

## Shell Scripting Project

This is shell scripting project for automating the manual deployment done of the LAMP Application that we did for DevOps Pre-Requisite Course Project.

Before doing this project refer to that project once.

**GIT Repo for this project:** <https://github.com/kodekloudhub/shell-scripting-for-beginners-course>

Commands used for that project: <https://github.com/kodekloudhub/learning-app-ecommerce>

Now we have to write a bash script where all the commands we used in linux project will be added in this bash script and by running bash script every command will be executed and LAMP stack application will be deployed.

### Commands:

#### 1. Install FirewallD

```
sudo yum install -y firewalld
sudo systemctl start firewalld
sudo systemctl enable firewalld
sudo systemctl status firewalld
```



## Deploy and Configure Database

#### 1. Install MariaDB

```
sudo yum install -y mariadb-server
sudo vi /etc/my.cnf
sudo systemctl start mariadb
sudo systemctl enable mariadb
```



#### 2. Configure firewall for Database

```
sudo firewall-cmd --permanent --zone=public --add-port=3306/tcp
sudo firewall-cmd --reload
```



#### 3. Configure Database

```
$ mysql
MariaDB > CREATE DATABASE ecomdb;
MariaDB > CREATE USER 'ecomuser'@'localhost' IDENTIFIED BY 'ecompassword';
MariaDB > GRANT ALL PRIVILEGES ON *.* TO 'ecomuser'@'localhost';
MariaDB > FLUSH PRIVILEGES;
```



ON a multi-node setup remember to provide the IP address of the web server here: 'ecomuser'@'web-server-ip'

#### 4. Load Product Inventory Information to database

Create the db-load-script.sql

```
cat > db-load-script.sql <<-EOF
```

USE ecomdb;

```
CREATE TABLE products (id mediumint(8) unsigned NOT NULL auto_increment, Name varchar(255)
default NULL, Price varchar(255) default NULL, ImageUrl varchar(255) default NULL, PRIMARY KEY (id))
AUTO_INCREMENT=1;
```

```
INSERT INTO products (Name,Price,ImageUrl) VALUES ("Laptop","100","c-1.png"),("Drone","200","c-2.png"),("VR","300","c-3.png"),("Tablet","50","c-5.png"),("Watch","90","c-6.png"),("Phone Covers","20","c-7.png"),("Phone","80","c-8.png"),("Laptop","150","c-4.png");

EOF
```

Run sql script

```
sudo mysql < db-load-script.sql
```

## Deploy and Configure Web

### 1. Install required packages

```
sudo yum install -y httpd php php-mysqlnd
sudo firewall-cmd --permanent --zone=public --add-port=80/tcp
sudo firewall-cmd --reload
```

### 2. Configure httpd

Change `DirectoryIndex index.html` to `DirectoryIndex index.php` to make the php page the default page

```
sudo sed -i 's/index.html/index.php/g' /etc/httpd/conf/httpd.conf
```

### 3. Start httpd

```
sudo systemctl start httpd
sudo systemctl enable httpd
```

### 4. Download code

```
sudo yum install -y git
sudo git clone https://github.com/kodekloudhub/learning-app-ecommerce.git /var/www/html/
```

### 5. Update index.php

Update [index.php](#) file to connect to the right database server. In this case `localhost` since the database is on the same server.

```
sudo sed -i 's/172.20.1.101/localhost/g' /var/www/html/index.php

<?php
    $link = mysqli_connect('172.20.1.101', 'ecomuser', 'ecompassword', 'ecomdb');
    if ($link) {
        $res = mysqli_query($link, "select * from products;");
        while ($row = mysqli_fetch_assoc($res)) { ?>
```

ON a multi-node setup remember to provide the IP address of the database server here.

```
sudo sed -i 's/172.20.1.101/localhost/g' /var/www/html/index.php
```

### 6. Test

```
curl http://localhost
```

Create a CentOS machine and create bash script file called deploy.sh using **touch deploy.sh** or **vi deploy.sh** command then **chmod +x deploy.sh**

In the bash file first add all the commands we used in LAMP Project. Then we will start modifying it.

Before adding commands into deploy.sh file check them once executing in CLI whether they are working or not. Then add them in deploy.sh file.

All commands till **configure firewall for database** can be added. But the commands we used for configure Database can not be added simply like that. Because we logged into MySQL shell and then executed commands. So, we need to find an alternative approach.

