Apache Server.

NFS Server

FTP Server

MariaDB Server

What is Apache Web Server?

Apache HTTP Server is a free and open-source web server that delivers web content through the internet. It is commonly referred to as Apache and after development, it quickly became the most popular HTTP client on the web. It's widely thought that Apache gets its name from its development history and process of improvement through applied patches and modules but that was corrected back in chkconfig httpd on 2000. It was revealed that the name originated from the respect of the Native American tribe for its resiliency and durability.

Now, before we get too in depth on Apache, we should first go over what a web application is and the standard architecture usually found in web apps.

Working of Apache

Apache is not any physical server; it is software that executes on the server. However, we define it as a web server. Its objective is to build a connection among the website visitor browsers (Safari, Google Chrome, Firefox, etc.) and the server. Apache can be defined as cross-platform software, so it can work on Windows servers and UNIX.

Apache web server port number is 80.

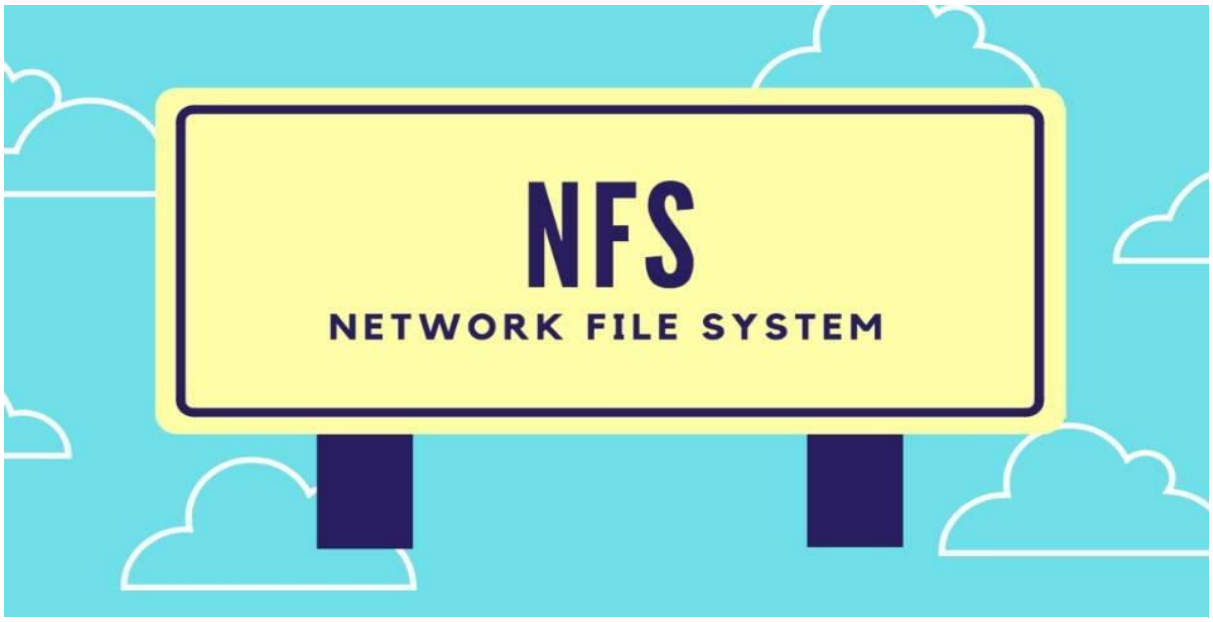
Install and Configure Apache HTTP Server in Red Hat Enterprise Linux

yum install httpd

chkconfig httpd on

system-config-services

service httpd start



Linux provides several tools for accessing files on remote systems connected to network.

The Network File System (NFS) enables you to connect to and directly access resources such as files or devices like CD-ROMs that reside on another machine. The new version, NFS4, provides greater security, with access allowed by your firewall. The Network Information Service (NIS) maintains configuration files for all systems on a network.

Network File Systems: NFS and /etc/exports

NFS enables you to mount a file system on a remote computer as if it were local to your own system. You can then directly access any of the files on that remote file system. This has the advantage of allowing different systems on a network to access the same files directly, without each having to keep its own copy. Only one copy will be on a remote file system, which each computer can then access.

Website for more details about NFS: nfs.sourceforge.net

NFS Services

Its a **System V-launched** service. The **NFS** server package includes three facilities, included in the **portmap** and **nfs-utils** packages.

