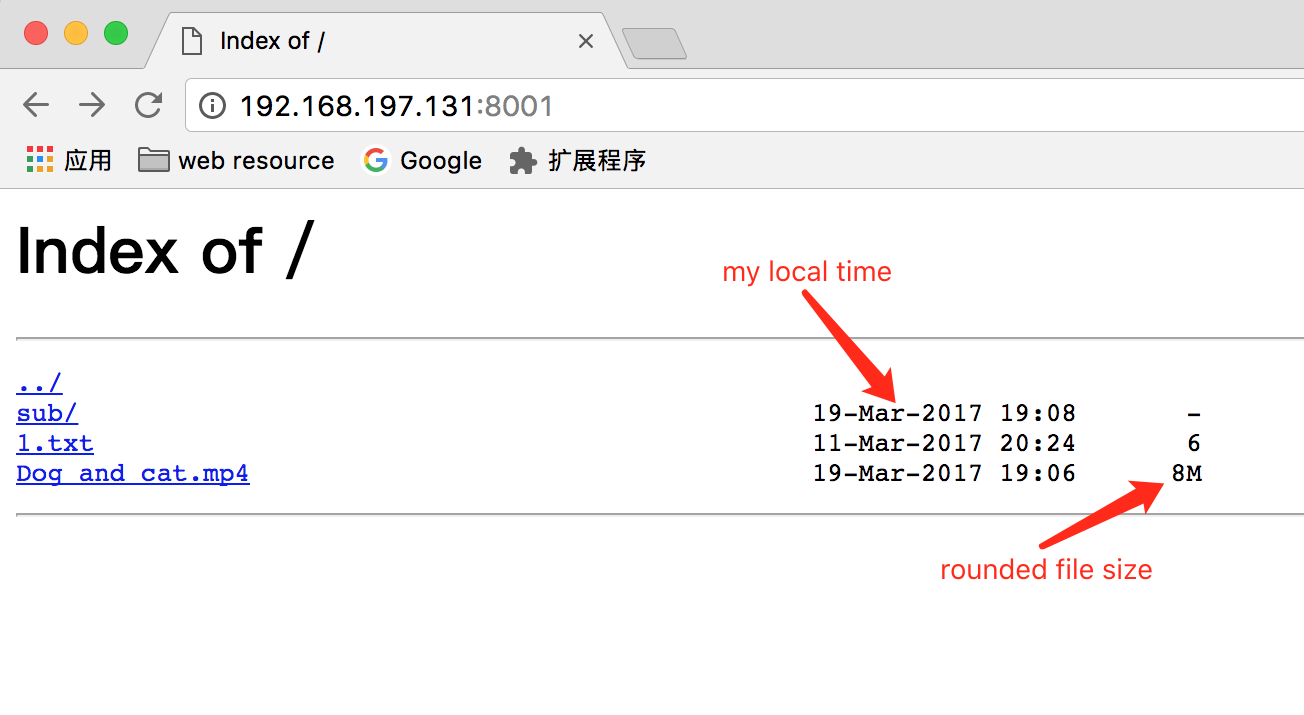
upload/

├── 1.txt

├── Dog and cat.mp4

└── sub

└── 2.txt



If you use a modern browser, you can directly preview many file types such as videos, pdfs, etc. Like the built-in autoindex module, [aperezdc/ngx-fancyindex](https://github.com/aperezdc/ngx-fancyindex) is a fancier alternative of autoindexing by adding customized theme.

Notice: Executable permissions of the listing directory are required besides read permissions.

location / {

root upload;

}

is equivalent to:

location /download {

alias upload;

}

See [Nginx — static file serving confusion with root & alias](http://stackoverflow.com/questions/10631933/nginx-static-file-serving-confusion-with-root-alias)

## Authentication

The [ngx\_http\_auth\_basic\_module](http://nginx.org/en/docs/http/ngx_http_auth_basic_module.html) allows limiting access to resources by validating the user name and password using "HTTP Basic Authentication" protocol.

server {

# auth

auth\_basic "Restricted site";

auth\_basic\_user\_file /opt/nginx/.htpasswd;

}

Only two directives are provided by this module.

