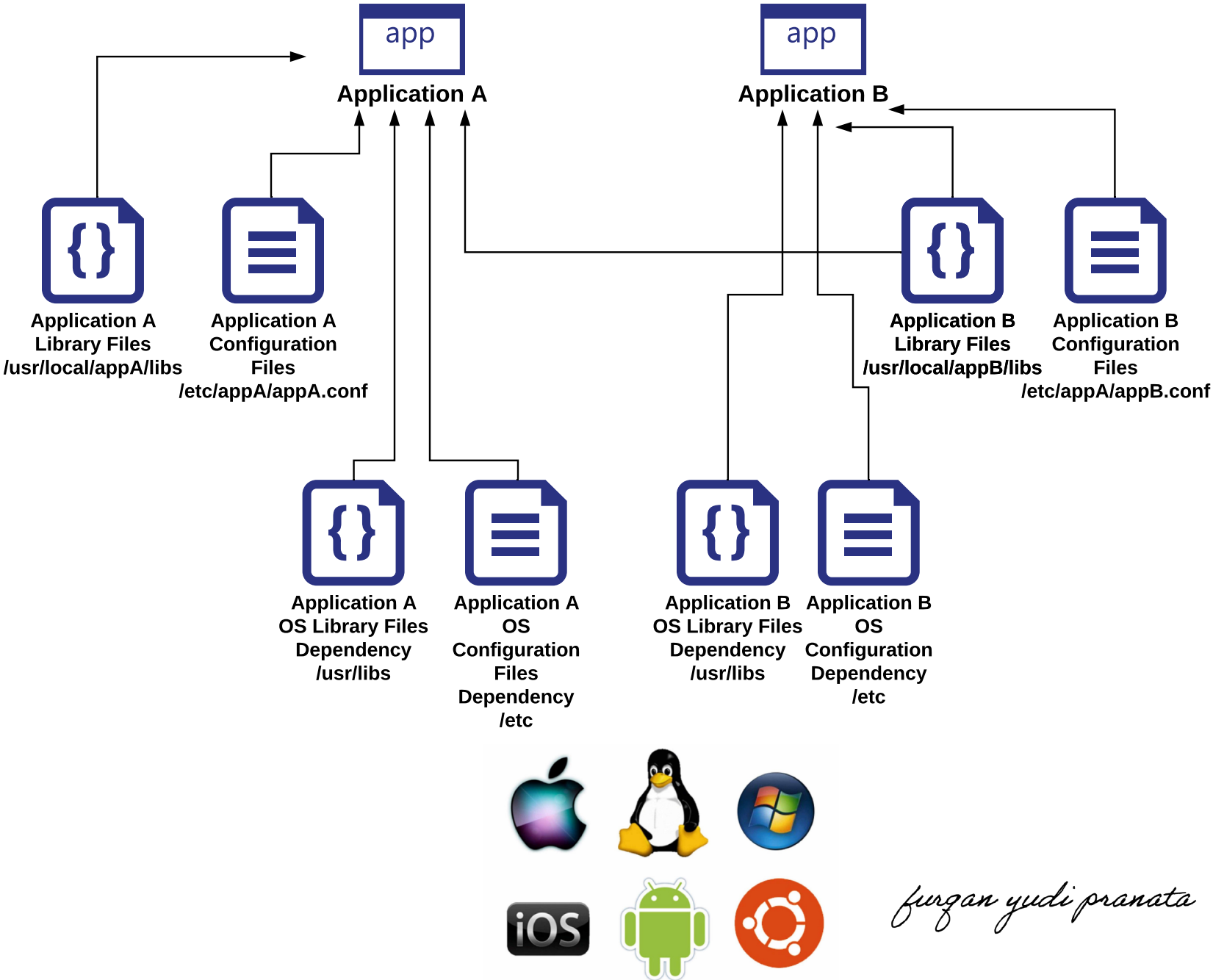
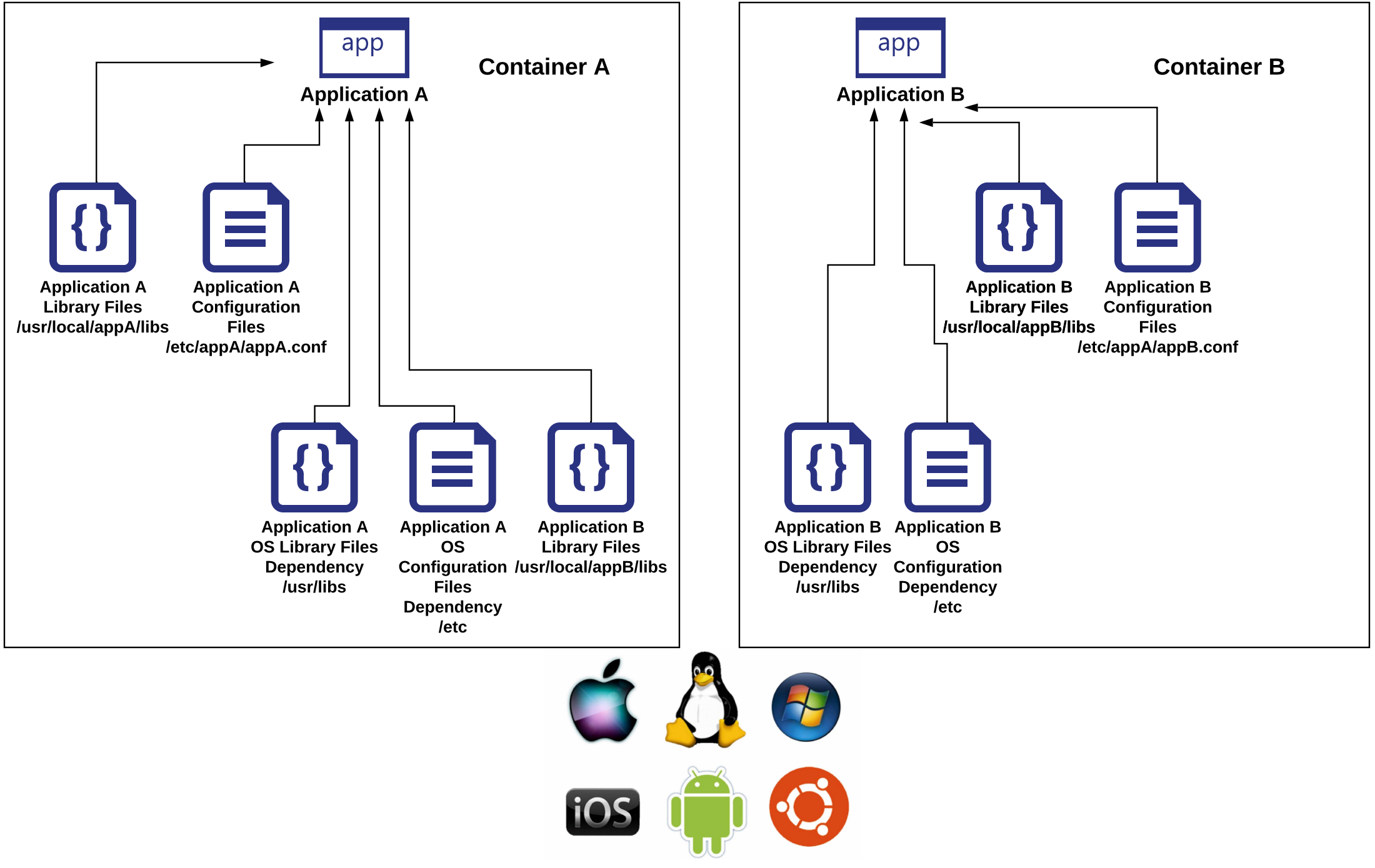
Traditional OS vs Container: Files Dependencies



Traditional OS vs Container: Files Dependencies



Comparison of Deploying Traditional App vs Stand Alone Containerized App vs Clustered Containerized App with Kubernetes/OpenShift

Traditional MariaDB Deployment

Install MariaDB Software Packages

```
yum group install mariadb mariadb-client
```

