How will you solve these common problems when a site is not working correctly?

Check the Logs

Before blindly trying to track down a problem, try to check the logs of your web server and any related components. These will usually be in `/var/log` in a subdirectory specific to the service. For instance, if you have an Apache server running on an Ubuntu server, by default the logs will be kept in `/var/log/apache2`. Check the files in this directory to see what kind of error messages are being generated. If you have a database backend that is giving you trouble, that will likely keep its logs in `/var/log` as well.
Other things to check are whether the processes themselves leave you error messages when the services are started. If you attempt to visit a web page and get an error, the error page can contain clues too (although not as specific as the lines in the log files).

Use a search engine to try to find relevant information that can point you in the right direction. The steps below can help you troubleshoot further.

Is your Web Server Installed?

The first thing you need to serve your websites correctly is a web server.

Most people will have installed a server before getting to this point, but there are some situations where you may have actually accidentally uninstalled the server when performing other package operations.

If you are on an Ubuntu or Debian system and wish to install the Apache web server, you can type:

```
sudo apt-get update
sudo apt-get install apache2
```

On these systems, the Apache process is called **apache2**.

If you are running Ubuntu or Debian and want the Nginx web server, you can instead type:

```
sudo apt-get update
sudo apt-get install nginx
```

On these systems, the Nginx process is called **nginx**.

If you are running CentOS or Fedora and wish to use the Apache web server, you can type this. You can remove the "sudo" if you are logged in as root:

```
sudo yum install httpd
```

On these systems, the Apache process is called **httpd**.

If you are running CentOS or Fedora and want to use Nginx, you can type this. Again, remove the "sudo" if you are logged in as root:

```
sudo rpm -Uvh http://download.fedoraproject.org/pub/epel/6/i386/epel-release-6-8.noarch.rpm
sudo yum install nginx
```

On these systems, the Nginx process is called **nginx**.

Is your Web Server Running?

Now that you are sure your server is installed, is it running?

There are plenty of ways of finding out if the service is running or not. One method that is fairly cross-platform is to use the `netstat` command.
This will tell you all of the processes that are using ports on the server. We can then `grep` for the name of the process we are looking for:

```
sudo netstat -plunt | grep apache2
tcp6       0      0 :::80           :::*            LISTEN
2000/apache2
```

You should change "apache2" to the name of the web server process on your server. If you see a line like the one above, it means your process is up and running. If you don't get any output back, it means you queried for the wrong process or that your web server is not running.

If this is the case, you can start it using your distribution's preferred method. For instance, on Ubuntu, you could start the Apache2 service by typing:

```
sudo service apache2 start
```

On CentOS you might type something like this:

```
sudo /etc/init.d/httpd start
```

If your web server starts, you can check with `netstat` again to verify that everything is correct.

Is the Syntax of your Web Server Configuration File Correct?

If your web server refused to start, often this is an indication that your configuration files need some attention. Both Apache and Nginx require strict adherence to their directive syntax in order for the files to be read.

The configuration files for these services are usually located within a subdirectory of the `/etc/` directory named after the process itself.
Thus, we could get to the main configuration directory of Apache on Ubuntu by typing:

```
cd /etc/apache2
```

In a similar way, the Apache configuration directory on CentOS also mirrors the CentOS name for that process:

```
cd /etc/httpd
```

The configuration will be spread out among many different files. If your service failed to start, it will usually point you to the configuration file and the line where the problem was first found. Check that file for errors.

Each of these web servers also provide you with the ability to check the configuration syntax of your files.

If you are using Apache, you can use the `apache2ctl` or `apachectl` command to check your configuration files for syntax errors:

```
apache2ctl configtest
AH00558: apache2: Could not reliably determine the server's fully
qualified domain name, using 127.0.0.1. Set the 'ServerName' directive
globally to suppress this message
Syntax OK
```

As you can see above, we've received an informational message about a detail in our configuration, but there were no errors. This is good.

If you have an Nginx web server, you can run a similar test by typing:

```
sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

As you can see, this process checks your syntax as well. If we remove an ending semi-colon from a line in the file (a common error for Nginx configurations), you would get a message like this:

```
sudo nginx -t
nginx: [emerg] invalid number of arguments in "tcp_nopush" directive in
/etc/nginx/nginx.conf:18
nginx: configuration file /etc/nginx/nginx.conf test failed
```

There is an invalid number of arguments because Nginx looks for a semi-colon to end statements. If it doesn't find one, it drops down to the next line and interpret that as further arguments for the last line.

Are the Ports you Configured Open?

Generally, web servers run on port 80 for normal web traffic and use port 443 for traffic encrypted with TLS/SSL. In order for you to reach the site correctly, these ports must be made accessible.

You can test whether your server has its port open by using `netcat` from your local machine. You just need to use your server's IP address and tell it what port you wish to check, like this:

```
sudo nc -z 111.111.111.111 80
```

This will check whether port 80 is open on the server at `111.111.111.111`. If it is open, the command will return right away. If it is **not** open, the command will continuously try to form a connection, unsuccessfully. You can stop this process by hitting CTRL-C in the terminal window.
If your web ports is not accessible, you should look at your firewall configuration. You may need to open up port 80 or port 443.

Are your DNS Settings Directing you to the Correct Place?

If you can reach your site through the IP address, but not through the domain name, you may need to take a look at your DNS settings.

In order for visitors to reach your site through its domain name, you should have an "A" or "AAAA" record pointing to your server's IP address in the DNS settings. You can query for your domain's "A" record by running this command:

```
host -t A example.com
```

```
example.com has address 93.184.216.119
```

The line that is returned to you should match the IP address of your server. If you need to check an "AAAA" record (for IPv6 connections), you can type:

```
host -t AAAA example.com

example.com has IPv6 address 2606:2800:220:6d:26bf:1447:1097:aa7
```

Keep in mind that any changes you make to the DNS records will take quite a long time to propagate. You may receive inconsistent results to these queries after a change since your request will often hit different servers that are not all up-to-date yet.

If you are using DigitalOcean, you can learn how to configure DNS settings for your domain here.

Make sure your Configuration Files Also Handle your Domain Correctly

If your DNS settings are correct, you may also want to check your Apache virtual host files or the Nginx server block files to make sure they are configured to respond to requests for your domain.

In Apache, the section of your virtual host file might look like this:

```
<VirtualHost *:80>

    ServerName example.com

    ServerAlias www.example.com

    ServerAdmin admin@example.com

    DocumentRoot /var/www/html

. . .
```

This virtual host is configured to respond to any requests on port 80 for the domain `example.com`.

A similar chunk in Nginx might look something like this:

```
server {

    listen 80 default_server;

    listen [::]:80 default_server ipv6only=on;

    root /usr/share/nginx/html;

    index index.html index.htm;

    server_name example.com www.example.com;

. . .
```

This block is configured to respond to the same type of request that we discussed above.

Does the Document Root Point to the Location of your Files?

Another consideration is whether your web server is pointed at the correct file location.

Each virtual server in Apache or server block in Nginx is configured to point to a specific directory. If this is configured incorrectly, the server will throw an error message when you try to access the page.

In Apache, the document root is configured through the `DocumentRoot` directive:

```
<VirtualHost *:80>

    ServerName example.com

    ServerAlias www.example.com

    ServerAdmin admin@example.com
```

```
    DocumentRoot /var/www/html

. . .
```

This line tells Apache that it should look for the files for this domain in the `/var/www/html` directory. If your files are kept elsewhere, you'll have to modify this line to point to the correct location.

In Nginx, the `root` directive configures the same thing:

```
server {

    listen 80 default_server;

    listen [::]:80 default_server ipv6only=on;

    root /usr/share/nginx/html;

    index index.html index.htm;

    server_name example.com www.example.com;

. . .
```

In this configuration, Nginx looks for files for this domain in the `/usr/share/nginx/html` directory.

Is your Web Server Serving the Correct Index Files?

If your document root is correct and your index pages are not being served correctly when you go to your site or a directory location on your site, you may have your indexes configured incorrectly.

When a visitor requests a directory, typically your server will want to give them an index file. This is usually an `index.html` file or an `index.php` file depending on your configuration.

In Apache, you may find a line in your virtual host file that configures the index order that will be used for specific directories explicitly, like this:

```
<Directory /var/www/html>

    DirectoryIndex index.html index.php

</Directory>
```

This means that when the directory is being served, Apache will look for a file called `index.html` first, and try to serve `index.php` as a backup if the first file is not found.
You can set the order that will be used to serve index files for the entire server by editing the `mods-enabled/dir.conf` file, which will set the defaults for the server. If your server is not serving an index file, make sure you have an index file in your directory that matches one of the options in your file.
In Nginx, the directive that does this is called `index` and it is used like this:

```
server {

    listen 80 default_server;

    listen [::]:80 default_server ipv6only=on;

    root /usr/share/nginx/html;

    index index.html index.htm;

    server_name example.com www.example.com;

. . .
```

Are the Permissions and Ownership Set Correctly?

In order for the web server to correctly serve files, it must be able to read the files and have access to the directories where they are kept. This can be controlled through the file and directory permissions and ownership.

To read files, the directories containing the content must be readable and executable by the web server process. User and group name that are used to run the web server differ from distribution to distribution.

On Ubuntu and Debian, both Apache and Nginx run as the user `www-data` which is a member of the`www-data` group.
On CentOS and Fedora, Apache runs under a user called `apache` which belongs to the `apache`group. Nginx runs under a user called `nginx` which is a part of the `nginx` group.
Using this information, you can look at the directories and files that make up your site content:

```
ls -l /path/to/web/root
```

The directories should be readable and executable by the web user or group, and the files should be readable in order to read content. In order to upload, write or modify content, the directories must additionally be writeable and the files need to be writeable as well. Set directories to be writable with great caution though, because it can be a security risk.

To modify the ownership of a file, you can do this:

```
sudo chown user_owner:group_owner /path/to/file
```

This can also be done to a directory. You can change the ownership of a directory and all of the files under it by passing the `-R` flag:

```
sudo chown -R user_owner:group_owner /path/to/file
```

You can learn more about Linux permissions here.

Are you Restricting Access through your Configuration Files?

It's possible that your configurations are set up to deny access from the files you are trying to serve.

In Apache, this would be configured in the virtual host file for that site, or through an `.htaccess` file located in the directory itself.
Within these files, it is possible to restrict access in a few different ways. Directories can be restricted like this in Apache 2.4:

```
<Directory /usr/share>

    AllowOverride None

    Require all denied

</Directory>
```

This line is telling the web server not to let anybody access the contents of this directory. In Apache 2.2 and below, this would be written like this:

```
<Directory /usr/share>

    AllowOverride None

    Order deny,allow

    Deny from all

</Directory>
```

If you find a directive like this for the directory containing the content you are trying to access, this will prevent your success.

In Nginx, these restrictions will take the form of `deny` directives and will be located in your server blocks or main config files:

```
location /usr/share {

    deny all;

}
```

If you have a Database Backend, is it Running?

If your site relies on a database backend like MySQL, PostreSQL, MongoDB, etc. you need to make sure that it is up and running.

You can do that in the same way that you checked that the web server was running. Again, we can search for the running processes and then pick out the name of the process we are looking for:

```
sudo netstat -plunt | grep mysql


tcp        0      0 127.0.0.1:3306        0.0.0.0:*          LISTEN
3356/mysqld
```

As you can see, the service is running on this machine. Make sure you know the name that your service runs under when searching for it.

An alternative is to search for the port that your service runs on if you know that. Look at the documentation for your database to find the default port that it runs on or check your configuration files.

If you have a Database Backend, can your Site Connect Successfully?

The next step to take if you are troubleshooting an issue with a database backend is to see if you can connect correctly. This usually means checking the files that your site reads to find out the database information.

For instance, for a WordPress site, the database connection settings are stored in a file called `wp-config.php`. You need to check that the `DB_NAME`, `DB_USER`, and `DB_PASSWORD` are correct in order for your site to connect to the database.
You can test whether the file has the correct information by trying to connect to the database manually using something like:

```
mysql -u DB_USER_value -pDB_PASSWORD_value DB_NAME_value
```

If you cannot connect using the values you found in the file, you may need to create the correct accounts and databases or modify the access permissions of your databases.

Is your Web Server Configured to Pass Dynamic Content to a Script Processor?

If you are using a database backend, you almost certainly are using a programming language like PHP to process request for dynamic content, fetch the information from the database, and render the results.

If this is the case, you need to make sure that your web server is configured correctly to pass requests to the script processor.

In Apache, this generally means making sure that `mod_php5` is installed and enabled. You can do that on Ubuntu or Debian by typing:

```
sudo apt-get update
sudo apt-get install php5 libapache2-mod-php5
sudo a2enmod php5
```

For CentOS/Fedora systems, you'll have to type:

```
sudo yum install php php-mysql
sudo service httpd restart
```

In Nginx, this is a bit more complicated. Nginx doesn't have a PHP module that can be enabled, so we need make sure that we have `php-fpm` installed and enabled in our configurations.
On an Ubuntu or Debian server, ensure that the components are installed by typing:

```
sudo apt-get update
sudo apt-get install php5-fpm php5-mysql
```

On CentOS or Fedora, you'd do this by typing:

```
sudo yum install php-fpm php-mysql
```

Since the PHP processor isn't a part of Nginx, you need to tell it to pass the PHP files explicitly. Follow step four of this guide to learn how to configure Nginx to pass PHP files to php-fpm. You should also take a look at the portion of step three that deals with configuring the PHP processor. This should be pretty much the same regardless of your distribution.

If all else Fails, Check the Logs Again

Checking the logs should actually be your first step, but it is also a good last step prior to asking for more help.

If you have reached the end of your ability to troubleshoot on your own and need some help (from a friend, by opening a support ticket, etc.), you will get more relevant help faster by providing log files and error messages. Experienced administrators will probably have a good idea of what is happening if you give them the pieces of information that they need.

Conclusion

Hopefully these troubleshooting tips will have helped you track down and fix some of the more common issues that administrators face when trying to get their sites up and running.

How to Find Web Server Type Running On a Linux or Windows Machine?