### auth\_basic

enables authentication and the string paramter is used as realm. Some browers can display this message in the prompt.

### auth\_basic\_user\_file

is the file that keeps user names and **hashed passwords**. Most directives which need a path, access\_log for example, can take a relative path(relative to nginx's prefix) as argument . But auth\_basic\_user\_file must be an absolute path otherwith you will see "403 Forbidden" error page.

Let's generate this file by OpenSSL utilities which may already be available on most servers:

echo -n 'foo:' >> .htpasswd

openssl passwd >> .htpasswd

# type your password twice

cat .htpasswd

foo:xOVvNJCt4.P76

add another user bar:

echo -n 'bar:' >> .htpasswd

openssl passwd -apr1 >> .htpasswd

# type your password twice

cat .htpasswd

foo:xOVvNJCt4.P76

bar:$apr1$/hbFh44e$D5RZ91WBHCQlBymeuMCIv.

-apr1 means the password is hashed with the Apache variant of the MD5-based password algorithm which is more secure than the default **crypt** algorithm.

Notice that the encrypted password will be different each time even if you use an identical password. It's magical, isn't it? Because a unique salt is chosen each time. The first 2 characters of crypt's output is salt and in the second case the output format is $apr1$salt$hash. If you specify salt by using -salt option you will always get the same result.

Now refresh the page, user will be asked for username and password. If you enter the correct credentials, you will be allowed to access these locations. Otherwith, you will see "401 Authorization Required" error page.

## upload

I know it's not the best way of handling file upload, but NGINX is versatile enough to achieve this without any backend support. Until now, all the features above are provided by nginx itself. Making NGINX support uploading needs help from NGINX 3rd Party Modules](<https://www.nginx.com/resources/wiki/modules/>).

### lua-resty-upload

Here I use [openresty/lua-resty-upload](https://github.com/openresty/lua-resty-upload) which is a lua module based on ngx\_lua cosocket. It **requires [lua-nginx-module](https://github.com/openresty/lua-nginx-module)** to be compiled into NGINX.

lua-resty-upload contains only one file upload.lua. Place this file to /usr/local/lib/lua/5.1/resty/upload.lua and then add it to ngx\_lua's LUA\_PATH search path by directive lua\_package\_path.

lua\_package\_path '/usr/local/lib/lua/5.1/?.lua;;';

This module is so simple that user just needs to call the read method chunk by chunk. Please refer it’s readme documentation for details about this api.

### nginx\_upload.conf

Bellow is the full configuration:

pid logs/nginx\_upload.pid;

events {

worker\_connections 1024;

}

http {

lua\_package\_path '/usr/local/lib/lua/5.1/?.lua;;';

server {

listen 8001;

# download

autoindex on;

autoindex\_exact\_size off;

autoindex\_localtime on;

# auth

auth\_basic "Restricted site";

auth\_basic\_user\_file /opt/nginx/.htpasswd;

location /download {

alias upload;

}

location ~ ^/upload\_lua(/.\*)?$ {

set $store\_path upload$1/;

content\_by\_lua\_file conf/lua/my\_upload.lua;

}

location ~ ^/delete/(.\*)$ {

set $file\_path upload/$1;

content\_by\_lua\_file conf/lua/my\_delete.lua;

}

}

}

### my\_upload.lua

local upload = require "resty.upload"

local function my\_get\_file\_name(header)

local file\_name

for i, ele in ipairs(header) do

file\_name = string.match(ele, 'filename="(.\*)"')

if file\_name and file\_name ~= '' then

return file\_name

end

end

return nil

end

local chunk\_size = 4096

local form = upload:new(chunk\_size)

local file

local file\_path

while true do

local typ, res, err = form:read()

if not typ then

ngx.say("failed to read: ", err)

return

end

if typ == "header" then

local file\_name = my\_get\_file\_name(res)

if file\_name then

file\_path = ngx.var.store\_path..file\_name

file = io.open(file\_path, "w+")

if not file then

ngx.say("failed to open file ", file\_path)

return

end

end

elseif typ == "body" then

if file then

file:write(res)

end

elseif typ == "part\_end" then

if file then

file:close()

file = nil

ngx.say("upload to "..file\_path.." successfully!")

end

elseif typ == "eof" then

break

else

-- do nothing

end

end

Every time a multipart/form-data form field is encouterred a file is created using the original file name if the header of that field contains filename attribute.