Main packages:
MySQL-python.x86_64 0:1.2.5-1.el7
mariadb.x86_64 1:5.5.64-1.el7
mariadb-server.x86_64 1:5.5.64-1.el7
mysql-connector-odbc.x86_64 0:5.2.5-8.el7

Dependency Packages:
perl-DBD-MySQL.x86_64 0:4.023-6.el7
unixODBC.x86_64 0:2.3.1-14.el7

Enable and Start MariaDB Service

```
systemctl enable mariadb
systemctl start mariadb
systemctl status mariadb
```

Create Database and User in MariaDB for WordPress

mysql -u root -p
SQL> CREATE DATABASE wordpress;
SQL> CREATE USER wordpress@localhost IDENTIFIED BY 'redhat';
SQL> GRANT ALL PRIVILEGES ON wordpress.* TO wordpress@localhost IDENTIFIED BY 'redhat';
SQL> FLUSH PRIVILEGES;

Use Physical Networking

Storage is always *persistent*

Stand Alone Containerized MariaDB

Pull MariaDB Image

```
docker image pull mariadb
docker image ls
```

Run MariaDB Container

```
docker container run -d --name wordpressdb \
-e MYSQL_ROOT_PASSWORD='redhat' \
-e MYSQL_DATABASE='wordpress' \
-e MYSQL_USER='wordpress' \
-e MYSQL_PASSWORD='redhat' \
-v /root/var/lib/mysql:/var/lib/mysql \
--network mariadb-wp-privnet \
-p 3306:3306
mariadb
```

Use Virtual Network (Software Defined Network)

```
docker network create --attachable mariadb-wp-privnetc
```

Create Persistent Storage/Volume

Default storage is *ephemeral*

```
mkdir /root/var/lib/mysql
mount /dev/sdd
/root/var/lib/mysql
```

Clustered Containerized MariaDB with Kubernetes/OpenShift

Deploy Containerized MariaDB in Kubernetes

```
kubectl apply -f mariadb-deployment.yaml
```

Deploy Containerized MariaDB in OpenShift

```
oc create -f mariadb-deployment.yaml
```

Run as 3 containers on different worker servers

Image Name

The `mariadb-deployment.yaml` File

```
apiVersion: v1
kind: Deployment
metadata:
  name: wordpressdb-deploy
  labels:
    app: wordpress
spec:
  replicas: 3
  selector:
    matchLabels:
      app: wordpress
      tier: wordpressdb
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: wordpress
        tier: wordpressdb
    spec:
      containers:
      - image: mariadb:10.5.2-bionic
        name: wordpressdb-container
        env:
          # Use secret in real usage
        - name: MYSQL_ROOT_PASSWORD
          value: redhat
        - name: MYSQL_USER
          value: wordpress
        - name: MYSQL_PASSWORD
          value: redhat
        - name: MYSQL_DATABASE
          value: wordpress
        ports:
        - containerPort: 3306
          name: wordpressdb-svc
        volumeMounts:
        - name: wordpressdb-persistent-storage
          mountPath: /var/lib/mysql
      volumes:
      - name: wordpressdb-persistent-storage
        persistentVolumeClaim:
          claimName: wordpressdb-pvc
```

Deploying WordPress as Traditional Application on Non-clustered Environment

1. Install and Update CentOS 7

Download the ISO file of latest CentOS from available mirror site then install it on Virtual Host or Physical host. After finishing the install, update the Centos 7 software using the following command:

```
# yum clean all
# yum repolist
# yum update
```

For **reducing complexity,** in this tutorial I use the following scenario:

- login as root to run all command
- change **SELinux** setting to *permissive*
- *disable* **Dynamic Firewall**

Use the following command to change SELinux settings and disable Dynamic Firewall:

```
# setenforce 0
# getenforce
# sestatus
# vi /etc/selinux/config
  SELINUX=permissive
# systemctl disable firewalld
# systemctl stop firewalld
```

2. Install and Configure Apache Web Server with PHP Support

Use the following command to install Apache Web Server (*httpd*) and PHP:

```
# yum group list -v hidden |grep -i web
…
Maybe run: yum groups mark convert (see man yum)
   Basic Web Server (web-server-environment)
   Perl for Web (perl-web)
   Python (python-web)
   Web Server (web-server)
   Web Servlet Engine (web-servlet)

# yum group install web-server
…
Installed:
  crypto-utils.x86_64 0:2.4.1-42.el7                    httpd.x86_64 0:2.4.6-90.el7.centos
  httpd-manual.noarch 0:2.4.6-90.el7.centos             mod_fcgid.x86_64 0:2.3.9-6.el7
  mod_ssl.x86_64 1:2.4.6-90.el7.centos

Dependency Installed:
apr.x86_64 0:1.4.8-5.el7        apr-util.x86_64 0:1.5.2-6.el7    httpd-tools.x86_64 0:2.4.6-90.el7.centos
mailcap.noarch 0:2.1.41-2.el7   perl-Newt.x86_64 0:1.08-36.el7

Complete!

# yum group install php
…
Installed:
php.x86_64 0:5.4.16-46.1.el7_7     php-gd.x86_64 0:5.4.16-46.1.el7_7      php-pdo.x86_64 0:5.4.16-46.1.el7_7
php-pear.noarch 1:1.9.4-21.el7     php-xml.x86_64 0:5.4.16-46.1.el7_7
```

```
Dependency Installed:
libzip.x86_64 0:0.10.1-8.el7            php-cli.x86_64 0:5.4.16-46.1.el7_7      php-common.x86_64 0:5.4.16-46.1.el7_7
php-process.x86_64 0:5.4.16-46.1.el7_7  t1lib.x86_64 0:5.1.2-14.el7

Complete!

# yum install php-mysql
…
Installed:
  php-mysql.x86_64 0:5.4.16-46.1.el7_7

Complete!
```

3. Install MariaDB

Use the following command to install MariaDB Database software:

```
# yum group list -v hidden|grep -i maria
…
  MariaDB Database Client (mariadb-client)
  MariaDB Database Server (mariadb)

# yum group install mariadb mariadb-client
…
Installed:
  MySQL-python.x86_64 0:1.2.5-1.el7     mariadb.x86_64 1:5.5.64-1.el7    mariadb-server.x86_64 1:5.5.64-1.el7
  mysql-connector-odbc.x86_64 0:5.2.5-8.el7

Dependency Installed:
  perl-DBD-MySQL.x86_64 0:4.023-6.el7           unixODBC.x86_64 0:2.3.1-14.el7

Complete!
```

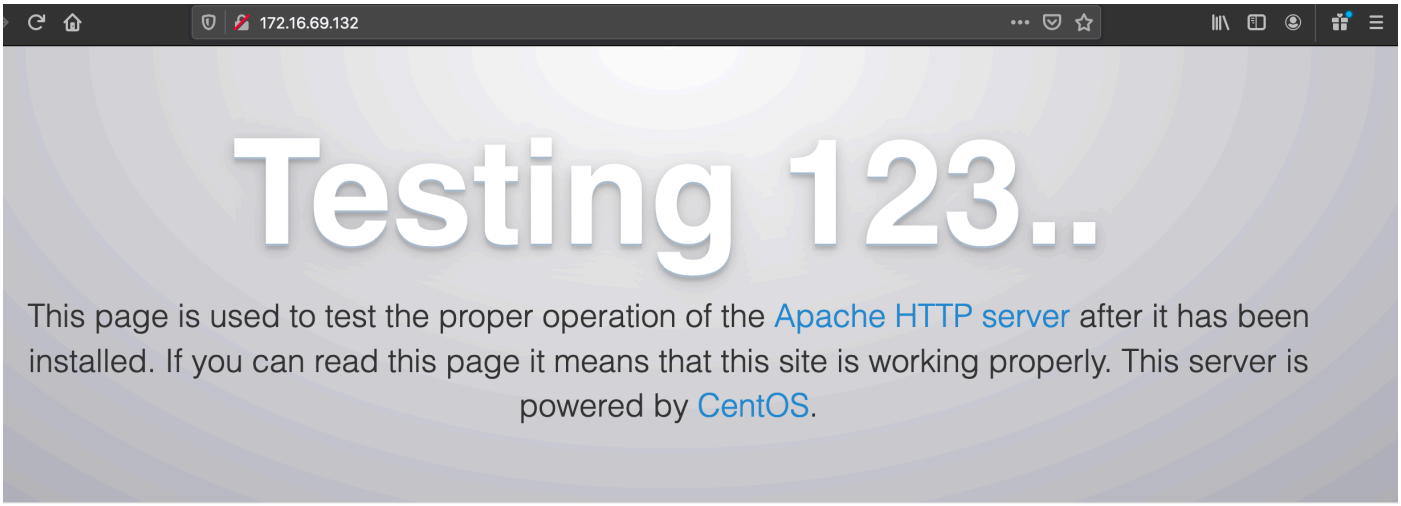
4. Enable, Start and Test Apache Web Functionality