### 3. Configure Database

```
$ mysql
MariaDB > CREATE DATABASE ecomdb;
MariaDB > CREATE USER 'ecomuser'@'localhost' IDENTIFIED BY 'ecompassword';
MariaDB > GRANT ALL PRIVILEGES ON *.* TO 'ecomuser'@'localhost';
MariaDB > FLUSH PRIVILEGES;
```

We will create a database script where we will add all the commands that we executed in MySQL shell. Then we will run that script using MySQL Client.

**Commands:** Checking in CentOS machine before adding in deploy.sh script file.

```
[bob@caleston-lp10 ~]$ cat > configure-db.sql
CREATE DATABASE ecomdb;
CREATE USER 'ecomuser'@'localhost' IDENTIFIED BY 'ecompassword';
GRANT ALL PRIVILEGES ON *.* TO 'ecomuser'@'localhost';
FLUSH PRIVILEGES;

[bob@caleston-lp10 ~]$ sudo mysql < configure-db.sql
```

Since, this is working in CLI. Add these in Script file. Also, add **EOF** as we need to press ctrl+c to exit after cat command.

**Script:**

```
cat > configure-db.sql <<-EOF
CREATE DATABASE ecomdb;
CREATE USER 'ecomuser'@'localhost' IDENTIFIED BY 'e
GRANT ALL PRIVILEGES ON *.* TO 'ecomuser'@'localhos
FLUSH PRIVILEGES;
EOF
sudo mysql < configure-db.sql
```

Add rest of the commands as it is like initial steps.

**Version 1 of deploy.sh:**

```
sudo yum install -y firewalld
sudo service firewalld start
sudo systemctl enable firewalld
```

```
sudo yum install -y mariadb-server
sudo service mariadb start
sudo systemctl enable mariadb
```

```
sudo firewall-cmd --permanent --zone=public --add-p
sudo firewall-cmd --reload
```

```
cat > configure-db.sql <<-EOF
CREATE DATABASE ecomdb;
CREATE USER 'ecomuser'@'localhost' IDENTIFIED BY 'e
GRANT ALL PRIVILEGES ON *.* TO 'ecomuser'@'localhos
FLUSH PRIVILEGES;
EOF
```

```
sudo mysql < configure-db.sql
```

```
cat > db-load-script.sql <<-EOF
USE ecomdb;
CREATE TABLE products (id mediumint(8) unsigned NOT
INSERT INTO products (Name,Price,ImageUrl) VALUES (
EOF
```

```
mysql < db-load-script.sql
```

```

sudo yum install -y httpd php php-mysql
sudo firewall-cmd --permanent --zone=public --add-port=80/tcp
sudo firewall-cmd --reload

sudo sed -i 's/index.html/index.php/g' /etc/httpd/conf/httpd.conf

sudo service httpd start
sudo systemctl enable httpd

sudo yum install -y git
sudo git clone https://github.com/kodekloudhub/learn-php.git

sudo sed -i 's/172.20.1.101/localhost/g' /var/www/html/index.php

```

Now in CLI give curl <http://localhost> you test once using: <http://localhost:80>

**KODEKLOUD**
Laptops
Drones
Gadgets
Phones
VR
Contact us

## Product List

**Laptop Pro**

Purchase MB at the lowest price **100\$**

**Drone**

Purchase Multifunctional drones **200\$**

**VR**

Explore our VR Devices

**Macbook air**

Purchase MB at the lowest price **500\$**

Let's try to improve the Script.

When we try to execute this script, the output it displays is not user friendly. It executes next steps if any step fails in between. So, let's add some comments to the scripts and some print messages.



**V2: Adding comments for each section for better readability and echo commands to print what exactly is being done.**

```
# ----- Database Configuration -----
# Install and configure FirewallD
echo "Installing firewallD..."
sudo yum install -y firewallD
sudo service firewallD start
sudo systemctl enable firewallD

# Install and configure MariaDB
echo "Installing MariaDB..."
sudo yum install -y mariadb-server
sudo service mariadb start
sudo systemctl enable mariadb

# Add FirewallD rules for database
echo "Adding Firewall rules for DB..."
sudo firewall-cmd --permanent --zone=public --add-port=3306/tcp
sudo firewall-cmd --reload

# Configure Database
echo "Configuring DB..."
cat > configure-db.sql <<-EOF
CREATE DATABASE ecomdb;
CREATE USER 'ecomuser'@'localhost' IDENTIFIED BY 'ecompassword';
GRANT ALL PRIVILEGES ON *.* TO 'ecomuser'@'localhost';
FLUSH PRIVILEGES;
EOF

sudo mysql < configure-db.sql

# Load inventory data into Database
echo "Loading inventory data into DB..."
cat > db-load-script.sql <<-EOF
USE ecomdb;
CREATE TABLE products (id mediumint(8) unsigned NOT NULL auto_increment,Name varchar(255) default NULL,Price v

INSERT INTO products (Name,Price,ImageUrl) VALUES ("Laptop","100","c-1.png"),("Drone","200","c-2.png"),("VR","

EOF

sudo mysql < db-load-script.sql

# ----- Web Server Configuration -----
echo "Configuring Web Server..."
# Install apache web server and php
sudo yum install -y httpd php php-mysql

echo "Configuring FirewallD rules for web server..."
# Configure Firewall rules for web server
sudo firewall-cmd --permanent --zone=public --add-port=80/tcp
sudo firewall-cmd --reload

sudo sed -i 's/index.html/index.php/g' /etc/httpd/conf/httpd.conf

# Start and enable httpd service
echo "Starting web server..."
sudo service httpd start
sudo systemctl enable httpd

# Install GIT and download source code repository
echo "Cloning GIT Repo..."
sudo yum install -y git
sudo git clone https://github.com/kodekloudhub/learning-app-ecommerce.git /var/www/html/

# Replace database IP with localhost
sudo sed -i 's/172.20.1.101/localhost/g' /var/www/html/index.php

echo "All set."
```

