# Review Questions:

What two commands do we use after making changes to an apache related .conf file?

* Configtest: **sudo apachectl configtest**
* And then reload the apache server:  **sudo systemctl reload apache2**

What directory is the .htaccess file located in?

* + /var/www/virtualhostname/ 🡨 to apply to individual VH’s.

What line do we add to the .htaccess file to allow custom error pages?

* + **ErrorDocument 404 /new404.html**

Where are custom error pages located?

* + /var/www/virtualhostname/

# Example of securing through .htaccess

In this example, we will create a new website called AvengersHQ. We’ll be adding some custom HTML to our index as well as some custom HTML to our error pages (Status 404 and 401). In addition, we’ll turn on basic authentication to only allow certain users to access the website. This is all done through configuration files and .htaccess.

1. Create a directory for the AvengersHQ website: **sudo mkdir /var/www/avengershq**

Text

Description automatically generated

1. Switch to this directory and change permissions:
   1. cd a
   2. sudo **chown -R $USER:$USER .**

Text

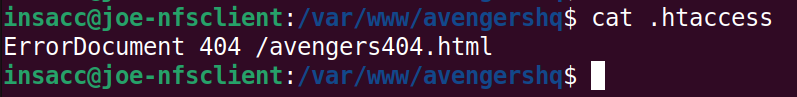
Description automatically generated

1. Create the index.html (have fun!)

Text

Description automatically generated

1. Create the .htaccess file and then the appropriate 404 error page.
   1. sudo vi .htaccess
   2. Enter “**ErrorDocument 404 /ErrorPages/avengers404.html**”



1. Create the avengers404.html
   1. **vi avengers404.html**
      1. Enter the following lines:

**<html>**

**<body bgcolor=”yellow”>**

**<h1>404 Page Not Found</h1>**

**<h2>Quinjet has been dispatched to your location!</h2>**

**</body>**

**</html>**

Text

Description automatically generated

1. Add [www.avengershq.com](http://www.avengershq.com) to our hosts file:
   1. **vi /etc/hosts** & add **127.0.1.1** [**www.avengershq.com**](http://www.avengershq.com)

Text

Description automatically generated

1. Next we need to edit/add additional configuration files to get this site up and running.
   1. Copy coos291.conf to **/etc/apache2/sites-available/avengershq.conf** and add make the appropriate changes: Text

      Description automatically generated
   2. Configtest and restart Apache
      1. **sudo apache2ctl configtest**
      2. **sudo systemctl restart apache2**
   3. Enable this site via the a2ensite command, configtest, and reload:
      1. **sudo a2ensite avengershq.conf**
      2. **sudo apache2ctl configtest**
      3. **sudo systemctl reload apache2**

Text

Description automatically generated

1. Browsing to [www.avengershq.com](http://www.avengershq.com) should give us our new website:

Graphical user interface, text, application

Description automatically generated

1. And purposely visiting a page that will result in a 404 should return our custom error page.

Graphical user interface, text

Description automatically generated

Since [www.avengershq.com/test.html](http://www.avengershq.com/test.html) does not exist, it will serve a 404 HTTP status code in its response. Apache picks this up and returns our custom error page to the end user.

# Basic Authentication

Next, let’s secure our website using Basic Authentication (this is outdated and unsecure, so you’d want to add better authentication mechanisms if this was a real site).

Let’s say I have a user (not a Linux user, a web user) called nfury and they should log in to access the site via their password (oneeye).

1. Add a **.htpasswd** file to /var/www/avengershq.
   1. This file contains a list of users and encrypted passwords for those users.
   2. We also have to edit our existing .htaccess in order to turn on authentication.
2. Use the command: **sudo htpasswd -c .htpasswd nfury**
   1. This runs the htpasswd application.
   2. The -c option creates our passwd file.
   3. The .htpasswd parameter is the name of the file we are creating.
   4. Nfury is the user. We used the password “oneeye”

Text

Description automatically generated

1. If we reload the page, there is no difference in functionality. This is because we haven’t enabled authentication in our .htaccess file. Let’s do that now:

Add the following 5 new directives to the top of our .**htaccess** file:

1. AuthBasicProvider file
   * Tells us that accounts are stored in a file, not a database
2. AuthUserFile /var/www/avengershq/.htpasswd
   * Tells Apache that our users are located in the .htpasswd file in that directory
3. AuthName “…”
   * This is the message that will appear to the user on sign in
4. AuthType Basic
   * Tells Apache that this site uses Basic Authentication (username and password)
5. Require valid-user

Text

Description automatically generated

1. And now browse to our site: (note that we do NOT need to reload apache2 here as that is the whole reason for using .htaccess)

Graphical user interface, application

Description automatically generated

1. Once we enter a correct username/password, it will be cached, so to continue testing, we need to clear our cache in FireFox (**Ctrl-Shift-Del**):

Graphical user interface, application

Description automatically generated

1. Enter an incorrect username / password a few times to get kicked out: (or just press cancel!)

Graphical user interface, text, application, email

Description automatically generatedNotice that instead of 404 status code being returned on bad authentication, it’s returning 401.

We can create a custom 401 error page for this site, but first we need to add that directive to .htaccess:

Text

Description automatically generated

Text

Description automatically generatedWe’ve added a 401 directive to respond with a custom html document for all 401 errors. Let’s test it out (cancel):

Graphical user interface, text, application, email

Description automatically generated

Let’s add some additional users:

* sudo htpasswd .htpasswd srogers (brooklyn)
* sudo htpasswd .htpasswd tstark (luv9k)
* sudo htpasswd .htpasswd bbanner (strongest)
* sudo htpasswd .htpasswd todinson (pointbreak)
* sudo htpasswd .htpasswd nromonov (widow)
* sudo htpasswd .htpasswd barton (arrow)

Text

Description automatically generated

Text

Description automatically generated

Let’s say we only want the active roster for the Avengers to be able to access our site:

We can create a group (not a Linux group) and give access to that group

To do this, we’ll need to create another directive file called **.htgroup**:

Text

Description automatically generated

Before we can use it, we need to enable Group files in our .htaccess file:

Text

Description automatically generated

Let’s try it out!

Graphical user interface, text, application, email

Description automatically generated

We get a 500 error! That’s a server error, so we’ll need to investigate!

Opening up our dev tools and look at responses doesn’t really give us useful information:

We’ll have to look at the Apache Error logs to get any information on this.

Let’s take a look at the following logs: **/var/log/apache2/error.log**

Text

Description automatically generated

Looks like AuthGroupFile isn’t a recognized directive (either misspelled or not loaded) in our .htaccess file. I know that the spelling is correct, so it’s probably just a missing module in Apache.

We can add group authentication to Apache via the following command:

**sudo a2enmod authz\_groupfile**

With underscore. Afterwards, we’ll need to **restart** as well via restart command (use restart to enable new modules).

Text

Description automatically generated

Graphical user interface, application

Description automatically generated

Visiting our site again, we can test out our active accounts. Our active accounts work (members of the group “active”) (namely: nfury, bbanner, todinson, barton) but our other accounts to our .htpasswd file do not work, as they are not included in the “active” group.

Another thing you may want to add in your configuration is redirects. Open apache2.conf and add the following line:

**Redirect permanent /search** [**http://duckduckgo.com**](http://duckduckgo.com)

and test it out:

Text

Description automatically generated

When trying to load [www.avengershq.com/search](http://www.avengershq.com/search) we are now redirected:

Graphical user interface, text, application, email

Description automatically generated

Exercise 1: Create a virtual host for Shield which serves content from /var/www/shield. Put an index.html in there that says Top Secret Webpage for SHIELD. Make a 404 that says “We deny all knowledge of that”

Exercise 2: Create a virtual host for Asgard, which serves from /var/www/asgard. Make it the default host for people visiting your external IP address. Put an index.html there that says “Crossing the Rainbow bridge”. HINT: Grab your other ip address in your VM, use that in your hosts file for asgard.com. Also, add that same ip address to your asgard.conf in place of the wildcard “\*”