Use the following command to enable and start Apache service:

```
# rpm -qa|grep httpd
# rpm -ql httpd|grep systemd
/etc/httpd/conf.modules.d/00-systemd.conf
/usr/lib/systemd/system/htcacheclean.service
/usr/lib/systemd/system/httpd.service
/usr/lib64/httpd/modules/mod_systemd.so

# systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor preset: disabled)
   Active: inactive (dead)
     Docs: man:httpd(8)
           man:apachectl(8)

# systemctl status htcacheclean
● htcacheclean.service - Disk Cache Cleaning Daemon for Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/htcacheclean.service; static; vendor preset:
disabled)
   Active: inactive (dead)
     Docs: man:htcacheclean(8)

# systemctl enable httpd
Created symlink from /etc/systemd/system/multi-user.target.wants/httpd.service to
/usr/lib/systemd/system/httpd.service.
```

```
# systemctl start httpd

# systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
   Active: active (running) since Mon 2020-04-06 08:44:48 WIB; 4s ago
     Docs: man:httpd(8)
           man:apachectl(8)
 Main PID: 3909 (httpd)
   Status: "Processing requests..."
    Tasks: 7
   CGroup: /system.slice/httpd.service
           ├─3909 /usr/sbin/httpd -DFOREGROUND
           ├─3910 /usr/sbin/httpd -DFOREGROUND
           ├─3912 /usr/sbin/httpd -DFOREGROUND
           ├─3913 /usr/sbin/httpd -DFOREGROUND
           ├─3914 /usr/sbin/httpd -DFOREGROUND
           ├─3915 /usr/sbin/httpd -DFOREGROUND
           └─3916 /usr/sbin/httpd -DFOREGROUND

Apr 06 08:44:41 myserver1.mydomain.com systemd[1]: Starting The Apache HTTP Server...
Apr 06 08:44:47 myserver1.mydomain.com httpd[3909]: AH00558: httpd: Could not reliably
determine the server'...sage
Apr 06 08:44:48 myserver1.mydomain.com systemd[1]: Started The Apache HTTP Server.
Hint: Some lines were ellipsized, use -l to show in full.
```

Use web browser or *curl* to test access to the web server:




```
$ curl http://localhost|grep -i test
  …
            <title>Apache HTTP Server Test Page powered by CentOS</title>
        <h1>Testing 123..</h1>
            <p class="lead">This page is used to test the proper operation of the <a
href="http://apache.org">Apache HTTP server</a> after it has been installed. If you can read
this page it means that this site is working properly. This server is powered by <a
href="http://centos.org">CentOS</a>.</p>
```

5. Enable, Start and Test PHP Functionality

PHP packages **do not have service** to start. The following command show the content of main PHP package:

```
# rpm -qa|grep php
php-5.4.16-46.1.el7_7.x86_64
…
# rpm -ql php
/etc/httpd/conf.d/php.conf
/etc/httpd/conf.modules.d/10-php.conf
/usr/lib64/httpd/modules/libphp5.so
/usr/share/httpd/icons/php.gif
/var/lib/php/session
```

Create the following PHP script and use web browser or curl to run that script:

```
# cd /var/www/html

# vi info.php
<?php phpinfo(); ?>

$ curl http://localhost/info.php|grep -i version
…
<a href="http://www.php.net/"><img border="0" src="/info.php?=PHPE9568F34-D428-11d2-A769-
00AA001ACF42" alt="PHP Logo" /></a><h1 class="p">PHP Version 5.4.16</h1>
<tr><td class="e">Apache Version </td><td class="v">Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips
mod_fcgid/2.3.9 PHP/5.4.16 </td></tr>
…
```

Here is the output from the web browser:

172.16.69.132/info.php

PHP Version 5.4.16

System	Linux myserver1.mydomain.com 3.10.0-1062.18.1.el7.x86_64 #1 SMP Tue Mar 17 23:49:17 UTC 2020 x86_64
Build Date	Nov 1 2019 16:05:03
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc
Loaded Configuration File	/etc/php.ini
Scan this dir for additional .ini files	/etc/php.d
Additional .ini files parsed	/etc/php.d/curl.ini, /etc/php.d/dom.ini, /etc/php.d/fileinfo.ini, /etc/php.d/gd.ini, /etc/php.d/json.ini, /etc/php.d/mysql.ini, /etc/php.d/mysqli.ini, /etc/php.d/pdo.ini, /etc/php.d/pdo_mysql.ini, /etc/php.d/pdo_sqlite.ini, /etc/php.d/phar.ini, /etc/php.d/posix.ini, /etc/php.d/sqlite3.ini, /etc/php.d/sysvmsg.ini, /etc/php.d/sysvsem.ini, /etc/php.d/sysvshm.ini, /etc/php.d/wddx.ini, /etc/php.d/xmlreader.ini, /etc/php.d/xmlwriter.ini, /etc/php.d/xsl.ini, /etc/php.d/zip.ini
PHP API	20100412
PHP Extension	20100525
Zend Extension	220100525
Zend Extension Build	API220100525,NTS

6. Enable, Start and Configure MariaDB Service

Use the following command to enable, start and configure MariaDB service:

```
# rpm -qa|grep mariadb
mariadb-5.5.64-1.el7.x86_64
mariadb-libs-5.5.64-1.el7.x86_64
mariadb-server-5.5.64-1.el7.x86_64

# rpm -ql mariadb-server|grep systemd
/usr/lib/systemd/system/mariadb.service

# systemctl status mariadb
● mariadb.service - MariaDB database server
   Loaded: loaded (/usr/lib/systemd/system/mariadb.service; disabled; vendor preset: disabled)
   Active: inactive (dead)

# systemctl enable mariadb
Created symlink from /etc/systemd/system/multi-user.target.wants/mariadb.service to
/usr/lib/systemd/system/mariadb.service.

# systemctl start mariadb

# systemctl status mariadb
● mariadb.service - MariaDB database server
   Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; vendor preset: disabled)
   Active: active (running) since Mon 2020-04-06 09:23:27 WIB; 5s ago
  Process: 4696 ExecStartPost=/usr/libexec/mariadb-wait-ready $MAINPID (code=exited,
status=0/SUCCESS)
  Process: 4609 ExecStartPre=/usr/libexec/mariadb-prepare-db-dir %n (code=exited,
status=0/SUCCESS)
 Main PID: 4695 (mysqld_safe)
    Tasks: 20
   CGroup: /system.slice/mariadb.service
           ├─4695 /bin/sh /usr/bin/mysqld_safe --basedir=/usr
           └─4857 /usr/libexec/mysqld --basedir=/usr --datadir=/var/lib/mysql --plugin-
dir=/usr/lib64/mysql/plug...

Apr 06 09:23:25 myserver1.mydomain.com mariadb-prepare-db-dir[4609]: MySQL manual for more
instructions.
Apr 06 09:23:25 myserver1.mydomain.com mariadb-prepare-db-dir[4609]: Please report any problems
at http://mari...ra
Apr 06 09:23:25 myserver1.mydomain.com mariadb-prepare-db-dir[4609]: The latest information
about MariaDB is a.../.
Apr 06 09:23:25 myserver1.mydomain.com mariadb-prepare-db-dir[4609]: You can find additional
information about...t:
Apr 06 09:23:25 myserver1.mydomain.com mariadb-prepare-db-dir[4609]: http://dev.mysql.com
Apr 06 09:23:25 myserver1.mydomain.com mariadb-prepare-db-dir[4609]: Consider joining MariaDB's
strong and vib...y:
Apr 06 09:23:25 myserver1.mydomain.com mariadb-prepare-db-dir[4609]: https://mariadb.org/get-
involved/
Apr 06 09:23:25 myserver1.mydomain.com mysqld_safe[4695]: 200406 09:23:25 mysqld_safe Logging
to '/var/log/ma...g'.
Apr 06 09:23:25 myserver1.mydomain.com mysqld_safe[4695]: 200406 09:23:25 mysqld_safe Starting
mysqld daemon ...sql
Apr 06 09:23:27 myserver1.mydomain.com systemd[1]: Started MariaDB database server.
Hint: Some lines were ellipsized, use -l to show in full.

# mysql_secure_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
      SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
```

```
password for the root user.  If you've just installed MariaDB, and
you haven't set the root password yet, the password will be blank,
so you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password ensures that nobody can log into the MariaDB
root user without the proper authorisation.

Set root password? [Y/n] y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
 ... Success!


By default, a MariaDB installation has an anonymous user, allowing anyone
to log into MariaDB without having to have a user account created for
them.  This is intended only for testing, and to make the installation
go a bit smoother.  You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] y
 ... Success!

Normally, root should only be allowed to connect from 'localhost'.  This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] y
 ... Success!

By default, MariaDB comes with a database named 'test' that anyone can
access.  This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] y
 - Dropping test database...
 ... Success!
 - Removing privileges on test database...
 ... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] y
 ... Success!

Cleaning up...

All done!  If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!
```