Now, the messages will be displayed but you won't be able to identify them as they will scroll in the flow. We can add some colours to these echo commands and then we can differentiate the sections also easily.

### V3: Adding colours to echo commands

We can do like this:

```
# Install and configure FirewallD
echo -e "\033[0;32mInstalling firewallld...\033[0m"

[ bob@caleston-lp10 ~]$ echo -e "\033[0;32mInstalling firewallld...\033[0m"
Installing firewallld...
```

but the problem is we cannot change the colours for all the echo commands when ever we want to change. Better to assign this colour value to a variable and call that variable then we can just change the variable value of any desired colour.

**Step – 1:** Added Green and No Colour (NC) variables to the first section echo command. We have to do this for all commands in the script.

```
1 ► GREEN="\033[0;32m"
2   NC="\033[0m"
3
4   # ----- Database Configuration -----
5   # Install and configure FirewallD
6   echo -e "${GREEN}Installing firewallld...${NC}"
7   sudo yum install -y firewallld
8   sudo service firewallld start
9   sudo systemctl enable firewallld
```

**Checking:**

```
[ bob@caleston-lp10 ~]$ GREEN="\033[0;32m"
[ bob@caleston-lp10 ~]$ NC="\033[0m"
[ bob@caleston-lp10 ~]$
[ bob@caleston-lp10 ~]$ # ----- Database Configuration -----
[ bob@caleston-lp10 ~]$ # Install and configure FirewallD
[ bob@caleston-lp10 ~]$ echo -e "${GREEN}Installing firewallld...${NC}"
Installing firewallld...
```

Since we are using those variables multiple times, we have to add these variables everywhere in the script where ever there is an echo command. Then instead, we can create a function and call that function.

```
function print_green(){
    GREEN="\033[0;32m"
    NC="\033[0m"

    echo -e "${GREEN} $1 ${NC}"
}

# ----- Database Configuration -----
# Install and configure FirewallD
print_green "Installing firewallld..."
```

**print\_green** is calling function and **"Installing firewallld..."** is an argument passed to that function which will be used as \$1 in the echo command which is inside the **print\_green** function.

Now, Whatever message you pass it will be printed with Green Colour. In future, we can change color also.

Now, we just have to replace all the echo commands similar to above one.

**Checking:**

```
[ bob@caleston-lp10 ~]$ function print_green(){
>   GREEN="\033[0;32m"
>   NC="\033[0m"
>
>   echo -e "${GREEN} $1 ${NC}"
> }
[ bob@caleston-lp10 ~]$
```

```
[bob@caleston-lp10 ~]$ print_green "Hello"
Hello
```