- **portmap** : It maps calls made from other machines to the correct **RPC** service (not required with **NFSv4**).
- **nfs**: It translates remote **file sharing** requests into requests on the local file system.
- **rpc.mountd**: This service is responsible for **mounting** and **unmounting** of file systems

Important Files for NFS Configuration

- **/etc/exports** : Its a main configuration file of **NFS**, all exported **files** and **directories** are defined in this file at the **NFS Server** end.
- **/etc/fstab** : To mount a **NFS directory** on your system across the **reboots**, we need to make an entry in **/etc/fstab**.
- **/etc/sysconfig/nfs** : Configuration file of **NFS** to control on which port **rpc** and other services are **listening**.

Setup NFS (Network File System)

To setup **NFS, We need at least two servers.**

NFS Server- krishanblog.example.com with IP-**192.188.100.900**

NFS Client- krishantech.example.com with IP-**192.188.100.901**

Installing NFS Server

We need to install **NFS** packages on our **NFS Server** as well as on **NFS Client** machine.

```
[root@krishanblog ~]# yum install nfs-utils nfs-utils-lib

[root@krishanblog ~]# yum install portmap (not required
with NFSv4)

[root@krishanblog ~]# yum install nfs-utils nfs-utils-lib
```

Now start the **services** on both machines.

```
[root@krishanblog ~]# /etc/init.d/portmap start

[root@krishanblog ~]# /etc/init.d/nfs start

[root@krishanblog ~]# chkconfig --level 35 portmap on

[root@krishanblog ~]# chkconfig --level 35 nfs on
```

After installing packages and starting services on both the machines, we need to configure both the machines for file sharing.

Setting Up the NFS Server

First we will be configuring the **NFS** server.

Configure Export directory

For sharing a directory with **NFS**, we need to make an entry in "**/etc/exports**" configuration file. Here I'll be creating a new directory named "**nfsshare**" in "**/**" partition to share with **client server**.

```
[root@krishanblog ~]# mkdir /myfiles
```

Now we need to make an entry in "**/etc/exports**" and **restart** the services to make our directory shareable in the network.

```
[root@krishanblog ~]# vi /etc/exports

/myfiles 192.188.100.900(rw,sync,no_root_squash)
```

NFS Options

Some other options we can use in "**/etc/exports**" file for file sharing is as follows.

- **ro**: With the help of this option we can provide **read only access** to the shared files i.e **client** will only be able to **read**.
- **rw**: This option allows the **client server** to both **read** and **write** access within the shared directory.
- **sync**: Sync confirms requests to the shared directory only once the **changes** have been committed.
- **no_subtree_check**: This option prevents the **subtree** checking. When a shared directory is the subdirectory of a larger file system, **nfs** performs scans of every directory above it, in order to verify its permissions and details. Disabling the **subtree** check may increase the reliability of **NFS**, but reduce **security**.
- **no_root_squash**: This phrase allows **root** to **connect** to the designated directory.

For more options with "**/etc/exports**", you are recommended to read the **man pages** for **export**.

Setting Up the NFS Client

After configuring the **NFS** server, we need to **mount** that shared directory or partition in the **client** server.

Mount Shared Directories on NFS Client

Now at the **NFS client** end, we need to **mount** that directory in our server to access it locally. To do so, first we need to find out that shares available on the remote server or NFS Server.

```
[root@krishantech ~]# showmount -e 192.188.100.900



Export list for 192.188.100.900:

/myfiles 192.188.100.901
```

Above command shows that a directory named "**myfiles**" is available at "**192.188.100.900**" to share with your server.

Mount Shared NFS Directory

To **mount** that shared **NFS** directory we can use following mount command.

```
[root@krishantech ~]# mount -t nfs
192.188.100.900:/myfiles /mnt/myfiles
```

The above mount command mounted the **nfs shared directory** on to **nfs client** temporarily, to mount an NFS directory **permanently** on your system across the **reboots**, we need to make an entry in "**/etc/fstab**".

```
[root@krishantech ~]# vi /etc/fstab
```

Add the following new line as shown below.

```
192.188.100.900:/myfiles /mnt  nfs defaults 0 0
```

Important commands for NFS

Some more important commands for **NFS**.

- **showmount -e** : Shows the available **shares** on your local machine
- **showmount -e <server-ip or hostname>**: Lists the available **shares** at the **remote** server
- **showmount -d** : Lists all the **sub directories**
- **exportfs -v** : Displays a list of shares **files** and **options** on a server
- **exportfs -a** : Exports all shares listed in **/etc/exports**, or given name
- **exportfs -u** : Unexports all shares listed in **/etc/exports**, or given name
- **exportfs -r** : Refresh the server's list after modifying **/etc/exports**

What is FTP?