7. Create Database and User in MariaDB for WordPress

Use the following command to create database and user in MariaDB to be accessed by WordPress:

```
# mysql -u root -p
Enter password:
…
MariaDB [(none)]> create database wordpress;
Query OK, 1 row affected (0.01 sec)
MariaDB [(none)]> create user wordpress@localhost identified by 'redhat';
Query OK, 0 rows affected (0.01 sec)
MariaDB [(none)]> grant all privileges on wordpress.* to wordpress@localhost identified by
'redhat';
Query OK, 0 rows affected (0.00 sec)
MariaDB [(none)]> flush privileges;
Query OK, 0 rows affected (0.00 sec)
MariaDB [(none)]> exit
Bye
```

8. Download, Install and Create Initial Configuration of WordPress

Make sure you know your PHP version, use the following command:

```
# php -v
PHP 5.4.16 (cli) (built: Nov  1 2019 16:04:20)
Copyright (c) 1997-2013 The PHP Group
Zend Engine v2.4.0, Copyright (c) 1998-2013 Zend Technologies
```

The compatibility of **PHP 5.4** version is **WordPress 5.0** version. Use graphical browser, *wget* or *curl* to download the WordPress software on the following URL:

https://wordpress.org/download/releases/

5.0 Branch

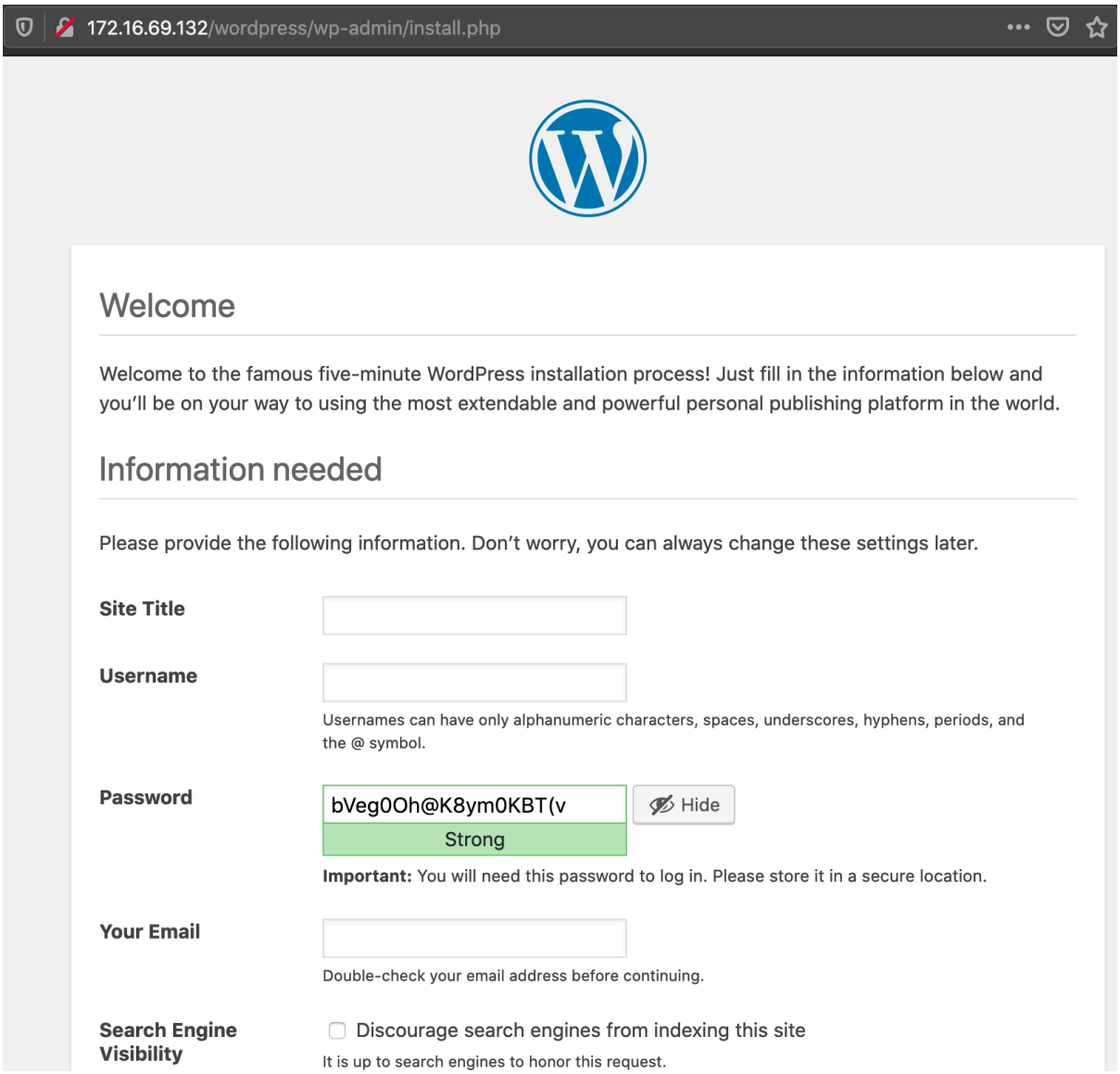
5.0.8	December 12, 2019	zip (md5 | sha1)	tar.gz (md5 | sha1)	IIS zip (md5 | sha1)
5.0.7	October 14, 2019	zip (md5 | sha1)	tar.gz (md5 | sha1)	IIS zip (md5 | sha1)
5.0.6	September 5, 2019	zip (md5 | sha1)	tar.gz (md5 | sha1)	IIS zip (md5 | sha1)
5.0.4	March 13, 2019	zip (md5 | sha1)	tar.gz (md5 | sha1)	IIS zip (md5 | sha1)
5.0.3	January 9, 2019	zip (md5 | sha1)	tar.gz (md5 | sha1)	IIS zip (md5 | sha1)
5.0.2	December 19, 2018	zip (md5 | sha1)	tar.gz (md5 | sha1)	IIS zip (md5 | sha1)
5.0.1	December 13, 2018	zip (md5 | sha1)	tar.gz (md5 | sha1)	IIS zip (md5 | sha1)

The following commands demonstrate how to download, install and create initial configuration for WordPress:

```
# cd Downloads

# wget https://wordpress.org/wordpress-5.0.8.tar.gz

# ls
wordpress-5.0.8.tar.gz

# tar xzf wordpress-5.0.8.tar.gz

# ls -l
total 10276
drwxr-xr-x. 5 nobody nfsnobody     4096 Dec 13 04:36 wordpress
-rw-r--r--. 1 root   root      10516757 Apr 12 08:34 wordpress-5.0.8.tar.gz

# cp -a wordpress /var/www/html

# cd /var/www/html/wordpress

# mkdir wp-content/uploads

# cp wp-config-sample.php wp-config.php

# vi wp-config.php
…
// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define( 'DB_NAME', 'wordpress' );

/** MySQL database username */
define( 'DB_USER', 'wordpress' );

/** MySQL database password */
define( 'DB_PASSWORD', 'redhat' );

/** MySQL hostname */
define( 'DB_HOST', 'localhost' );
…

# cd ..
# chown -R apache:apache wordpress
```

9. Access WordPress for Creating Additional Configuration

Now, WordPress application is ready on the following URL: http://hostname/wordpress for creating additional configuration.