#### Version4: With Colour echo commands to differentiate sections.

```
function print_green(){
    GREEN="\033[0;32m"
    NC="\033[0m"

    echo -e "${GREEN} $1 ${NC}"
}

# ----- Database Configuration -----
# Install and configure FirewallD
print_green "Installing firewallD..."

sudo yum install -y firewallD
sudo service firewallD start
sudo systemctl enable firewallD

# Install and configure MariaDB
print_green "Installing MariaDB..."
sudo yum install -y mariadb-server
sudo service mariadb start
sudo systemctl enable mariadb

# Add FirewallD rules for database
print_green "Adding Firewall rules for DB..."
sudo firewall-cmd --permanent --zone=public --add-port=3306/tcp
sudo firewall-cmd --reload

# Configure Database
print_green "Configuring DB..."
cat > configure-db.sql <<-EOF
CREATE DATABASE ecomdb;
CREATE USER 'ecomuser'@'localhost' IDENTIFIED BY 'ecompassword';
GRANT ALL PRIVILEGES ON *.* TO 'ecomuser'@'localhost';
FLUSH PRIVILEGES;
EOF

sudo mysql < configure-db.sql

# Load inventory data into Database
print_green "Loading inventory data into DB..."
cat > db-load-script.sql <<- EOF
USE ecomdb;
CREATE TABLE products (id mediumint(8) unsigned NOT NULL auto_increment,Name varchar(255) default NULL,Pr
INSERT INTO products (Name,Price,ImageUrl) VALUES ("Laptop","100","c-1.png"),("Drone","200","c-2.png"),("
EOF

sudo mysql < db-load-script.sql

# ----- Web Server Configuration -----
print_green "Configuring Web Server..."
# Install apache web server and php
sudo yum install -y httpd php php-mysql

print_green "Configuring FirewallD rules for web server..."
# Configure Firewall rules for web server
sudo firewall-cmd --permanent --zone=public --add-port=80/tcp
sudo firewall-cmd --reload

sudo sed -i 's/index.html/index.php/g' /etc/httpd/conf/httpd.conf

# Start and enable httpd service
print_green "Starting web server..."
sudo service httpd start
sudo systemctl enable httpd
```



```
# Install GIT and download source code repository
print_green "Cloning GIT Repo..."
sudo yum install -y git
sudo git clone https://github.com/kodekloudhub/learning-app-ecommerce.git /var/www/html/

# Replace database IP with localhost
sudo sed -i 's/172.20.1.101/localhost/g' /var/www/html/index.php

print_green "All set."
```

Let's, implement some checks in order to identify the failures:

If first step, firewalld setup is not done properly then the script needs to stop execution.

To do that we need to know the status of the firewalld service. We can know that by systemctl is-active command. So, we will assign the value of this command to a variable is\_firewalld\_active and then prints the echo command in green if it is active and prints in red if it is not active.

```
# ----- Database Configuration -----
# Install and configure FirewallD
print_green "Installing firewalld..."

sudo yum install -y firewalld
sudo service firewalld start
sudo systemctl enable firewalld

is_firewalld_active=$(systemctl is-active firewalld)

if [ $is_firewalld_active = "active" ]
then
    print_green "Firewalld Service is active"
else
    echo "Firewalld Service is not active"
    exit 1
fi
```

You can create new function again for red color, but having two functions for same thing is not a good practice.

```
function print_red(){
    RED="\033[0;31m"
    NC="\033[0m"

    echo -e "${RED} $1 ${NC}"
}
```

These two functions could be merged into a single function. We will pass color and message as two parameters to the function. We can use Case block to identify the argument 1 – color and then pass the respective color code.

```
function print_color(){
    case $1 in
        "green") COLOR="\033[0;32m" ;;
        "red") COLOR="\033[0;31m" ;;
        "*") COLOR="\033[0m" ;;
    esac
    echo -e "${COLOR} $2 ${NC}"
}
```

## Checking:

```
[bob@caleston-lp10 ~]$ function print_color(){
>
> case $1 in
>   "green") COLOR="\033[0;32m" ;;
>   "red") COLOR="\033[0;31m" ;;
>   "*") COLOR="\033[0m" ;;
>   esac
>
> echo -e "${COLOR} $2 ${NC}"
>
> }
[bob@caleston-lp10 ~]$
[bob@caleston-lp10 ~]$ print_color "green" "Hello"
Hello
[bob@caleston-lp10 ~]$ print_color "red" "Hello"
Hello
```

It's working fine now. So, we have to remove the two colour functions and update all the function call commands in the script:

We will just change the function name and adds one more argument – colour into the command.

```
# ----- Database Configuration -----
# Install and configure FirewallD
print_color "green" "Installing firewallld..."

sudo yum install -y firewallld
sudo service firewallld start
sudo systemctl enable firewallld

is_firewalld_active=$(systemctl is-active firewallld)

if [ $is_firewalld_active = "active" ]
then
    print_color "green" "Firewalld Service is active"
else
    print_color "red" "Firewalld Service is not active"
    exit 1
fi
```

This change we have to do all the places wherever we need colour statements.