### my\_delete.lua

I also add a location which enables user to delete a file or an empty folder.

local function file\_exists(path)

local file = io.open(path, "rb")

if file then file:close() end

return file ~= nil

end

if not file\_exists(ngx.var.file\_path) then

ngx.say("file not found: "..ngx.var.file\_path)

end

r, err = os.remove(ngx.var.file\_path)

if not r then

ngx.say("failed to delete: "..err)

else

ngx.say("delete successfully!")

end

### usage

❯ curl -H "Authorization: Basic Zm9vOjEyMzQ1Ng==" -F filea=@a.txt -F fileb=@b.txt http://192.168.197.131:8001/upload\_lua

upload to upload/a.txt successfully!

upload to upload/b.txt successfully!

I also set a variable store\_path to denote the path where this file should be saved. Including the part after /upload\_lua/ of URI enables user to control in which sub folder to save files in this path. But such sub folders should already exist or error will occur.

❯ curl -H "Authorization: Basic Zm9vOjEyMzQ1Ng==" -F filea=@a.txt -F fileb=@b.txt http://192.168.197.131:8001/upload\_lua/sub

upload to upload/sub/a.txt successfully!

upload to upload/sub/b.txt successfully!

I prefer cURL to Postman for debugging. Postman can't display the full request body of Content-Type multipart/form-data for now. cURL with --trace-ascii option can dump all incoming and outgoing data flow although it's a little hard to read. Here is an example:

❯ curl -H "Authorization: Basic Zm9vOjEyMzQ1Ng==" -F filea=@a.txt --trace-ascii - http://192.168.197.131:8001/upload\_lua

== Info: Trying 192.168.197.131...

== Info: Connected to 192.168.197.131 (192.168.197.131) port 8001 (#0)

=> Send header, 263 bytes (0x107)

0000: POST /upload\_lua HTTP/1.1

001b: Host: 192.168.197.131:8001

0037: User-Agent: curl/7.49.1

0050: Accept: \*/\*

005d: Authorization: Basic Zm9vOjEyMzQ1Ng==

0084: Content-Length: 191

0099: Expect: 100-continue

00af: Content-Type: multipart/form-data; boundary=--------------------

00ef: ----2ccbb2d137903ef2

0105:

<= Recv header, 23 bytes (0x17)

0000: HTTP/1.1 100 Continue

=> Send data, 136 bytes (0x88)

0000: --------------------------2ccbb2d137903ef2

002c: Content-Disposition: form-data; name="filea"; filename="a.txt"

006c: Content-Type: text/plain

0086:

=> Send data, 7 bytes (0x7)

0000: hi yxr!

=> Send data, 48 bytes (0x30)

0000:

0002: --------------------------2ccbb2d137903ef2--

<= Recv header, 17 bytes (0x11)

0000: HTTP/1.1 200 OK

<= Recv header, 22 bytes (0x16)

0000: Server: nginx/1.11.8

<= Recv header, 37 bytes (0x25)

0000: Date: Tue, 21 Mar 2017 03:15:05 GMT

<= Recv header, 40 bytes (0x28)

0000: Content-Type: application/octet-stream

<= Recv header, 28 bytes (0x1c)

0000: Transfer-Encoding: chunked

<= Recv header, 24 bytes (0x18)

0000: Connection: keep-alive

<= Recv header, 2 bytes (0x2)

0000:

<= Recv data, 48 bytes (0x30)

0000: 25

0004: upload to upload/a.txt successfully!.

002b: 0

002e:

upload to upload/a.txt successfully!