That's all, thank you.

Deploying WordPress as Docker Containerized Application on Non-clustered Environment

1. Install and Update CentOS 7

Download an ISO file of the latest CentOS from available mirror sites then install it on Virtual Host or Physical host. After finishing the install, update the CentOS 7 software using the following command:

```
# yum clean all
# yum repolist
# yum update
```

For **reducing complexity** in this tutorial, I use the following scenario:

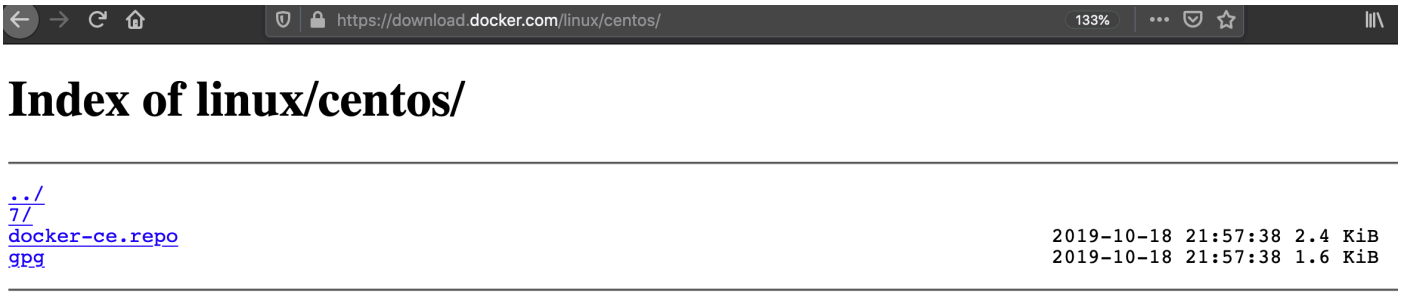
- Login as `root` to run all command
- Change **SELinux** setting to `permissive`

Use the following command to change SELinux settings and disable Dynamic Firewall:

```
# setenforce 0
# getenforce
# sestatus
# vi /etc/selinux/config
  SELINUX=permissive
```

2. Install Docker Software and Start Docker Service

Use the following command to install Docker Desktop Community Edition from Docker Repository:




```
# wget https://download.docker.com/linux/centos/docker-ce.repo

# ls
docker-ce.repo

# cp docker-ce.repo /etc/yum.repos.d/

# yum repolist
…
docker-ce-stable                                          | 3.5 kB   00:00
(1/2): docker-ce-stable/x86_64/primary_db                 |  41 kB   00:00
(2/2): docker-ce-stable/x86_64/updateinfo                 |   55 B   00:00
```

```
repo id                              repo name                              status
base/7/x86_64                        CentOS-7 - Base                        10,097
docker-ce-stable/x86_64              Docker CE Stable - x86_64                  70
extras/7/x86_64                      CentOS-7 - Extras                         341
updates/7/x86_64                     CentOS-7 - Updates                      1,787
repolist: 12,295

# yum list|grep docker
…
docker.x86_64                                2:1.13.1-109.gitcccb291.el7.centos
docker-ce.x86_64                             3:19.03.8-3.el7            docker-ce-stable
docker-ce-cli.x86_64                         1:19.03.8-3.el7            docker-ce-stable
docker-ce-selinux.noarch                     17.03.3.ce-1.el7           docker-ce-stable
…
podman-docker.noarch                         1.4.4-4.el7.centos         extras
…
# yum install docker-ce
…
Dependencies Resolved

=================================================================================================
 Package            Arch         Version             Repository          Size
=================================================================================================
Installing:
docker-ce           x86_64       3:19.03.8-3.el7     docker-ce-stable    25 M

Installing for dependencies:
container-selinux   noarch       2:2.107-3.el7       extras              39 k
containerd.io       x86_64       1.2.13-3.1.el7      docker-ce-stable    23 M
docker-ce-cli       x86_64       1:19.03.8-3.el7     docker-ce-stable    40 M

Transaction Summary
=================================================================================================
Install  1 Package (+3 Dependent packages)
…

# rpm -qa|grep docker
docker-ce-cli-19.03.8-3.el7.x86_64
docker-ce-19.03.8-3.el7.x86_64

# rpm -ql docker-ce|grep systemd
/usr/lib/systemd/system/docker.service
/usr/lib/systemd/system/docker.socket

# systemctl enable docker.service
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to
/usr/lib/systemd/system/docker.service

# systemctl start docker.service

# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
   Active: active (running) since Sun 2020-04-12 19:44:11 WIB; 6min ago
     Docs: https://docs.docker.com
 Main PID: 1426 (dockerd)
    Tasks: 12
   Memory: 130.4M
   CGroup: /system.slice/docker.service
           └─1426 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
Apr 12 19:44:09 myserver2.mydomain.com dockerd[1426]: time="2020-04-
12T19:44:09.669644698+07:00" level=info msg="ccResolverWrapper: sending update to cc:
{[{unix:///run/containerd/containerd.sock..." module=grpc
…
```

3. Test Docker Functionality

Use the following command to test Docker functionality:

```
# docker search hello
NAME         DESCRIPTION                                     STARS      OFFICIAL     AUTOMATED
hello-world  Hello World! (an example of minimal Dockeriz…   1161       [OK]
…

# docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:f9dfddf63636d84ef479d645ab5885156ae030f611a56f3a7ac7f2fdd86d7e4e
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest

# docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello-world         latest              fce289e99eb9        15 months ago       1.84kB

# docker container run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

4. Download MariaDB and WordPress Image from Docker Image Registry

Use the following commands to download MariaDB and WordPress Image from Docker Image Registry:

```
# docker search mariadb
NAME                                      DESCRIPTION                                     STARS
OFFICIAL            AUTOMATED
mariadb                                   MariaDB is a community-developed fork of MyS…   3362
[OK]

# docker pull mariadb
Using default tag: latest
latest: Pulling from library/mariadb
5bed26d33875: Pull complete
f11b29a9c730: Pull complete
930bda195c84: Pull complete
78bf9a5ad49e: Pull complete
e9e3c043ec68: Pull complete
141e45c6af4b: Pull complete
a26245908a82: Pull complete
40ccaf895c8a: Pull complete
2d665f60c94a: Pull complete
c7bcd9961bee: Pull complete
80f1ddb594ce: Pull complete
0647ec428f9f: Pull complete
9cb6e30e72ca: Pull complete
60890c0035d8: Pull complete
Digest: sha256:d0e2c681c41e91aba6e9c8c0a588eedd48291a70464e83c40da2e3de01998eef
Status: Downloaded newer image for mariadb:latest
docker.io/library/mariadb:latest