## Version 5 – Adding multiple colours in single function

```
function print_color(){

    case $1 in
        "green") COLOR="\033[0;32m" ;;
        "red") COLOR="\033[0;31m" ;;
        "*") COLOR="\033[0m" ;;
    esac

    echo -e "${COLOR} $2 ${NC}"

}

# ----- Database Configuration -----
# Install and configure FirewallD
print_color "green" "Installing firewallld..."

sudo yum install -y firewallld
sudo service firewallld start
sudo systemctl enable firewallld

is_firewalld_active=$(systemctl is-active firewallld)

if [ $is_firewalld_active = "active" ]
then
    print_color "green" "Firewalld Service is active"
else
    print_color "red" "Firewalld Service is not active"
    exit 1
fi
```

```

# Install and configure MariaDB
print_color "green" "Installing MariaDB..."
sudo yum install -y mariadb-server
sudo service mariadb start
sudo systemctl enable mariadb

# Add FirewallD rules for database
print_color "green" "Adding Firewall rules for DB..."
sudo firewall-cmd --permanent --zone=public --add-port=3306/tcp
sudo firewall-cmd --reload

# Configure Database
print_color "green" "Configuring DB..."
cat > configure-db.sql <<-EOF
CREATE DATABASE ecomdb;
CREATE USER 'ecomuser'@'localhost' IDENTIFIED BY 'ecompassword';
GRANT ALL PRIVILEGES ON *.* TO 'ecomuser'@'localhost';
FLUSH PRIVILEGES;
EOF

sudo mysql < configure-db.sql

# Load inventory data into Database
print_color "green" "Loading inventory data into DB..."
cat > db-load-script.sql <<-EOF
USE ecomdb;
CREATE TABLE products (id mediumint(8) unsigned NOT NULL auto_increment,Name varchar(255) default NULL,Price

INSERT INTO products (Name,Price,ImageUrl) VALUES ("Laptop","100","c-1.png"),("Drone","200","c-2.png"),("VR",

EOF

sudo mysql < db-load-script.sql

# ----- Web Server Configuration -----
print_color "green" "Configuring Web Server..."
# Install apache web server and php
sudo yum install -y httpd php php-mysql

print_color "green" "Configuring FirewallD rules for web server..."
# Configure Firewall rules for web server
sudo firewall-cmd --permanent --zone=public --add-port=80/tcp
sudo firewall-cmd --reload

sudo sed -i 's/index.html/index.php/g' /etc/httpd/conf/httpd.conf

# Start and enable httpd service
print_color "green" "Starting web server..."
sudo service httpd start
sudo systemctl enable httpd

# Install GIT and download source code repository
print_color "green" "Cloning GIT Repo..."
sudo yum install -y git
sudo git clone https://github.com/kodekloudhub/learning-app-ecommerce.git /var/www/html/

# Replace database IP with localhost
sudo sed -i 's/172.20.1.101/localhost/g' /var/www/html/index.php

print_color "green" "All set."

```

In future, you can add new lines of code inside the function if you need to have multiple colour options.

In above version 5 we are checking the firewall status only, we have to check the status of all the services we are installing. To do that we have to add the if else block for other services as well. We have to just replace the variables with mariadb. But it is like duplicating the code.

```
# Install and configure MariaDB
print_color "green" "Installing MariaDB..."
sudo yum install -y mariadb-server
sudo service mariadb start
sudo systemctl enable mariadb

is_mariadb_active=$(systemctl is-active mariadb)

if [ $is_mariadb_active = "active" ]
then
    print_color "green" "MariaDB Service is active"
else
    print_color "red" "MariaDB Service is not active"
    exit 1
fi
```

These two if else blocks of code are identical except for the service names. What we can do is convert that into another function. We can simply pass the service name to that function.

```
function check_service_status(){

    is_service_active=$(systemctl is-active $1)

    if [ $is_service_active = "active" ]
    then
        print_color "green" "$1 Service is active"
    else
        print_color "red" "$1 Service is not active"
        exit 1
    fi
}
```

#### Checking:

```
[bob@caleston-lp10 ~]$ function check_service_status(){
>
>     is_service_active=$(systemctl is-active $1)
>
>     if [ $is_service_active = "active" ]
>     then
>         print_color "green" "$1 Service is active"
>     else
>         print_color "red" "$1 Service is not active"
>         exit 1
>     fi
>
> }
[bob@caleston-lp10 ~]$
[bob@caleston-lp10 ~]$ check_service_status firewalld
firewalld Service is active
[bob@caleston-lp10 ~]$
[bob@caleston-lp10 ~]$ check_service_status firewalldd
firewalldd Service is not active
exit
[bob@caleston-lp10 ~]$
```

Since, it is working fine. Now, let's modify the Script. Remove the two if else blocks for checking the service status instead just write this one line and it will check and print the service status based on its status in either red or green colour.