== Info: Connection #0 to host 192.168.197.131 left intact

As you can see the content of file a.txt is hi yxr!.

Deleting a file is as simple as:

❯ curl --user foo:123456 http://192.168.197.131:8001/delete/1.txt

delete successfully!

Now we have set up a simple file server using NINGX.

## nginx-upload-module

Actually I seek help from [nginx-upload-module](https://www.nginx.com/resources/wiki/modules/upload/) at first. It was an awesome module. However, the owner is no longer maintaining this module.

Below is what I tried with this module which may be helpful in case you want to play with it yourself.

You should download branch [2.2](https://github.com/vkholodkov/nginx-upload-module/tree/2.2) as it contains the newest commits.

There is a compile error: [md5.h: No such file or directory](https://github.com/vkholodkov/nginx-upload-module/issues/79) because NIGNX uses internal MD5 and SHA implementations and further the internal ngx\_md5.h is different from openssl's md5.h. Here is a simple patch which fixes this problem: [fix-md5.h-No-such-file-or-directory](https://www.yanxurui.cc/posts/server/2017-03-21-NGINX-as-a-file-server/0001-fix-md5.h-No-such-file-or-directory.patch)

If you see error simimlar to undefined reference to 'MD5\_Upddate' when compiling NGINX you may need to add --with-ld-opt='-lssl -lcrypto':

./configure --prefix=/opt/nginx \

--with-ld-opt='-lssl -lcrypto' \

--add-module=/home/yanxurui/nginx/nginx-upload-module

--with-debug

### common usage

This module is typlically used to handle file uploads without passing them to backend server. Here is a simple example from this module with slight modification.

# upload

client\_max\_body\_size 100m;

# Upload form should be submitted to this location

location /upload {

# Pass altered request body to this location

upload\_pass /example.php;

# 开启resumable

upload\_resumable on;

# Store files to this directory

# The directory is hashed, subdirectories 0 1 2 3 4 5 6 7 8 9 should exist

upload\_store /tmp/upload 1;

upload\_state\_store /tmp/state;

# Allow uploaded files to be read only by user

upload\_store\_access user:r;

# Set specified fields in request body

upload\_set\_form\_field "${upload\_field\_name}\_name" $upload\_file\_name;

upload\_set\_form\_field "${upload\_field\_name}\_content\_type" $upload\_content\_type;

upload\_set\_form\_field "${upload\_field\_name}\_path" $upload\_tmp\_path;

# Inform backend about hash and size of a file

upload\_aggregate\_form\_field "${upload\_field\_name}\_md5" $upload\_file\_md5;

upload\_aggregate\_form\_field "${upload\_field\_name}\_size" $upload\_file\_size;

upload\_pass\_form\_field "^submit$|^description$";

}

location ~ \.php$ {

# fastcgi\_pass unix:/run/php-fpm/php-fpm.sock;

fastcgi\_pass 127.0.0.1:9000;

fastcgi\_index index.php;

# fastcgi\_param SCRIPT\_FILENAME /scripts$fastcgi\_script\_name;

fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;

include fastcgi\_params;

}

example.php should be placed in html folder:

<?php

$header\_prefix = 'file';

$slots = 6;

?>

<html>

<head>

<title>Test upload</title>

</head>

<body>

<?php

if ($\_POST){

echo "<h2>Uploaded files:</h2>";

echo "<table border=\"2\" cellpadding=\"2\">";

echo "<tr><td>Name</td><td>Location</td><td>Content type</td><td>MD5</td><td>Size</tr>";

for ($i=1;$i<=$slots;$i++){

$key = $header\_prefix.$i;

if (array\_key\_exists($key."\_name", $\_POST) && array\_key\_exists($key."\_path",$\_POST)) {

$tmp\_name = $\_POST[$key."\_path"];

$name = $\_POST[$key."\_name"];

$content\_type = $\_POST[$key."\_content\_type"];

$md5 = $\_POST[$key."\_md5"];

$size = $\_POST[$key."\_size"];

echo "<tr><td>$name</td><td>$tmp\_name</td><td>$content\_type</td><td>$md5</td><td>$size</td>";