# docker search wordpress
NAME                                        DESCRIPTION                                     STARS
OFFICIAL            AUTOMATED
wordpress                                   The WordPress rich content management system…   3454
[OK]
…
# docker pull wordpress
Using default tag: latest
latest: Pulling from library/wordpress
c499e6d256d6: Pull complete
3a635b94b3b9: Pull complete
cf28be682a33: Pull complete
b7118ab6e551: Pull complete
925f628a16b8: Pull complete
a77cff9973b5: Pull complete
..
6b1397000eb2: Pull complete
7b387d8d3957: Pull complete
04673b988ee3: Pull complete
0e2da6305da6: Pull complete
f0224352bc00: Pull complete
d5e8b4e26a84: Pull complete
Digest: sha256:191d5caf4ef5b8c57721ade777820f3267654325f7902b2ccd377ceeba1c3fe2
Status: Downloaded newer image for wordpress:latest
docker.io/library/wordpress:latest

# docker image ls
REPOSITORY                    TAG                 IMAGE ID            CREATED             SIZE
wordpress                     latest              0d205d4886fe        11 days ago         540MB
mariadb                       latest              37f5f0a258bf        3 weeks ago         356MB
hello-world                   latest              fce289e99eb9        15 months ago      1.84kB
```

5. Create Persistent Volume for MariaDB and WordPress

Use the following commands to create *persistent volume* for both MariaDB and WordPress:

```
# mkdir -p /root/var/lib/mysql

# mkdir -p /root/var/www/html

# ls -lR var
var:
total 0
drwxr-xr-x. 3 root root 19 Apr 13 06:38 lib
drwxr-xr-x. 3 root root 18 Apr 13 06:38 www

var/lib:
total 0
drwxr-xr-x. 2 root root 6 Apr 13 06:38 mysql

var/lib/mysql:
total 0

var/www:
total 0
drwxr-xr-x. 2 root root 6 Apr 13 06:38 html

var/www/html:
total 0
```

6. Create Virtual Private Network for WordPress and MariaDB Container

Use the following commands to create Docker *virtual private network*:

```
# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
ceb9d2e9cbad        bridge              bridge              local
9d3c1b497be8        host                host                local
9da389311d9e        none                null                local

# docker network create --attachable mariadb-wp-privnet
90f36038fcaf6c19c598d0a7a6ddcc902d0af6f59dd2cdfed1e1df2d35eff02e

# docker network ls
NETWORK ID          NAME                 DRIVER              SCOPE
ceb9d2e9cbad        bridge               bridge              local
9d3c1b497be8        host                 host                local
90f36038fcaf        mariadb-wp-privnet   bridge              local
9da389311d9e        none                 null                local
```

7. Run MariaDB and WordPress Container

For the purpose of running MariaDB and WordPress container, create the following shell scripts:

```
# vi run_mariadb_with_persistent_volume_and_private_network.sh
#!/bin/bash

docker container run -d \
--name wordpressdb \
-e MYSQL_ROOT_PASSWORD='redhat' \
-e MYSQL_DATABASE='wordpress' \
-e MYSQL_USER='wordpress' \
-e MYSQL_PASSWORD='redhat' \
-v /root/var/lib/mysql:/var/lib/mysql \
--network mariadb-wp-privnet \
mariadb

# vi run_wordpress_with_persistent_volume_and_published_port.sh
#!/bin/bash

docker container run -d \
--name wordpress \
-e WORDPRESS_DB_HOST=wordpressdb \
-e WORDPRESS_DB_USER='wordpress' \
-e WORDPRESS_DB_PASSWORD='redhat' \
-e WORDPRESS_DB_NAME='wordpress' \
-v /root/var/www/html:/var/www/html \
--network mariadb-wp-privnet  \
-p 80:80 \
wordpress

# chmod +x run_mariadb_with_persistent_volume_and_private_network.sh
# chmod +x run_wordpress_with_persistent_volume_and_published_port.sh

# ./run_mariadb_with_persistent_volume_and_private_network.sh
b596071feb0035fa3590860bc979a0769141a2de8b080235722ce0b2f1b3345c

# ./run_wordpress_with_persistent_volume_and_published_port.sh
10f0489842c8887aba174399af3cb308885ae32cdc3e0603638192f250eee48a

# docker container ls
CONTAINER ID   IMAGE       COMMAND                  CREATED         STATUS        PORTS                 NAMES
10f0489842c8   wordpress  "docker-entrypoint.s…"    2 seconds ago  Up 2 seconds  0.0.0.0:80->80/tcp  wordpress
b596071feb00   mariadb     "docker-entrypoint.s…"  28 second ago   Up 28 seconds 3306/tcp            wordpressdb
```

8. Open HTTP Port in Dynamic Firewall and Access WordPress

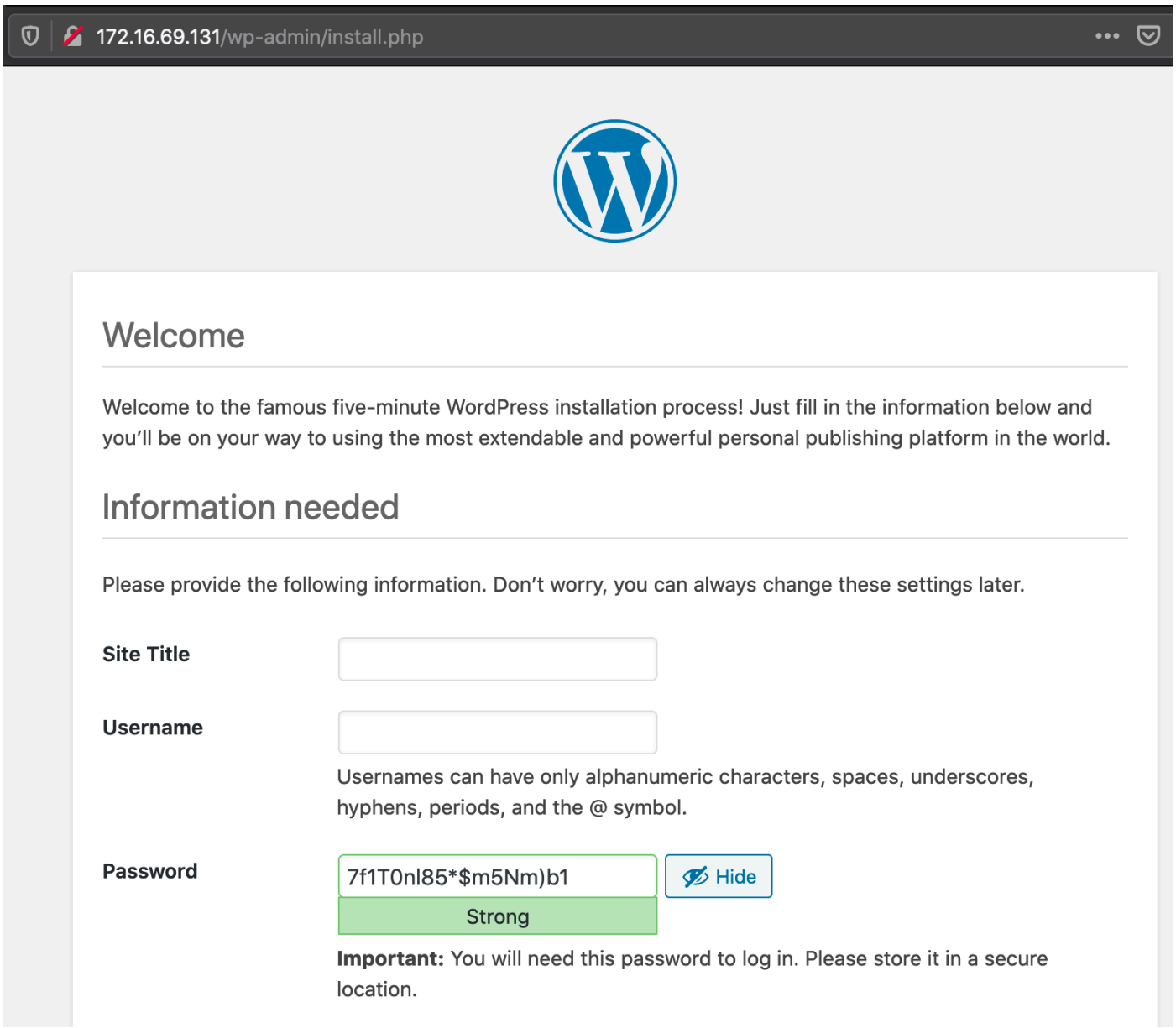
Use the following command to open HTTP port in Dynamic Firewall:

```
# firewall-cmd --get-active-zones
public
  interfaces: ens33

# firewall-cmd --list-services
dhcpv6-client ssh

# firewall-cmd --permanent --add-service=http

# firewall-cmd –reload

# firewall-cmd --list-services
dhcpv6-client http ssh
```

To perform simple accessibility testing of WordPress, from localhost run the following command:

```
# curl -L http://localhost|grep -i welcome
…
        <h1>Welcome</h1>
<p>Welcome to the famous five-minute WordPress installation process! Just fill in the
information below and you&#8217;ll be on your way to using the most extendable and powerful
personal publishing platform in the world.</p>
…
```

Finally access WordPress using graphical web browser:



That's all, thank you.

Deploying WordPress as Docker Containerized Application on Clustered Environment with Kubernetes

1. Install and Update CentOS 7

Download an ISO file of the latest CentOS from available mirror sites then install it on Virtual Host or Physical host. After finishing the install, update the CentOS 7 software using the following command:

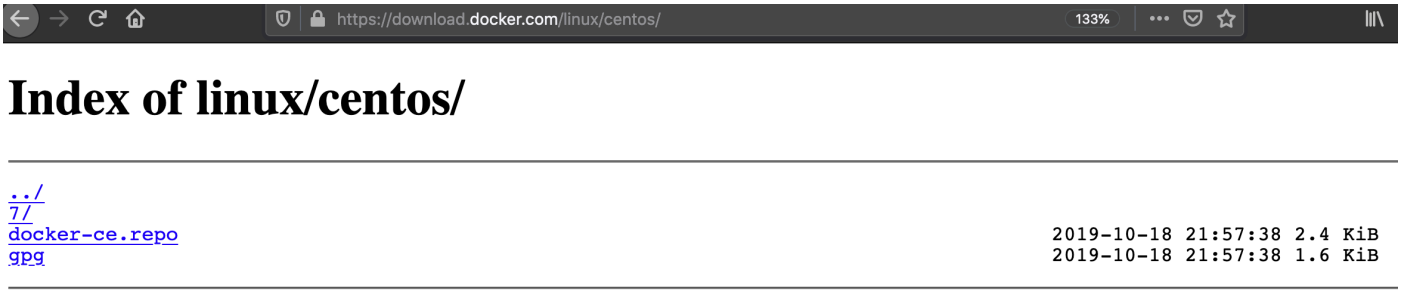
```
# yum clean all
# yum repolist
# yum update
```

For **reducing complexity** in this tutorial, set SELinux setting to `permissive`. Use the following command to change SELinux settings:

```
# setenforce 0
# getenforce
# sestatus
# vi /etc/selinux/config
  SELINUX=permissive
```

2. Install Docker Software and Start Docker Service

Use the following command to install Docker Desktop Community Edition from Docker Repository:




```
# wget https://download.docker.com/linux/centos/docker-ce.repo

# ls
docker-ce.repo

# cp docker-ce.repo /etc/yum.repos.d/

# yum repolist
…
docker-ce-stable                                          | 3.5 kB   00:00
(1/2): docker-ce-stable/x86_64/primary_db                 |  41 kB   00:00
(2/2): docker-ce-stable/x86_64/updateinfo                 |   55 B   00:00
```

```
repo id                              repo name                              status
base/7/x86_64                        CentOS-7 - Base                        10,097
docker-ce-stable/x86_64              Docker CE Stable - x86_64                  70
extras/7/x86_64                      CentOS-7 - Extras                         341
updates/7/x86_64                     CentOS-7 - Updates                      1,787
repolist: 12,295

# yum list|grep docker
…
docker.x86_64                                2:1.13.1-109.gitcccb291.el7.centos
docker-ce.x86_64                             3:19.03.8-3.el7            docker-ce-stable
docker-ce-cli.x86_64                         1:19.03.8-3.el7            docker-ce-stable
docker-ce-selinux.noarch                     17.03.3.ce-1.el7           docker-ce-stable
…
podman-docker.noarch                         1.4.4-4.el7.centos         extras
…
# yum install docker-ce
…
Dependencies Resolved

=================================================================================================
 Package            Arch         Version             Repository          Size
=================================================================================================
Installing:
docker-ce           x86_64       3:19.03.8-3.el7     docker-ce-stable    25 M

Installing for dependencies:
container-selinux   noarch       2:2.107-3.el7       extras              39 k
containerd.io       x86_64       1.2.13-3.1.el7      docker-ce-stable    23 M
docker-ce-cli       x86_64       1:19.03.8-3.el7     docker-ce-stable    40 M

Transaction Summary
=================================================================================================
Install  1 Package (+3 Dependent packages)
…

# rpm -qa|grep docker
docker-ce-cli-19.03.8-3.el7.x86_64
docker-ce-19.03.8-3.el7.x86_64

# rpm -ql docker-ce|grep systemd
/usr/lib/systemd/system/docker.service
/usr/lib/systemd/system/docker.socket

# systemctl enable docker.service
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to
/usr/lib/systemd/system/docker.service

# systemctl start docker.service

# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
   Active: active (running) since Sun 2020-04-12 19:44:11 WIB; 6min ago
     Docs: https://docs.docker.com
 Main PID: 1426 (dockerd)
    Tasks: 12
   Memory: 130.4M
   CGroup: /system.slice/docker.service
           └─1426 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
Apr 12 19:44:09 myserver2.mydomain.com dockerd[1426]: time="2020-04-
12T19:44:09.669644698+07:00" level=info msg="ccResolverWrapper: sending update to cc:
{[{unix:///run/containerd/containerd.sock..." module=grpc
…
```

3. Test Docker Functionality

Use the following command to test Docker functionality:

```
# docker search hello
NAME         DESCRIPTION                                     STARS      OFFICIAL     AUTOMATED
hello-world  Hello World! (an example of minimal Dockeriz…   1161       [OK]
…

# docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:f9dfddf63636d84ef479d645ab5885156ae030f611a56f3a7ac7f2fdd86d7e4e
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest

# docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello-world         latest              fce289e99eb9        15 months ago       1.84kB

# docker container run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

4. Download and Install ***kubectl*** Program

Use the following command to check the current version of `kubectl` is available at Google API site:

```
# curl https://storage.googleapis.com/kubernetes-release/release/stable.txt
v1.18.1
```

Use the following command to download and install `kubectl`:

```
# cd Downloads
# wget https://storage.googleapis.com/kubernetes-release/release/v1.18.1/bin/linux/amd64/kubectl
…
Connecting to storage.googleapis.com (storage.googleapis.com)|172.253.118.128|:443...
connected.
```

```
HTTP request sent, awaiting response... 200 OK
Length: 44027904 (42M) [application/octet-stream]
Saving to: 'kubectl'

100%[===========================================================>] 44,027,904  4.37MB/s   in 9.4s

2020-04-13 14:24:38 (4.45 MB/s) - 'kubectl' saved [44027904/44027904]

# chmod +x kubectl

# ls -l
total 63168
-rwxr-xr-x. 1 root root 44027904 Apr  9 02:18 kubectl

# cp kubectl /usr/local/bin

# kubectl version --client
Client Version: version.Info{Major:"1", Minor:"18", GitVersion:"v1.18.0",
GitCommit:"9e991415386e4cf155a24b1da15becaa390438d8", GitTreeState:"clean", BuildDate:"2020-03-
25T14:58:59Z", GoVersion:"go1.13.8", Compiler:"gc", Platform:"linux/amd64"}
```