#### Version 6 – Multiple functions for colour & checking service status

```
function print_color(){

    case $1 in
        "green") COLOR="\033[0;32m" ;;
        "red") COLOR="\033[0;31m" ;;
        "*") COLOR="\033[0m" ;;
    esac

    echo -e "${COLOR} $2 ${NC}"

}
```

```
function check_service_status(){
    is_service_active=$(systemctl is-active $1)
    if [ $is_service_active = "active" ]
    then
        print_color "green" "$1 Service is active"
    else
        print_color "red" "$1 Service is not active"
        exit 1
    fi
}
```

```
# ----- Database Configuration -----
# Install and configure FirewallD
print_color "green" "Installing firewalld..."

sudo yum install -y firewalld
sudo service firewalld start
sudo systemctl enable firewalld

check_service_status firewalld

# Install and configure MariaDB
print_color "green" "Installing MariaDB..."
sudo yum install -y mariadb-server
sudo service mariadb start
sudo systemctl enable mariadb
check_service_status mariadb
```

All the commands will be same in between and after these two sections.

```
# Start and enable httpd service
print_color "green" "Starting web server..."
sudo service httpd start
sudo systemctl enable httpd

check_service_status httpd
```

**Checking firewalld rules are configured correctly or not:**

First, run the firewall cmd command to check all the rules in the public zone:

```
[bob@caleston-lp10 ~]$ sudo firewall-cmd --list-all --zone=public
public
target: default
icmp-block-inversion: no
interfaces:
sources:
services: dhcpv6-client ssh
ports: 3306/tcp 80/tcp
protocols:
masquerade: no
forward-ports:
source-ports:
icmp-blocks:
rich rules:

[bob@caleston-lp10 ~]$
```

We will check the Ports line from above output. We can convert this into a check by assigning the output of the command to a variable and then checking if that variable contains given port.

```
[bob@caleston-lp10 ~]$ sudo firewall-cmd --list-all --zone=public | grep ports
ports: 3306/tcp 80/tcp
forward-ports:
source-ports:
[bob@caleston-lp10 ~]$
[bob@caleston-lp10 ~]$ firewallld_ports=$(sudo firewall-cmd --list-all --zone=public | grep ports)
[bob@caleston-lp10 ~]$ echo $firewalld_ports
ports: 3306/tcp 80/tcp forward-ports: source-ports:
[bob@caleston-lp10 ~]$
```



Now, let's include this one in the script. And also add the condition to check whether the port number we require is there or not.

```
# Add FirewallD rules for database
print_color "green" "Adding Firewall rules for DB..."
sudo firewall-cmd --permanent --zone=public --add-port=3306/tcp
sudo firewall-cmd --reload

firewalld_ports=$(sudo firewall-cmd --list-all --zone=public | grep ports)

if [[ $firewalld_ports = *3306* ]]
then
    print_color "green" "Port 3306 configured"
else
    print_color "red" "Port 3306 not configured"
    exit 1
fi
```

#### Checking:

```
[bob@caleston-lp10 ~]$ firewalld_ports=$(sudo firewall-cmd --list-all --zone=public | grep ports)
[bob@caleston-lp10 ~]$
[bob@caleston-lp10 ~]$ if [[ $firewalld_ports = *3306* ]]
> then
>   print_color "green" "Port 3306 configured"
> else
>   print_color "red" "Port 3306 not configured"
>   exit 1
> fi
Port 3306 configured
[bob@caleston-lp10 ~]$
```

In the same script we are configuring other firewall rules for other ports as well. So, it is better to convert the above block of code into a function for better re-usability.

```
function is_firewalld_rule_configured(){

    firewalld_ports=$(sudo firewall-cmd --list-all --zone=public | grep ports)

    if [[ $firewalld_ports = *$1* ]]
    then
        print_color "green" "Port $1 configured"
    else
        print_color "red" "Port $1 not configured"
        exit 1
    fi
}
```

#### Checking:

```
[bob@caleston-lp10 ~]$ function is_firewalld_rule_configured(){
>
>   firewalld_ports=$(sudo firewall-cmd --list-all --zone=public | grep ports)
>
>   if [[ $firewalld_ports = *$1* ]]
>   then
>       print_color "green" "Port $1 configured"
>   else
>       print_color "red" "Port $1 not configured"
>       exit 1
>   fi
> }
[bob@caleston-lp10 ~]$
[bob@caleston-lp10 ~]$ is_firewalld_rule_configured 3306
Port 3306 configured
[bob@caleston-lp10 ~]$
[bob@caleston-lp10 ~]$ is_firewalld_rule_configured 330666
Port 330666 not configured
exit
[bob@caleston-lp10 ~]$
```

