Knocks Server Setup

Linux Fedora

Linux Fedora is a distribution that developed by RedHat Enterprise, its the parent distro for many linux operating systems such as REHL, CentOS and many more

Fedora offers many editions, as an operating system with different functionalities.

1

WORKSTATION

Developers addition that makes it easy for the team to work on the same platform

2

SERVER

Server OS that offers many security modules that makes the configuration easier

3

ATOMIC

Platform for LDK application stack.

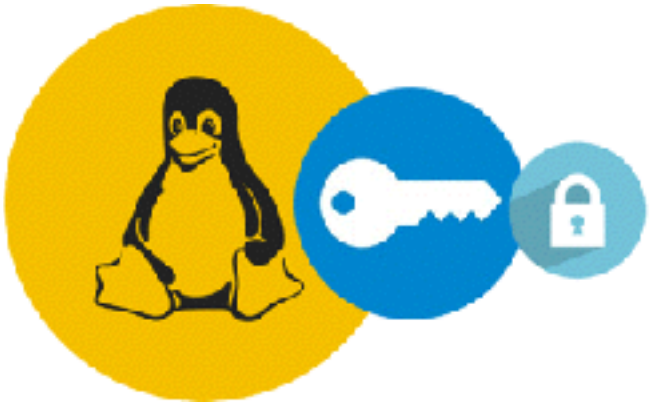
Knocks Sever Operating System setup and configuration

Knocks is using a RedHat Linux based operating system

Fedora Server 26 2017 Edition is our distribution for the server OS

Fedora partitioning

Linux id usually installed on a separated partitions, each of which has a different purpose



- /boot
  The boot partition is taking ext2 as an extension, and requires a boot flag , this is where the boot loader lays , Fedora is using GRUB as boot loader, which is stable and guarantees a quick start and modifying the loadable kernel modules; GRUP is also locating the other operating systems on the device if exists and makes it possible to have one boot loader to join all the operating systems.

- /swap
  Swap partition is taking SWAP as an extension, simply swap is assisting the device's cache memory and boasting up the performance whenever the cache memory is busy, SWAP partitions are not associated with a specific operating system on the machine.

- /root
  The root partition is taking EXT4 or EXT2, they are usually carries a journalist prefix , its optional for the host to encrypt his data using a very simple configuration while the setup is proceeding which can form an additional protection layer for the server's data and also makes it easier to handle the system user's privileges, generally the root partition holds the bin files and the rest of programs fils, and also the binaries for the system compilers.

WHY FEDORA SERVER ?

The simplicity introduced with Rolekit and Cockpit have made server deployments a breeze. What took me a few days on other operating systems took less than an hour with Fedora 26 Server.
Fedora also offers a very powerful firewall that already built in the OS, no need for an external installations or a complex configuration, SELinux is smart enough to prevent DDNS attacks automatically and protect the server from the scripting attacks and the invalid uploads in general.

Fedora is also much easier than many other distributions and more familiar to the development team which saves much time.

Network Authentication

Once the system installation is done then the next step is to begin installing the packages that server needs to run, installing such packages needs an internet connection.
Fedora server is coming with a console mode, which means that the admin needs to connect manually.
So first we need to check if there is a connected wireless devices attached to our server incase we desire to connect using wi-fi.

First we need to list our attached devices.

```
# ifconfig
```

To show us the current status for our server connections and display any associated networks if existed.

As we still don't know what is the name of the server wifi device we need to list all the network devices that connected to our sever.

To list them we need to use the command:

```
# ifconfig -a
```

You can find your wifi device is taking the name of:

wlan0 which is common for the Debian based systems (Debian , Kali Linux, Ubuntu).

wlp2s0 which is common for the redHat based systems (Fedora , CentOS , RHEL).

After finding your wireless interface you need to make sure whether the device is currently up or not.

You can do this using iwconfig library.

```
# iwconfig yourInterface
```

In our case:

```
# iwconfig wlp2s0
```

Then you need to know how to switch on and off your network devices

So back to ifconfig library and use ifconfig youInterface desiredStatus.

In our case:

```
# ifconfig wlp2s0 up
```

That is turning the device up, you can also make sure using iwconfig as we've mentioned,

then you will need to find which network to connect.

You can do this using a several libraries.

iwconfig offers listing and connection modules that is more specific than ever, it gives you a list for a specific device.

You can figure it out using:

```
# man -k iw
```

That will list a lot of commands and stuff you can do using iw library.