- FTP stands for File transfer protocol.
- FTP is a standard internet protocol provided by TCP/IP used for transmitting the files from one host to another.
- It is mainly used for transferring the web page files from their creator to the computer that acts as a server for other computers on the internet.
- It is also used for downloading the files to computer from other servers.

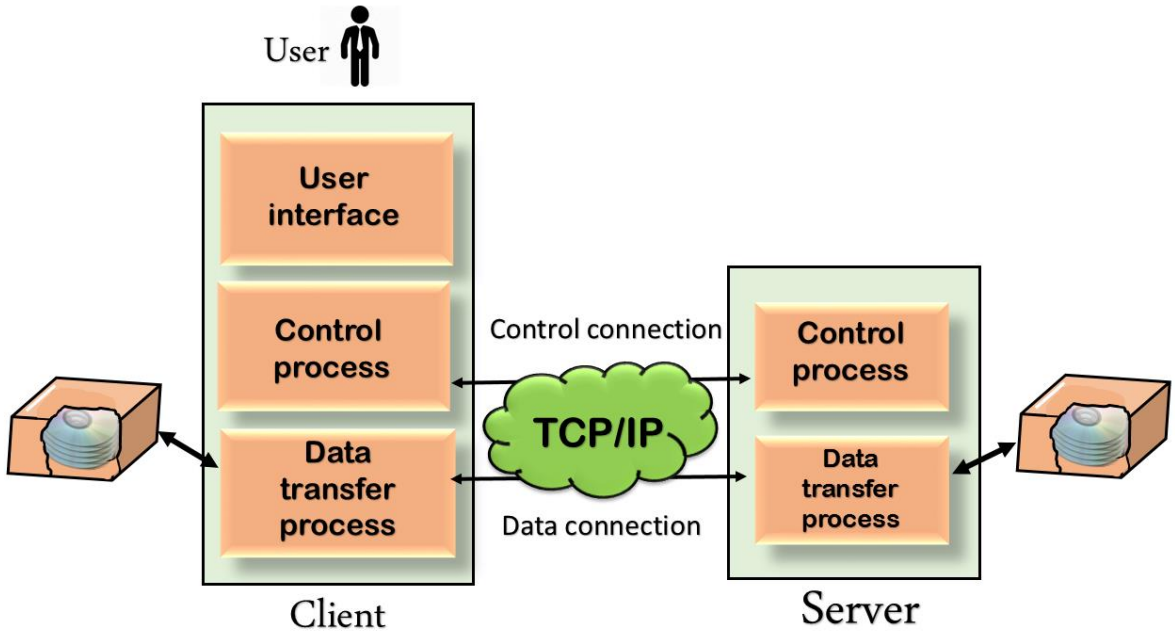
Objectives of FTP

- It provides the sharing of files.
- It is used to encourage the use of remote computers.
- It transfers the data more reliably and efficiently.

Why FTP?

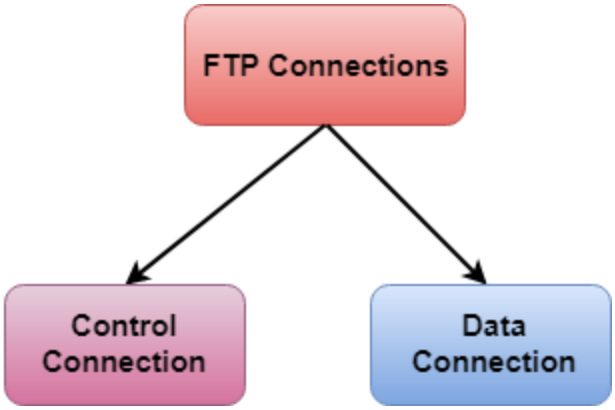
Although transferring files from one system to another is very simple and straightforward, but sometimes it can cause problems. For example, two systems may have different file conventions. Two systems may have different ways to represent text and data. Two systems may have different directory structures. FTP protocol overcomes these problems by establishing two connections between hosts. One connection is used for data transfer, and another connection is used for the control connection.

Mechanism of FTP



The above figure shows the basic model of the FTP. The FTP client has three components: the user interface, control process, and data transfer process. The server has two components: the server control process and the server data transfer process.