5. Download and Install ***minikube***

Use the following command to download and install `minikube`:

```
# wget https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
…
Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.200.128|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 54639377 (52M) [application/octet-stream]
Saving to: 'minikube-linux-amd64'

100%[===========================================================>] 54,639,377  4.68MB/s   in 12s

2020-04-13 14:41:45 (4.50 MB/s) - 'minikube-linux-amd64' saved [54639377/54639377]

# ls
kubectl  minikube-linux-amd64

# mv minikube-linux-amd64 minikube

# chmod +x minikube

# ls -l
total 102324
-rwxr-xr-x. 1 root root 44027904 Apr  9 02:18 kubectl
-rwxr-xr-x. 1 root root 54639377 Apr  5 04:04 minikube

# cp minikube /usr/local/bin

# minikube version
minikube version: v1.9.2
commit: 93af9c1e43cab9618e301bc9fa720c63d5efa393
```

6. Create Regular User with Membership of **docker** Group

Use the following command to create user with membership of `docker` group:

```
# useradd -G docker kube

# id kube
uid=1001(kube) gid=1001(kube) groups=1001(kube),982(docker)

# su - kube

$ id
uid=1001(kube) gid=1001(kube) groups=1001(kube),982(docker)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023

$ docker search hello
NAME                                      DESCRIPTION
STARS               OFFICIAL            AUTOMATED
hello-world                               Hello World! (an example of minimal Dockeriz…   1163
[OK]
```

7. Configure Single Node Kubernetes Cluster with ***minikube***

The following command will create *single node Kubernetes cluster that run as docker container*:

```
$ minikube config set driver docker
❗  These changes will take effect upon a minikube delete and then a minikube start

$ minikube start --driver=docker
😄  minikube v1.9.2 on Centos 7.7.1908
✨  Using the docker driver based on user configuration
👍  Starting control plane node m01 in cluster minikube
🚜  Pulling base image ...
💾  Downloading Kubernetes v1.18.0 preload ...
    > preloaded-images-k8s-v2-v1.18.0-docker-overlay2-amd64.tar.lz4: 542.91 MiB
🔥  Creating Kubernetes in docker container with (CPUs=2) (3 available), Memory=2200MB (5443MB
available) ...
🐳  Preparing Kubernetes v1.18.0 on Docker 19.03.2 ...
    ▪ kubeadm.pod-network-cidr=10.244.0.0/16
🌟  Enabling addons: default-storageclass, storage-provisioner
🏄  Done! kubectl is now configured to use "minikube"
```

8. Explore **minikube** Kubernetes Cluster Environment

The following commands show basic information of minikube Kubernetes cluster:

```
$ docker image ls
REPOSITORY                    TAG                 IMAGE ID            CREATED             SIZE
…
gcr.io/k8s-minikube/kicbase   v0.0.8              11589cdc9ef4        2 weeks ago         964MB

$ docker container ls
CONTAINER ID        IMAGE                                   COMMAND                  CREATED
STATUS              PORTS
NAMES
e5a1fd44b8d7        gcr.io/k8s-minikube/kicbase:v0.0.8   "/usr/local/bin/entr…"   12 minutes
ago       Up 12 minutes       127.0.0.1:32770->22/tcp, 127.0.0.1:32769->2376/tcp,
127.0.0.1:32768->8443/tcp   minikube
```

```
$ minikube status
m01
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

$ kubectl get po -A
NAMESPACE         NAME                                        READY   STATUS    RESTARTS   AGE
kube-system       coredns-66bff467f8-5gtqm                    1/1     Running   0          34m
kube-system       coredns-66bff467f8-nxzw2                    1/1     Running   0          34m
kube-system       etcd-minikube                               1/1     Running   0          34m
kube-system       kindnet-mg9tm                               1/1     Running   0          34m
kube-system       kube-apiserver-minikube                     1/1     Running   0          34m
kube-system       kube-controller-manager-minikube            1/1     Running   0          34m
kube-system       kube-proxy-lvmxs                            1/1     Running   0          34m
kube-system       kube-scheduler-minikube                     1/1     Running   0          34m
kube-system       storage-provisioner                         1/1     Running   0          34m
kubernetes-dashboard  dashboard-metrics-scraper-84bfdf55ff-8rdh4   1/1 Running   0          20m
kubernetes-dashboard  kubernetes-dashboard-bc446cc64-5hl87         1/1 Running   0          20m
```

9. Deploy MariaDB and WordPress on Kubernetes Cluster

Use the following command to create the persistent volume directory for MariaDB and WordPress:

```
$ pwd
/home/kube

$ mkdir wordpressdb-data

$ sudo chown polkitd:root wordpressdb-data

$ mkdir wordpress-data

$ sudo chown 33:tape wordpress-data
```

Create the necessary YAML files for deploying MariaDB:

```
$ mkdir wordpress-config

$ cd wordpress-config

$ vi wordpressdb-pv-pvc.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: wordpressdb-pv
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 20Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/home/kube/wordpressdb-data"
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
```

```
  name: wordpressdb-pvc
  labels:
    app: wordpress
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi

$ vi wordpressdb-svc.yaml
apiVersion: v1
kind: Service
metadata:
  name: wordpressdb-svc
  labels:
    app: wordpress
spec:
  ports:
  - port: 3306
    protocol: TCP
    targetPort: 3306
  selector:
    app: wordpress
    tier: wordpressdb
  type: ClusterIP
  clusterIP: None

$ vi wordpressdb-deployment.yaml
---
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: wordpressdb-deploy
  labels:
    app: wordpress
spec:
  selector:
    matchLabels:
      app: wordpress
      tier: wordpressdb
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: wordpress
        tier: wordpressdb
    spec:
      containers:
      - image: mariadb:10.5.2-bionic
        name: wordpressdb-container
        env:
          # Use secret in real usage
        - name: MYSQL_ROOT_PASSWORD
          value: redhat
        - name: MYSQL_USER
          value: wordpress
        - name: MYSQL_PASSWORD
          value: redhat
        - name: MYSQL_DATABASE
          value: wordpress
        ports:
        - containerPort: 3306
```

```
          name: wordpressdb-svc
        volumeMounts:
        - name: wordpressdb-persistent-storage
          mountPath: /var/lib/mysql
      volumes:
      - name: wordpressdb-persistent-storage
        persistentVolumeClaim:
          claimName: wordpressdb-pvc
```

Create the necessary YAML files for deploying WordPress:

```
$ vi wordpress-pv-pvc.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: wordpress-pv
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/home/kube/wordpress-data"
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: wordpress-pvc
  labels:
    app: wordpress
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi

$ vi wordpress-svc.yaml
apiVersion: v1
kind: Service
metadata:
  name: wordpress-svc
  labels:
    app: wordpress
spec:
  ports:
    - port: 80
  selector:
    app: wordpress
    tier: frontend
  type: LoadBalancer

$ vi wordpress-deployment.yaml
---
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: wordpress-deploy
  labels:
```

```
    app: wordpress
spec:
  selector:
    matchLabels:
      app: wordpress
      tier: frontend
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: wordpress
        tier: frontend
    spec:
      containers:
      - image: wordpress:5.4.0-php7.2-apache
        name: wordpress
        env:
        - name: WORDPRESS_DB_HOST
          value: wordpressdb-svc
        - name: WORDPRESS_DB_PASSWORD
          value: redhat
        - name: WORDPRESS_DB_USER
          value: wordpress
        - name: WORDPRESS_DB_NAME
          value: wordpress
        ports:
        - containerPort: 80
          name: wordpress-svc
        volumeMounts:
        - name: wordpress-persistent-storage
          mountPath: /var/www/html
      volumes:
      - name: wordpress-persistent-storage
        persistentVolumeClaim:
          claimName: wordpress-pvc
```

Create `kustomization.yaml` File:

```
$ vi kustomization.yaml
resources:
- wordpressdb-pv-pvc.yaml
- wordpressdb-svc.yaml
- wordpressdb-deployment.yaml
- wordpress-pv-pvc.yaml
- wordpress-svc.yaml
- wordpress-deployment.yaml
```

Start deploying MariaDB and WordPress Using the following command:

```
$ pwd
/home/kube/wordpress-config

$ kubectl apply -k ./
```