```
#curl -I www.linuxnix.com
# curl -I itig

HTTP/1.1 200 OK
Via: 1.1 us-PROXY1, 1.1 us-PROXY2
Connection: close
Proxy-Connection: close
Expires: Mon, 1 Jan 2001 00:00:00 GMT
Date: Mon, 14 Jun 2010 03:26:55 GMT
Content-Type: text/html; charset=utf-8
Server: Apache/2.2.3 (Red Hat)
X-Powered-By: PHP/5.1.6
Set-Cookie: b2933756b1f0288a74dd630e51954bff=5sl7jaqpgliambi2fq7q3ailv6;
path=/
P3P: CP="NOI ADM DEV PSAi COM NAV OUR OTRo STP IND DEM"
```

Last-Modified: Mon, 14 Jun 2010 03:26:55 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
In the above output we can see server as **Apache 2.2.3.**

How To See Total Processes Are Running, Files Opened, Memory Used By A User?

To see all the process run by a particular user
#ps -u username u
-u used to specify user name, and u at the end of the command shows you complete path of the command executed.
Example
#ps -u rajesh u
or

To see all the files opened by user
#lsof | grep username

Example
#lsof | grep rajesh
To see memory used by a particular user
#ps -u username u | awk '{print $2}'

Example
ps -u krishna u | awk '{print $2}'
Now you will get all the PID's run by user Krishna. So try to get memory usage by using **pmap** command.

How to Access the Web Server through a Hostname?

Step1: Check for the package installed or not
#rpm -qa | grep httpd
Step2 : Install the package
#yum install httpd
Step3 : For hostbased web access we require DNS server entries of our host.
Example host in this post is : server1.example.com(Actual server name) and **hostbase.example.com(our virtual host name)**
So we have to give **A record** for the **server1.example.com** and **CNAME record** for **hostbase.example.com** which points to**server1.example.com**
Step3 : Create the root web directory and a test **index.html** for this testing.
I want to create my web directory in **/hostnamebase**

#mkdir /hostnamebase
Now create **index.html** and try to write something in to that file then save the file.
#vi /hostnamebase/index.html
save and exit the file.

Step4: Now edit the **httpd.config** file
#vi /etc/httpd/conf/httpd.conf
Step4 (a): Please specify the **NameVirtualHost**
NameVirtualHost 192.168.0.1
What is the significance of this entry?
Ans: This directive is a set to IP address/actual system name if you want to configure your server to host. Name based virtual hosting. So here I am using ip address of my system. You can try to put actual hostname if you want to test it.
Step4 (b): Now configure virtual host entry as shown below.
Go to last line and write the below content
<VirtualHost 192.168.0.1>
ServerName hostbase.example.com
DocumentRoot /hostnamebase/
DirectoryIndex index.html
</VirtualHost>
Save the file and exit.

Let me explain **ServerName** entry
ServerName **hostbase.example.com** This is used to specify what is the virtual host name of your new **Virtualhost**. So user can type this **FQDN** to access this site.

Now save the file and exit.

Step5: Check for the syntax errors in the **httpd.conf** file before restarting the apache service.
#httpd -t
Or
#httpd -k graceful
Step6: Now start the service and then add it to booting scripts so that it will start automatically at every boot of the system

#service httpd restart
#chkconfig httpd on
Step7: Now check the site using **http://hostbase.example.com** in your browser.

How to Assign or Change Hostname in Linux and Unix?

Computers are assigned with a name as we do for human beings. This is called hostname in computer world. So how to assign a hostname to a Linux system? It's very much simple. Before doing this we should know every task in Linux can be done in two ways:

1. Temporary way (once you boot your system the changes made in this way will go away)
2. Permanent way (Changes made in this way will be retained even after rebooting of the system).
Before changing the hostname, check what is your present system name by using hostname command
#hostname
This command will show present system name.
Now we will see how to assign a host name to a Linux machine in both(temporary as well as permanent) ways.

Temporary way: Use **hostname** command to change the host name
#hostname your-system-nameExample:
#hostname office-laptop.example.com
Permanent way : Change hostname in Redhat/CentOS/Fedora:
Edit **/etc/sysconfig/network file**, use **HOSTNAME** variable in that file to denote hostname.
#vi /etc/sysconfig/network
HOSTNAME=office-laptop.example.com

Save the file and exit

Then your hostname is changed permanently to your desired name.
Change hostname in **Ubuntu/Debian**:

Edit /etc/hostname, just write down what is your hostname in to that file.
#vi /etc/hostname
office-laptop.example.com