There are two types of connections in FTP:



- **Control Connection:** The control connection uses very simple rules for communication. Through control connection, we can transfer a line of command or line of response at a time. The control connection is made between the control processes. The control connection remains connected during the entire interactive FTP session.
- **Data Connection:** The Data Connection uses very complex rules as data types may vary. The data connection is made between data transfer processes. The data connection opens when a command comes for transferring the files and closes when the file is transferred.

FTP Clients

- FTP client is a program that implements a file transfer protocol which allows you to transfer files between two hosts on the internet.
- It allows a user to connect to a remote host and upload or download the files.
- It has a set of commands that we can use to connect to a host, transfer the files between you and your host and close the connection.
- The FTP program is also available as a built-in component in a Web browser. This GUI based FTP client makes the file transfer very easy and also does not require to remember the FTP commands.

Advantages of FTP:

- **Speed:** One of the biggest advantages of FTP is speed. The FTP is one of the fastest way to transfer the files from one computer to another computer.
- **Efficient:** It is more efficient as we do not need to complete all the operations to get the entire file.
- **Security:** To access the FTP server, we need to login with the username and password. Therefore, we can say that FTP is more secure.

- **Back & forth movement:** FTP allows us to transfer the files back and forth. Suppose you are a manager of the company, you send some information to all the employees, and they all send information back on the same server.

Stepwise Implementation

At first SSH into your Linux virtual machine with a user who has sudo permissions and follows the following steps:

Step 1: Install FTP server

There are many FTP servers to choose from like ProFTPD, vsftpd, etc. We will be using vsftpd.

Features of vsftpd FTP server

vsftpd has a lot of features that make it a great option as an FTP server. It

- Supports SSL/TLS integration
- Can jail users into its home directory with a feature called chroot. We will set this up later in this article.
- Can limit bandwidth.
- Supports virtual users
- Supports virtual IP configuration
- Supports IPv6

Type in the following command to install vsftpd

```
sudo apt install vsftpd
```

Now we will check if the vsftpd service is active or not. Type in

```
sudo systemctl status vsftpd
```

```
jivendra@dovm1:~$ sudo systemctl status vsftpd
● vsftpd.service - vsftpd FTP server
     Loaded: loaded (/lib/systemd/system/vsftpd.service; enabled; vendor>
     Active: active (running) since Fri 2021-12-17 13:25:40 UTC; 18s ago
   Main PID: 15094 (vsftpd)
      Tasks: 1 (limit: 1136)
     Memory: 596.0K
     CGroup: /system.slice/vsftpd.service
             └─15094 /usr/sbin/vsftpd /etc/vsftpd.conf

Dec 17 13:25:40 dovm1 systemd[1]: Starting vsftpd FTP server...
Dec 17 13:25:40 dovm1 systemd[1]: Started vsftpd FTP server.
jivendra@dovm1:~$
```

You can see under the Active heading that it's active and running. **systemctl** command is used to manage and check services on Linux. We can also use this command to enable and disable services on Linux. If your vsftpd is not active, then type in

```
sudo systemctl enable --now vsftpd
```

The –now flag ensures that enable command affects our service immediately and not after a reboot.

Step 2: Configure Firewall

FTP uses port 20 for active mode, port 21 for commands, and a range of ports for passive mode. We need to open these ports from our firewall. If you do not use any firewall, you can skip this step. Most Linux systems use ufw to manage firewalls, however, some cloud service providers like Microsoft Azure have firewalls outside of the Virtual machine and you have to configure that from their portal. Whatever the case, just open ports 20 and 21 for TCP connections and open a range of ports for passive FTP connections. The range for passive ports depends upon how many concurrent user clients you expect to have. Also, a single client can use multiple ports to transfer multiple files or a large file. We also need to specify our FTP server to use those ports and we will see how to do it later in this tutorial The ports till 1024 are reserved and our passive FTP port range should be higher than that. I'll open ports from 5000-10000. We will also open port 990 for TLS which we will configure later. Let's do it for ufw. Type in

```
sudo ufw allow 20/tcp
sudo ufw allow 21/tcp
sudo ufw allow 990/tcp
sudo ufw allow 5000:10000/tcp
```

Step 3: Configure Users

The two most common use cases of FTP servers are:

- You want to host a public FTP server and a lot of public users are going to connect to your FTP server to download files.
- You want to upload your files to your Linux server for personal use and you would not have public users.

In the first case, you would need to create an additional user and share its username and password with your clients to access the files. Everything else is the same for the second case.