As a result you will find iwlist which you can use to list all the seen networks for some device.

```
# iwlist wlp2s0
```

That will return a lot of details about the network such as ESSID, signal strength, security protocols, and many more.

Then you can use the same library to connect to the network;

Just hit man iwlist and that will lit you see all the possible things that you can do

So for scanning you can use:

```
# iwlist wlp2s0 scanning
```

Then we can connect to any network using:

```
# iwconfig wlp2s0 nwESSID networkName key nwPassword
```

Another simple way is by using NMCLI library which is making everything easier but showing less details.

Scan for the available networks using :

```
# nmcli dev wifi
```

And no need to mention what device name you need it to scan.

And also

```
# nmcli dev wifi connect nwESSID password nwPassword
```

For associating with any network

The good news is that Fedora is including both NMCLI and iw as a built in module so need to install it as it already exists.

Working With DNF

DNF Packages Manager

A dependency manager that validate installing packages, uninstalling and updating them, it also makes sure the the installed packages does not require something that is not exists.

As we need to install many external packages, we need a good dependency manager, Fedora offers 2 dependency managers, YUM and DNF.

Both of them is great but DNF is being updated frequently more than YUM so DNF can guarantee installing the newest versions always as he is having a newer repositories.

So our very first thing will be updating the DNF packages and repositories.

DNF is a very powerful dependency manager that allows the user to update every single application on the whole system using only one command!

But before we begin updating our system there's something we need to focus on.

The server admin need to have the privilege to install or remove packages, as we've just made a fresh install we can login as 'root', this is a user that you must create while setting up the system and the root user is having all the privileges, but normally it's not recommended to work as a root, that means that you allow to lots of thing to happen without a permission, so using another created user will require the admin permission lots of actions which can make the admin always aware of whats going on.

So First Login as root, or use another user and in this case you need to begin your session with:

```
# su
```

which opens a root session or you can use ‘sudo’ as a prefix for most of commands.

Then let’s update our Fedora Server packages.

```
# dnf update
```

You can hit dnf and it will list you many options such as upgrading the whole distro version and many more.

After updating the system its better to restart the system as you can begin a fresh session with the new updates.

After restarting keep in mind to login as root or begin a session with ‘su’, DNF will not allow you to install any application without satisfying the administration privilege which is powerful to have one platform for packages installations and secure your system in the same time.

You’ll also need to keep in mind your network status, hit ‘ifconfig’ to see whether you’re connected to any network or not, if you’re connected you will also be able to see the assigned IP address that your device takes in the local network and many more details, otherwise you’ll need to reconnect using ‘nmcli’ or ‘iwconfig’.

Before we start installing those packages we need to know that most of those packages creates its own configuration files on ‘/etc/’ directory which makes it possible to edit in their settings from one directory.

Knocks Packages

Apache HTTP Server And Routes Manager

Composer PHP Dependency Manager

MongoDB NoSQL Database

Maria Database Server/Client MySQL Relational Database

Laravel PHP MVC Framework

GNOME GUI and Desktop environment

PHP 7.1.8 Server side compiler working with Zend Engine

NPM Javascript package manager

phpMyAdmin

Graphical database manager for MySQL

Cockpit Server Resource Manager

Mozilla Firefox Web browser includes development and testing tools

Compass Graphical database for MongoDB

Packages Installation

First of all we need to focus on our installation sequence as we don't want to stuck in dependency flags.

So we'll begin with installing a packages that don't require any missing dependencies.

1 HTTPD Apache Server

```
# dnf install httpd
# dnf systemctl enable httpd.service
# systemctl start httpd.service
```

First we need to request the package from DNF.

After installing the package we need to turn it to a service so we can easily control when to start or stop it.

We do this by enabling httpd as a service then we can start it and also start the service with every server boot.

After enabling httpd as a service we can start/ stop/restart it normally with those commands

```
# systemctl start httpd
# systemctl stop httpd
# systemctl restart httpd
```

2 Maria DB (MySQL)

Installing MariaDB is very similar to httpd, that's thats because MariaDB is a service too.

MariaDB is having 2 main packages one for the server and the other's client package.