}

}

echo "</table>";

}else{?>

<h2>Select files to upload</h2>

<form name="upload" method="POST" enctype="multipart/form-data" action="/upload">

<input type="file" name="file1"><br>

<input type="file" name="file2"><br>

<input type="file" name="file3"><br>

<input type="file" name="file4"><br>

<input type="file" name="file5"><br>

<input type="file" name="file6"><br>

<input type="submit" name="submit" value="Upload">

<input type="hidden" name="test" value="value">

</form>

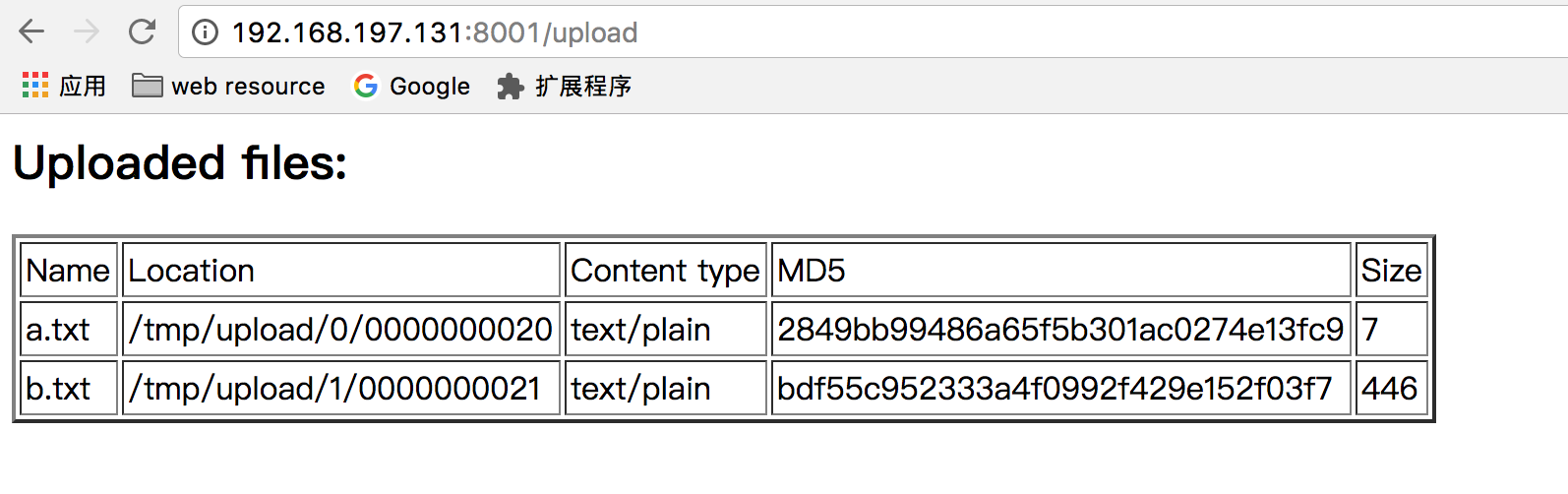
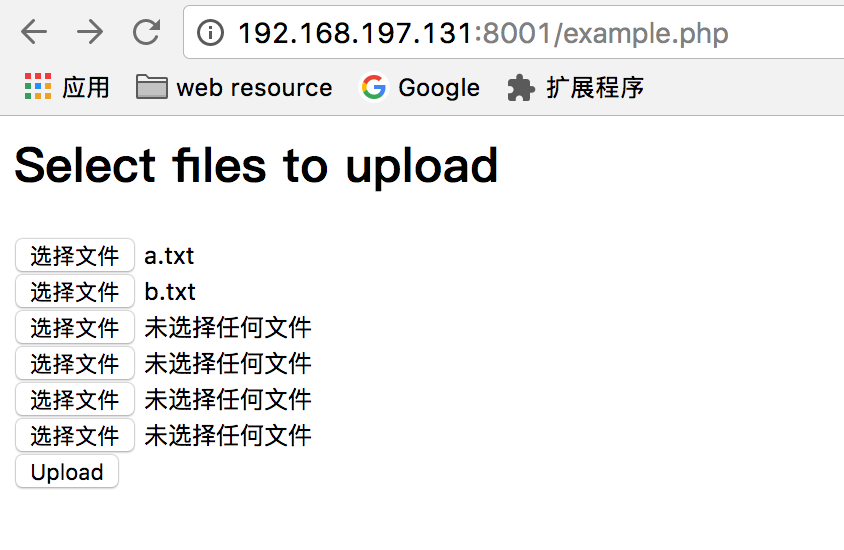
<?php