The basic idea is that the admin user should be able to upload files to any folder of the machine, and the public user should be able to view and download files from a specific directory only. To make this happen, you should have a basic idea of user permissions. The root user has the permission to write files into any folder of the server, and any other user has access to every folder inside their home directory which is **/home/username** , and most of the other directories are not writable by other users. So if you want to upload files to other directories outside of your admin user's home directory, let's say **/var/www**, then you would need to change the owner of this directory to your admin user with **chown** command, or change directory modification permissions with **chmod** command.
Let's start by creating our public user account. Type in

```
sudo adduser ftpuser
```

Enter your password, leave other values empty, and at last, enter Y to save changes.

```
jivendra@dovm1:~$ sudo adduser ftpuser
Adding user `ftpuser' ...
Adding new group `ftpuser' (1001) ...
Adding new user `ftpuser' (1001) with group `ftpuser' ...
Creating home directory `/home/ftpuser' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for ftpuser
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] Y
jivendra@dovm1:~$
```

Now, for security purposes, we will disable ssh permission for this user. Type in

```
sudo nano /etc/ssh/sshd_config
```

Add the following line in this file

```
DenyUsers ftpuser
```

Press **Ctrl + x** then **y** then **enter.** Now, restart the SSH service so that these new settings take effect.

```
sudo systemctl restart sshd
```

Step 4: Create the FTP folder and set permissions

We will create our FTP folder. Type in

```
sudo mkdir /ftp
```

Now, we will change this directory's owner to our admin user. Type in

```
sudo chown adminuser /ftp
```

If you want to upload files to any folder that is not owned by your admin user, you will have to change that folder's owner using the above-mentioned command.

Step 5: Configure and secure vsftpd

Open the vsftpd configuration file. Type in

```
sudo nano /etc/vsftpd.conf
```

Make sure the following lines are uncommented

```
...
anonymous_enable=NO


local_enable=YES


write_enable=YES
...
```

Also, we opened ports 5000 to 10000 in step 2 for passive mode, so now we will let vsftpd know which ports to use for passive FTP connection. Add the following lines in vsftpd.conf file

```
pasv_min_port=5000
```

```
pasv_max_port=10000
```

Now, we will specify the default directory for FTP connections which will open when someone connects to our FTP server. Add the following line

```
local_root=/ftp
```

Remember, do not put any space before and after = in this configuration file.

Locking user into the home directory

Now, for security reasons, we will lock the ftpuser to the default directory, as by default, a user can browse the whole Linux server. To do this, vsftpd uses chroot. To do this, un-comment the following lines

```
...
chroot_local_user=YES


chroot_list_enable=YES


chroot_list_file=/etc/vsftpd.chroot_list
...
```

Also, add the following line as it is not in the configuration file by default

```
allow_writeable_chroot=YES
```

The first line enables chroot feature for local users which includes our admin user and our ftpuser. The second and third lines let us choose which users to apply to chroot to.

Setting file permission

```
local_umask=0002
```

This line will set the modification permission of every new file created to 664(-rw-rw-r-) and of every new folder to 775(rwxrwxr-x). With this, the ftpuser can only read and download files from every sub-directory of our FTP directory, but it does not have permission to upload anything to our FTP directory since it is not the owner.

Press **Ctrl + x** then **y** then **enter.** Now, we need to create that list file. Type in

```
sudo touch /etc/vsftpd.chroot_list
```

```
sudo nano /etc/vsftpd.chroot_list
```

Whatever users you specify in this file, will not be chroot-ed. So add your admin username in this file because we do not want to lock it. Press **Ctrl + x** then **y** then **enter**. Now we need to restart our vsftpd server so that all these settings get applied immediately. Type in

```
sudo systemctl restart --now vsftpd
```

MairaDB

MariaDB, a fork of MySQL is one of the most popular open-source SQL (Structured Query Language) relational databases management systems, made by the original developers of **MySQL**. It is designed for speed, reliability, and ease of use.

It is the default **MySQL** type database system in the standard repositories of most if not all major Linux distributions including **RHEL** (**RedHat Enterprise Linux**) and **Fedora Linux**. It also works on Windows and macOS, and many other operating systems. It is used as a replacement for **MySQL** database system in the **LAMP** (**Linux + Apache + MariaDB + PHP**) and **LEMP** (**Linux + Engine-X + MariaDB + PHP**) stack.