It could be enough for us to install the server package only but currently Knocks server is a platform for development too, so we need both of them.

```
#  //Client Package
#  sudo dnf install mariadb
#  //Server Package
#  dnf install mariadb-server
#  systemctl enable mariadb
#  systemctl start mariadb
```

So we did the same as we did with httpd for enabling and starting mariadb as a service, but for MariaDB this is not everything, we still need more configurations, as a database we can't use it without having a users, So we need to add a root user at least, and configure other things such as enabling or disabling the remote access and the anonymous users position and many more, MariaDB offers a script that assist you to configure those settings without going to the configuration file, so we need to run this script, luckily we don't even need to search for it and run it manually, DNF is making an alias for this script so you can execute it using the following command

```
#  mysql_secure_installation
```

Then You can test you connection by joining the database, you can call mysql globally from the terminal such as the following:

```
#  mysql -uroot -pyourpassword
#  //The output should be 'mysql >' opening a new session
```

3 PHP 7.1.8

PHP is the 3rd in our installation sequence because each of the previous packages (httpd, mariadb) are needed to make php works.
httpd is required for both MariaDB and PHP, for PHP both of them are required, because PHP is working with MySQL database and includes a modules for it already.

As a compiler, installing PHP is much easier than mariadb and httpd all we need is to install the package from DNF and he'll do the rest and copy the compiler binaries to the '/usr/bin' directory which usually includes the binaries for most of compilers.

So all we need to do is installing from DNF and specifying the last version as we need it for upcoming packages such as Composer and Laravel.

So we can do this using only one command and then its done!

```
# dnf install php-cli
```

That means that DNF will install the last stable version automatically for me.

Usually PHP is installing many modules automatically, so we don't need to install any modules for the moment.

Make sure That the installed version is 7.1.8 or greater using the following command:

```
# php --version
# //or just php -V
```

This command is telling which version of PHP is installed on your system.

4 Composer

Composer is a PHP dependency manager, it makes it easier than ever to install plugins and external modules for PHP.

Composer's documentation is pretty good to understand what can Composer do for your PHP, they also offers a packages gallery that we can easily search and browse for packages on getcomposer.org .

As a dependency manager its very similar to DNF but only works for PHP, which means that we can install or remove PHP packages through Composer.

Obviously Composer is requiring PHP installed and lately its requiring a higher versions from PHP (<= 7), and as we've installed version 7.1.8 so we have no problem for this.

For installing Composer we can do it through DNF or we can use cURL project to do it, luckily cURL is already installed on Fedora so we can just use it.
if you are not familiar with cURL its a data transferring tool like a download manager but works extremely good with the command line environment, hit 'curl -h' for more.

The reason why we'll download it and install it manually instead of using DNF is that we want make sure that we're installing Composer globally.

```
#  curl -sS https://getcomposer.org/installer | php
#  mv composer.phar /usr/local/bin/composer
#  chmod +x /usr/local/bin/composer
#
```

What we've done here is downloading 'composer.phar' file from Composer's website, then we've moved it to lay on the local binaries directory so we can join composer globally, the last command is giving composer directory the executing privilege.

You can test you installation by running Composer's alias 'composer' or checking Composer's version.

```
#  composer
#  composer -V
```

5 Laravel

Laravel is PHP MVC Framework that makes it easier for the development team to work using PHP, Laravel is coming with many modules built in such as 'Laravel/Router'.

Laravel is supported by Composer and together they are totally helpful for the development team and saves a lot of time.

Installing laravel used to be easy as we only use Composer to make it happen.
all we've to do is to require Laravel Package globally from Composer.

```
#  composer global require "laravel/installer"
```

Unlike DNF, Composer does not verify Laravel after installation, and as we are going to use Laravel from the Terminal we need to verify Laravel paths to the system, and also create an alias for laravel so we can join it globally.

```
#  echo 'export PATH="$PATH:$HOME/.config/composer/vendor/bin"' >>
   ~/.bashrc
#  echo 'alias laravel="~/.composer/vendor/bin/laravel" >> ~/.bashrc
```

So now we've added two lines to the bashrc file that makes it possible to join Laravel.

If you want to test hit 'laravel' which is the alias that we've just created, and then the system should recognize the alias and begin executing the binaries in the mentioned directory.

6 MongoDB

MongoDB is an open source noSQL database which is widely known.
MongoDB is storing data as a JSON documents which makes it flexible with schemas and also makes it easier for the developers to work with it as they don't have to learn a new syntax.
MongoDB is great with the big data in general, Knocks Development team intends to use it for storing and processing the media blobs as an objects.

Installing MongoDB has no dependencies, you can just go ahead and install it from DNF and its totally done!

```
#  dnf install mongodb mongodb-server
```

As a database we need to mention MongoDB as a service, so for testing and running MongoDB all you need to do is:

```
#  service mongo start
#  mongo
#
```

7 NPM

NPM is a javascript dependency manager that contains many useful packages could boast up the development process.