Now, updated script for checking rules will be like this:

```
# Add FirewallD rules for database
print_color "green" "Adding Firewall rules for DB..."
sudo firewall-cmd --permanent --zone=public --add-port=3306/tcp
sudo firewall-cmd --reload

is_firewalld_rule_configured 3306
```

```
# Configure Firewall rules for web server
sudo firewall-cmd --permanent --zone=public --add-port=80/tcp
sudo firewall-cmd --reload

is_firewalld_rule_configured 80
```

**Adding check for Database Inventory:** whether inventory is loaded or not.

Like we will check whether tables are created or not and data is inserted into that table or not. For this we will have a simple SQL command to check:

```
[bob@caleston-lp10 ~]$ sudo mysql -e "use ecomdb; select * from products;"
+-----+-----+-----+-----+
| id | Name      | Price | ImageUrl |
+-----+-----+-----+-----+
| 1 | Laptop    | 100   | c-1.png  |
| 2 | Drone     | 200   | c-2.png  |
| 3 | VR        | 300   | c-3.png  |
| 4 | Tablet    | 50    | c-5.png  |
| 5 | Watch     | 90    | c-6.png  |
| 6 | Phone Covers | 20   | c-7.png  |
| 7 | Phone     | 80    | c-8.png  |
| 8 | Laptop    | 150   | c-4.png  |
+-----+-----+-----+-----+
[bob@caleston-lp10 ~]$
```

Let's write an IF Conditional block to check it is working as expected or not.

```
# Load inventory data into Database
print_color "green" "Loading inventory data into DB.."
cat > db-load-script.sql << EOF
USE ecomdb;
CREATE TABLE products (id mediumint(8) unsigned NOT NULL auto_increment,Name varchar(255) default NULL,Price

INSERT INTO products (Name,Price,ImageUrl) VALUES ("Laptop","100","c-1.png"),("Drone","200","c-2.png"),("VR",

EOF

sudo mysql < db-load-script.sql

mysql_db_results=$(sudo mysql -e "use ecomdb; select * from products;")

if [[ $mysql_db_results = *Laptop* ]]
then
    print_color "green" "Inventory data loaded"
else
    print_color "green" "Inventory data not loaded"
    exit 1
fi
```

**Checking:**

```
[bob@caleston-lp10 ~]$ mysql_db_results=$(sudo mysql -e "use ecomdb; select * from products;")
[bob@caleston-lp10 ~]$
[bob@caleston-lp10 ~]$ if [[ $mysql_db_results = *Laptop* ]]
> then
>   print_color "green" "Inventory data loaded"
> else
>   print_color "green" "Inventory data not loaded"
>   exit 1
> fi
Inventory data loaded
[bob@caleston-lp10 ~]$
```

**Checking Web Page is up or not from CLI:**

```
[bob@caleston-lp10 ~]$ curl http://localhost

    <div class="copy_right_area">
      <h4 class="copy_right">© Copyright 2019 Kodekloud Ecommerce | All Rights Reserved</h4>
    </div>
  </div>
</footer>

<!-- jQuery -->
<script src="js/jquery-1.12.4.min.js"></script>
<!-- Bootstrap -->
<script src="js/bootstrap.min.js"></script>
<!-- Wow js -->
<script src="vendors/wow-js/wow.min.js"></script>
<!-- Wow js -->
<script src="vendors/Counter-Up/waypoints.min.js"></script>
<script src="vendors/Counter-Up/jquery.counterup.min.js"></script>
<!-- Stellar js -->
```

Means, it is working. From here also we can check whether inventory is loaded or not. We can have a check here. Below, we are checking two items separately.

```
# Replace database IP with localhost
sudo sed -i 's/172.20.1.101/localhost/g' /var/www/html/index.php

print_color "green" "All set."

web_page=$(curl http://localhost)

if [[ $web_page = *Laptop* ]]
then
    print_color "green" "Item Laptop is present on the web page"
else
    print_color "red" "Item Laptop is not present on the web page"
fi

if [[ $web_page = *Drone* ]]
then
    print_color "green" "Item Drone is present on the web page"
else
    print_color "red" "Item Drone is not present on the web page"
fi
```

Instead, we can also do it by creating a function and call it multiple times for checking each products:

```
function check_item(){
    if [[ $1 = *$2* ]]
    then
        print_color "green" "Item $2 is present on the web page"
    else
        print_color "red" "Item $2 is not present on the web page"
    fi
}

# Replace database IP with localhost
sudo sed -i 's/172.20.1.101/localhost/g' /var/www/html/index.php

print_color "green" "All set."

web_page=$(curl http://localhost)

check_item $web_page Laptop
check_item $web_page Drone
check_item $web_page VR
```

\$1 is the first argument = "\$web\_page" – Double quotes are missing in above pic. They are must as the we are strong HTML in web\_page it will not have one word. It will have multiple words with spaces in between. So we have to pass it with in the double quotes.