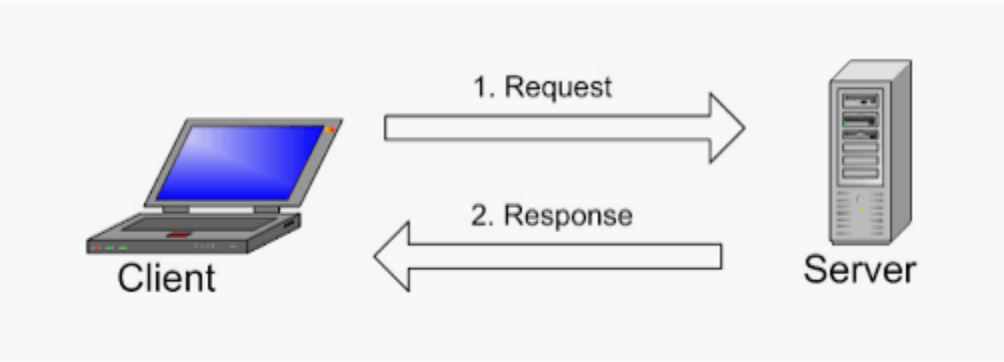
It's development started due to concerns that arose when **MySQL** was acquired by **Oracle Corporation** in 2009. Now, the developers and maintainers of **MariaDB** do monthly merges with the **MySQL** code base to ensure that **MariaDB** has any relevant bug fixes added to MySQL.

MariaDB server is available under the GPL license, version 2, and its client libraries for C, Java, and ODBC are distributed under the LGPL license, version 2.1 or higher. It is offered in two different editions.

The first is the **MariaDB Community Server** which you can download, use, and modify for free. The second edition is the **MariaDB Enterprise Server** intended to replace proprietary databases and adopt open source in the enterprise.

How Does MariaDB Work?

Just like **MySQL**, **MariaDB** also uses a client/server model with a server program that files requests from client programs. As is typical of client/server computer systems, the server and the client programs can be on different hosts.



MySQL Client-Server Structure

Key Features of MariaDB

MariaDB is highly compatible with **MySQL** as every **MariaDB** version works as a "**drop-in replacement**" for the equivalent **MySQL** version, however, with a couple of limitations.

If you are migrating to **MariaDB**, its data files are generally binary compatible with those from the equivalent **MySQL** version, and also MariaDB's client protocol is binary compatible with MySQL's client protocol.

- It supports many different SQL statements, structure, and rules, functions and procedures, user-defined functions (useful for extending MariaDB), server variables, and SQL modes, partitioning of tables, database backup, and restoration, server monitoring and logs. It also ships with several plugins such as the MariaDB audit plugin, and more.
- MariaDB comes with many new options, features, and extensions, storage engines, as well as bug fixes that are not in MySQL. Some of the new features in MariaDB are advanced clustering with Galera Cluster 4, several compatibility features with Oracle Database, and Temporal Data Tables (which allows you to query the data as it stood at any point in the past), and so much more.
- The same security features in MySQL exist in MariaDB. Additionally, you should consider best practices to secure your database server. Also, securing your database should start right at the network and server level.

It is important to understand that although MariaDB remains compatible with MySQL, it is truly open-source (and is developed by the community in true open-source spirit), it doesn't have any closed source modules like the ones that exist in MySQL Enterprise Edition.

The MariaDB documentation will help you to fully understand the differences between MySQL and MariaDB.

MariaDB Client and Tools

For both **MariaDB** and **MySQL**, all client APIs and structs are identical, all ports and sockets are generally the same, and all MySQL connectors for programming languages such as Python, Perl, PHP, Ruby, Java, and MySQL C connector, etc work unchanged under MariaDB.

Also, MariaDB comes with several client programs such as the popular command-line utilities: mysql, mysqladmin, and mysqldump, for administering databases.

Who is Using MariaDB?

Some of companies using **MariaDB** include **RedHat**, Ubuntu, Google, Wikipedia, Tumblr, Amazon Web Services, SUSE Linux, and more.

How to install MariaDB on RHEL

The procedure to install MariaDB on a Red Hat Enterprise Linux 8 is as follows:

1. Open the terminal application. Another option is to log in using the ssh command `ssh user@rhel-8-server-ip`
2. Installing the MariaDB on RHEL 8, type:
   `sudo yum install mariadb-server`
3. Securing the MariaDB server in RHEL 8, run:
   `sudo mysql_secure_installation`
4. Finally test your installation by running:
   `mysql -u root -p`