NPM stands for Node Package Manager, Node is meaning NodeJS which we're going to install through NPM

Installing NPM is totally done by DNF too.

So Next we'll Install NPM and use it to install another things such as MongoDB Compass, But first we need to install NodeJS which is a runtime server side language based on javascript

```
# dnf install nodejs
# dnf install npm
#
```

The default Version From NodeJS is running on Google V8 Engine which is exactly what we need.

NodeJS

NodeJS is a framework based on javascript and its well known as a fast engine for the server side, NodeJS includes many modules that will help Knocks Development Teams.

8 GNOME

GNOME is a graphical desktop environment that's works on most of Linux distros.

GNOME is a brilliant interface and light spaced on the disk and processor at the same time which is not achieved in many other GUIs such as KDE, KDE is more than the double of GNOME's space and a way heavier on the processor, However GNOME is installed with a switch which means that the admin can turn it on and off at anytime which is very flexible.

GNOME isn't actually one package, its a group package that comes with graphical directory browsers, and other regular utilities that will only run under GNOME as a process.

Installing GNOME is very easy by DNF, you don't need to mention every member in the group package, all you need to do is:

```
#  dnf groupinstall gnome
```

That means that DNF will handle installing and verifying all of those packages.

And then it's time to get out from the console mode

```
#  startx
```

And the X-Based system that we've just installed will start GNOME.

After installing GNOME now we can install and test all the graphical packages that we've mentioned.

9 Mozilla Firefox

Mozilla Firefox is a powerful web browser that contains many development, measuring, and testing tools.

Installing Mozilla Firefox is being done by DNF as the repository of it is already mentioned.

```
# dnf install firefox
```

Once the installation is complete you should see the application's icon in GNOME menu.

10 phpMyAdmin

phpMyAdmin is a graphical tool for the relational database MySQL that makes it easy for the developers to browse their tables and represent the schematics easier.

phpMyAdmin is a web application and it only requires:
(httpd, mariadb, PHP, php-mysqli) and all of them should be covered in the current stage.

Installing phpMyAdmin is very simple all we need to do is installing the package from DNF and DNF will do the rest.

```
# dnf install phpMyAdmin
```

Then you can join it locally on http://127.0.0.1/phpMyAdmin

11 MongoDB Compass

MongoDB Compass is a graphical tool for the noSQL database MongoDB, its ranked as the best among all the alternative tools, Compass is not only a query platform, it includes many statistical terminals that assists the developer to know much a way easier.

For Compass you can head to https://mongodb.com/compass and download it using your browser, Then you can go to the downloaded file and just execute it.

Once the installation is complete you should see the application's icon in GNOME menu.

12 Cockpit

Cockpit is a resource manager for Fedora Server that includes many statistical terminals to show the admin many details about the server performance, storage, network status and many more.

Installing Cockpit is being done by DNF but Cockpit still needs some configuration.

```
#  dnf install cockpit
#  systemctl enable cockpit.socket
#  //--now flag could be an option
#  firewall-cmd --add-service=cockpit
#  firewall-cmd --add-service=cockpit --parmanent
#
#
```

Enabling Cockpit and verifying it to the fire wall is necessary, but its optional to determine whether you want it to start with every boot or not.

Knocks Installation Script

```
#  su
#  dnf update
#  dnf install httpd
#  dnf systemctl enable httpd.service
#  systemctl start httpd.service
#  systemctl restart httpd
#  sudo dnf install mariadb
#  dnf install mariadb-server
#  systemctl enable mariadb
#  systemctl start mariadb
#  mysql_secure_installation
#  systemctl restart httpd
#  dnf install php-cli
#  systemctl restart httpd
#  curl -sS https://getcomposer.org/installer | php
#  mv composer.phar /usr/local/bin/composer
#  chmod +x /usr/local/bin/composer
#  composer global require “laravel/installer”
#  echo ‘export PATH=“$PATH:$HOME/.config/composer/vendor/bin”’ >>
   ~/.bashrc
#  echo ‘alias laravel=“~/.composer/vendor/bin/laravel” >> ~/.bashrc
#  dnf install mongodb mongodb-server
#  dnf install nodejs
#  dnf install npm
#  dnf groupinstall gnome
#  dnf install firefox
#  dnf install phpMyAdmin
#  systemctl restart httpd
#  dnf install cockpit
#  systemctl enable cockpit.socket
#  firewall-cmd --add-service=cockpit
#  firewall-cmd --add-service=cockpit --parmanent
#  systemctl restart httpd
#  reboot
#
#
```