}

?>

</body>

</html>



Files are stored under directory specified by upload\_store directive using a simple hash algorithm. For example upload\_store upload 1 2 may result the file saved as /opt/nginx/upload/1/05/0000000018. Subdirectories 1 and 05 are both randomly selected and should exist before uploading. Hashed subdirectory is optional. File is renamed to a 10 digits to avoid conflict in case a file with the same name exists.

Fortunately, upload\_store can also accept variables.

### Hack it as a normal file server

What if I want to save the file as its original file name? Let's hack it.

This module implements a mechanism of [resumable file upload](http://www.grid.net.ru/nginx/resumable_uploads.en.html) aka partial upload. It means a big file can be splitted into severl segments and then uploaded one by one in seperate Post requests. Client is responsible for choosing an unique Session-ID which is an identifier of a file being uploaded as well as the name of file saved in server.

Yes, I can achieve my goal by using the file name as Session-ID. Apply this patch to allow Session-ID to contain dot. [hack-allow-session-id-to-contain-dot](https://www.yanxurui.cc/posts/server/2017-03-21-NGINX-as-a-file-server/0001-hack-allow-session-id-to-contain-dot.patch)

configuration now looks like this:

# Upload form should be submitted to this location

location ~ ^/upload\_mod(/.\*)?$ {

# Pass altered request body to this location

upload\_pass /upload\_return;

# 开启resumable

upload\_resumable on;

upload\_store upload$1;

upload\_state\_store /tmp/state;

}

location /upload\_return {

return 200 ok;

}

test file upload, Content-Type and X-Content-Range are omitted:

❯ curl -X POST --user foo:123456 -H "Content-Disposition: attachment, filename=\"a.txt\"" -H "X-Session-ID: a.txt" --data-binary @a.txt "http://192.168.197.131:8001/upload\_mod"

ok%

File a.txt is uploaded successfully:

❯ cat a.txt

hi yxr!

❯ curl --user foo:123456 "http://192.168.197.131:8001/download/a.txt"

hi yxr!

resumable file uploads

❯ curl -X POST --user foo:123456 -H "Content-Type: text/plain" -H "Content-Disposition: attachment, filename=\"a.txt\"" -H "X-Session-ID: a.txt" -H "X-Content-Range: bytes 0-4/10" --data-binary hello "http://192.168.197.131:8001/upload\_mod"

0-4/10

❯ curl -X POST --user foo:123456 -H "Content-Type: text/plain" -H "Content-Disposition: attachment, filename=\"a.txt\"" -H "X-Session-ID: a.txt" -H "X-Content-Range: bytes 5-9/10" --data-binary world "http://192.168.197.131:8001/upload\_mod"

ok

❯ curl --user foo:123456 "http://192.168.197.131:8001/download/a.txt"

helloworld

## reference

* [how to set up password authentication with nginx on ubuntu-14-04](https://www.digitalocean.com/community/tutorials/how-to-set-up-password-authentication-with-nginx-on-ubuntu-14-04)
* [Resumable uploads over HTTP. Protocol specification](http://www.grid.net.ru/nginx/resumable_uploads.en.html)

[« python里的数值类型](https://www.yanxurui.cc/posts/python/2017-03-31-python-numeric-types/)[使用robotframework测试nginx的location指令 »](https://www.yanxurui.cc/posts/server/2017-03-15-write-test-for-nginx-location-by-robotframework/)