**Name of Title:** Learning Nginx

**Video Name:**

**Estimated Length:**

**Author Name:** Michael Jenkins

**Chapter\_Section\_Video:**

**Video Objective:**

At the end of this video the learner will be able to configure custom logs for the site.

**Introductory Statement:**

Type your introductory statement here.

**Speaking Points:**

1. What are logs?
2. Take a look at the current logs
   1. Explain that logs for all sites are written here
3. Take a look at the default config in /etc/nginx.conf
4. Edit wisdompetmed.local.conf
   1. Add custom error file
   2. Add custom access file
5. Check configuration
6. Restart server
7. Make some requests to the site
8. Review custom logs

**Script:**

Nginx uses log files to record various operational details. These logs are useful for monitoring normal operation and tracking down issues if a problem occurs.

GO TO TERMINAL

Let’s start by taking a look inside the main configuration file, nginx.conf.

vim /etc/nginx/nginx.conf

Inside this file, if we search for log we can find two entries: access log, and error log.

access\_log /var/log/nginx/access.log;

error\_log /var/log/nginx/error.log;

The access log is where nginx records each request that it serves, wether that request is served successfully or not. Each request can be recorded along with various metadata like the date and time of the request, the IP address of the requester, and the path to the document that was served.

Nginx uses the error log to store operational details ranging from debug information to warnings and critical errors.

If nginx is set up to serve multiple sites, but the default logging configuration is used, then all the requests, for all the sites will be written to the same logs. This can become an issue if, for example, you need to find the access logs for one specific site.

Fortunately, logging can be configured in the server and location contexts.

Let’s set up logging for our demo site.

First, we’ll edit our server configuration file:

vim /etc/nginx/conf.d/wisdompetmed.local.conf

Now we can add access log and error logs specifically for this virtual host with the access log and error log directives followed by the path to the file.

DON’T ADD SEMICOLONS HERE TO GET ERROR BELOW

access\_log /var/log/nginx/wisdompetmed.local.access.log

error\_log /var/log/nginx/wisdompetmed.local.error.log

And just for demonstration purposes, let's also add access and error logs in the images location:

access\_log /var/log/nginx/wisdompetmed.local.image.access.log;

error\_log /var/log/nginx/wisdompetmed.local.image.error.log;

Now let’s save this, test our configuration, and reload it:

nginx -t

nginx: [emerg] directive "access\_log" is not terminated by ";" in /etc/nginx/conf.d/wisdompetmed.local.conf:13

Bummer! Looks like we forgot to add semicolons after the new statements. Let’s fix that.

vim /etc/nginx/conf.d/wisdompetmed.local.conf

ADD SEMICOLONS TO THE LOG FILE STATEMENTS

And one more time, let’s test the configuration and reload it:

nginx -t

systemctl reload nginx

If we take a look at the log directory, we can see that our new log files have been created but they’re empty because no requests have come through.

cd /var/log/nginx/

ls -ltr

cat error.log

Instead of loading the page in the browser, let’s stay in the terminal and send a few requests to localhost using curl.

We can use a for loop to curl localhost 10 times while sending the output to dev null.

for i in {1..10}; do curl localhost > /dev/null; done

And now if we look inside the access log for our site, we can see 10 records that correspond to our 10 curls to the site:

cat wisdompetmed.local.access.log

We can do the same with the images location. Because we’re requesting a directory location, we need to add a trailing slash in this case.

Like before we can curl it 10 times:

for i in {1..10}; do curl localhost/images/ > /dev/null; done

And now if we check the contents of the images access log we can see the logs for each request:

cat wisdompetmed.local.image.access.log

Seeing the content in the access files for both of these locations lets us know that our custom logging directives are properly configured.

**Exercise Files:**

<https://docs.nginx.com/nginx/admin-guide/monitoring/logging/>

<https://www.keycdn.com/support/nginx-error-log/>

**Basement:**

### error\_log log\_file [ log\_level ]

### Logging Levels

The error\_log directive can be configured to log more or less information as required. The level of logging can be any one of the following:

* emerg: Emergency situations where the system is in an unusable state.
* alert: Severe situation where action is needed promptly.
* crit: Important problems that need to be addressed.
* error: An Error has occurred. Something was unsuccessful.
* warn: Something out of the ordinary happened, but not a cause for concern.
* notice: Something normal, but worth noting has happened.
* info: An informational message that might be nice to know.
* debug: Debugging information that can be useful to pinpoint where a problem is occurring.

The second parameter determines the *level* of logging, and can be one of the following: debug, info, notice,warn, error, crit, alert, or emerg. Log levels above are listed in the order of increasing severity. Setting a certain log level will cause all messages of the specified and more severe log levels to be logged. For example, the default level error will cause error, crit, alert, and emerg messages to be logged. If this parameter is omitted then error is used.

FOR ACCESS LOGS

The configuration always includes the predefined “combined” format:

log\_format combined '$remote\_addr - $remote\_user [$time\_local] '  
 '"$request" $status $body\_bytes\_sent '  
 '"$http\_referer" "$http\_user\_agent"';

NEED SLIDE TO DISCUSS THE ERROR LEVELS. SO SHOULD PROBABLY USE SLIDE FOR BOTH LOG TYPES.