\$2 is the second argument = Laptop/Drone/VR

### Checking:

```
[bob@caleston-lp10 ~]$ function check_item(){
>
>   if [[ $1 = *$2* ]]
>   then
>       print_color "green" "Item $2 is present on the web page"
>   else
>       print_color "red" "Item $2 is not present on the web page"
>   fi
>
> }
[bob@caleston-lp10 ~]$
[bob@caleston-lp10 ~]$ web_page=$(curl http://localhost)
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
   Dload  Upload  Total    Spent    Left    Speed
100 10502    0 10502    0    0  999k      0 --:--:-- --:--:-- --:--:-- 1025k
[bob@caleston-lp10 ~]$
[bob@caleston-lp10 ~]$ check_item "$web_page" Laptop
Item Laptop is present on the web page
[bob@caleston-lp10 ~]$
[bob@caleston-lp10 ~]$ check_item "$web_page" Diapers
Item Diapers is not present on the web page
[bob@caleston-lp10 ~]$ █
```

It is working fine. Now, we are checking all the items with single commands each time. It's like repeating the commands. So, instead we can use for loop.

```
# Replace database IP with localhost
sudo sed -i 's/172.20.1.101/localhost/g' /var/www/html/index.php

print_color "green" "All set."

web_page=$(curl http://localhost)

for item in Laptop Drone VR Watch
do
    check_item "$web_page" $item
done
```

### Checking:

```
[bob@caleston-lp10 ~]$ for item in Laptop Drone VR Watch
> do
>   check_item "$web_page" $item
> done
Item Laptop is present on the web page
Item Drone is present on the web page
Item VR is present on the web page
Item Watch is present on the web page
[bob@caleston-lp10 ~]$
```

Now, we can just add Shebang at the top of the file and some comment blocks to describe the each function in the script.

```
#!/bin/bash
#
# This script automates the deployment of KodeKloud e-commerce application
# Author: Mumshad Mannambeth
#####
# Print a given message in color
# Arguments:
#   Color.   eg: green, red
#####
function print_color(){
    NC='\033[0m' # No Color

    case $1 in
        "green") COLOR="\033[0;32m" ;;
        "red")   COLOR="\033[0;31m" ;;
        "*")    COLOR="\033[0m" ;;
    esac

    echo -e "${COLOR} $2 ${NC}"
}

#####
# Checks the status of a given service. Error and exit if not active
# Arguments:
#   Service. eg: httpd, firewall
#####
function check_service_status(){

    is_service_active=$(systemctl is-active $1)

    if [ $is_service_active = "active" ]
    then
        print_color "green" "$1 Service is active"
    else
        print_color "red" "$1 Service is not active"
        exit 1
    fi
}
```



```
#####
# Check if a port is enabled in a firewall rule
# Arguments:
#   Port. eg: 3306, 80
#####
function is_firewalld_rule_configured(){

    firewallld_ports=$(sudo firewall-cmd --list-all --zone=public | grep ports)


    if [[ $firewalld_ports = *$1* ]]
    then
        print_color "green" "Port $1 configured"
    #####
# Check if an Item is present in a given web page
# Arguments:
#   Webpage
#   Item
#####
function check_item(){

    if [[ $1 = *$2* ]]
    then
        print_color "green" "Item $2 is present on the web page"
    else
        print_color "red" "Item $2 is not present on the web page"
    fi
}
```


Now to let's test the full script. Create a new CentOS machine or delete and remove every package installed manually.

Create a deploy.sh file and copy all the script into that and then give executable permissions to that file. And execute the file.


**KODEKLOUD**
Laptops
Drones
Gadgets
Phones
VR
Contact us



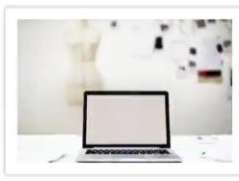
**Watch**  
Purchase Watch at the lowest price  
90\$




**Phone Covers**  
Purchase Phone Covers at the lowest price  
20\$




**Phone**  
Purchase Phone at the lowest price  
80\$




**Laptop**  
Purchase Laptop at the lowest price  
100\$



**Drone**  
Purchase Drone at the lowest price  
200\$



**VR**  
Purchase VR at the lowest price  
300\$



**Tablet**  
Purchase Tablet at the lowest price  
50\$