**Managing containerized system services with Podman**



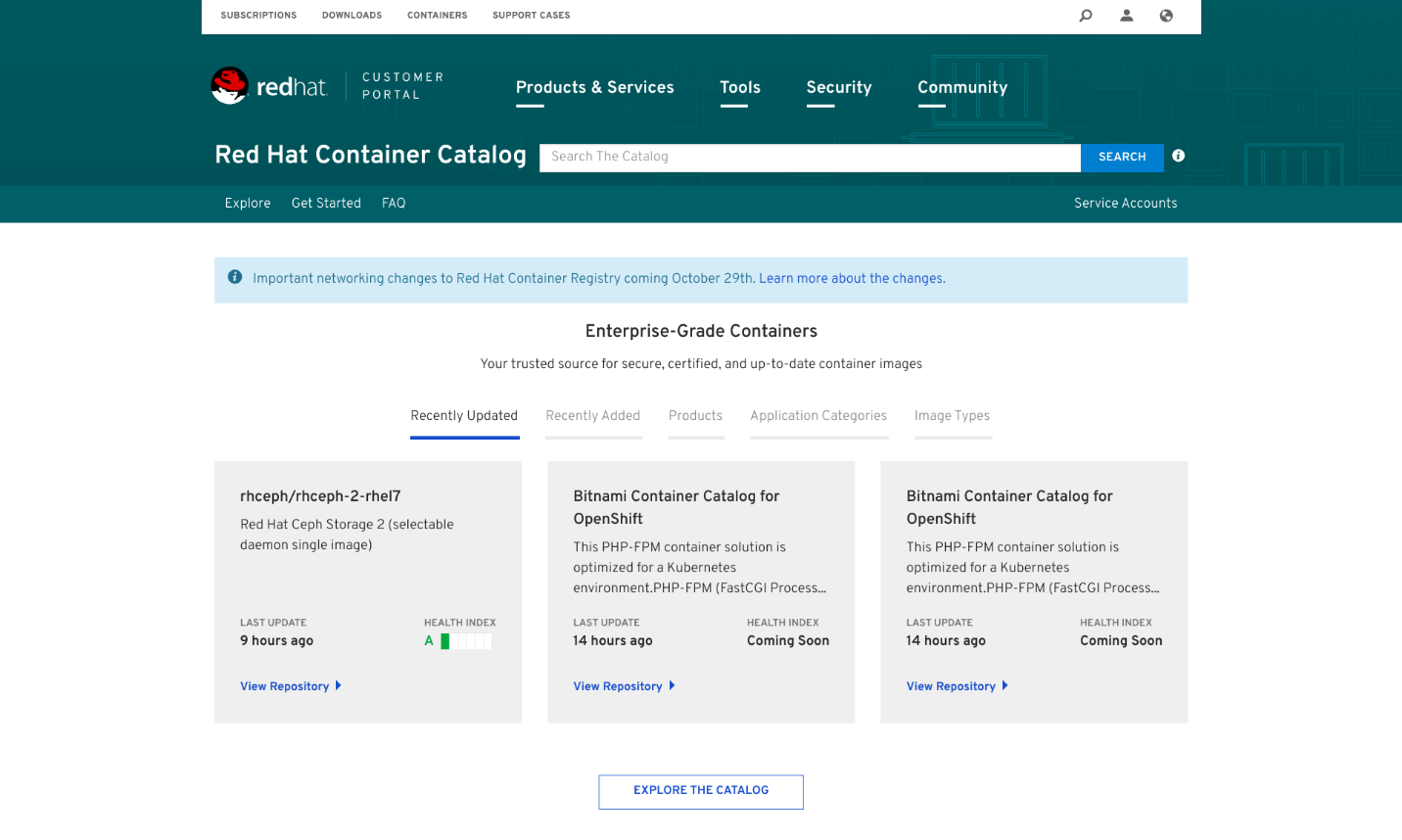
In this article, I discuss [containers](https://developers.redhat.com/blog/category/containers/), but look at them from another angle. We usually refer to containers as the best technology for developing new cloud-native applications and orchestrating them with something like [Kubernetes](https://developers.redhat.com/topics/kubernetes/). Looking back at the origins of containers, we’ve mostly forgotten that containers were born for simplifying application distribution on standalone systems.

In this article, we’ll talk about the use of containers as the perfect medium for installing applications and services on a [Red Hat Enterprise Linux](https://developers.redhat.com/products/rhel/) (RHEL) system. Using containers doesn’t have to be complicated, I’ll show how to run MariaDB, Apache HTTPD, and WordPress in containers, while managing those containers like any other service, through systemd and systemctl.

Additionally, we’ll explore Podman, which Red Hat has developed jointly with the Fedora community. If you don’t know what Podman is yet, see my previous article, [Intro to Podman (Red Hat Enterprise Linux 7.6)](https://developers.redhat.com/blog/2018/08/29/intro-to-podman/) and Tom Sweeney’s [Containers without daemons: Podman and Buildah available in RHEL 7.6 and RHEL 8 Beta](https://developers.redhat.com/blog/2018/11/20/buildah-podman-containers-without-daemons/).

**Red Hat Container Catalog**

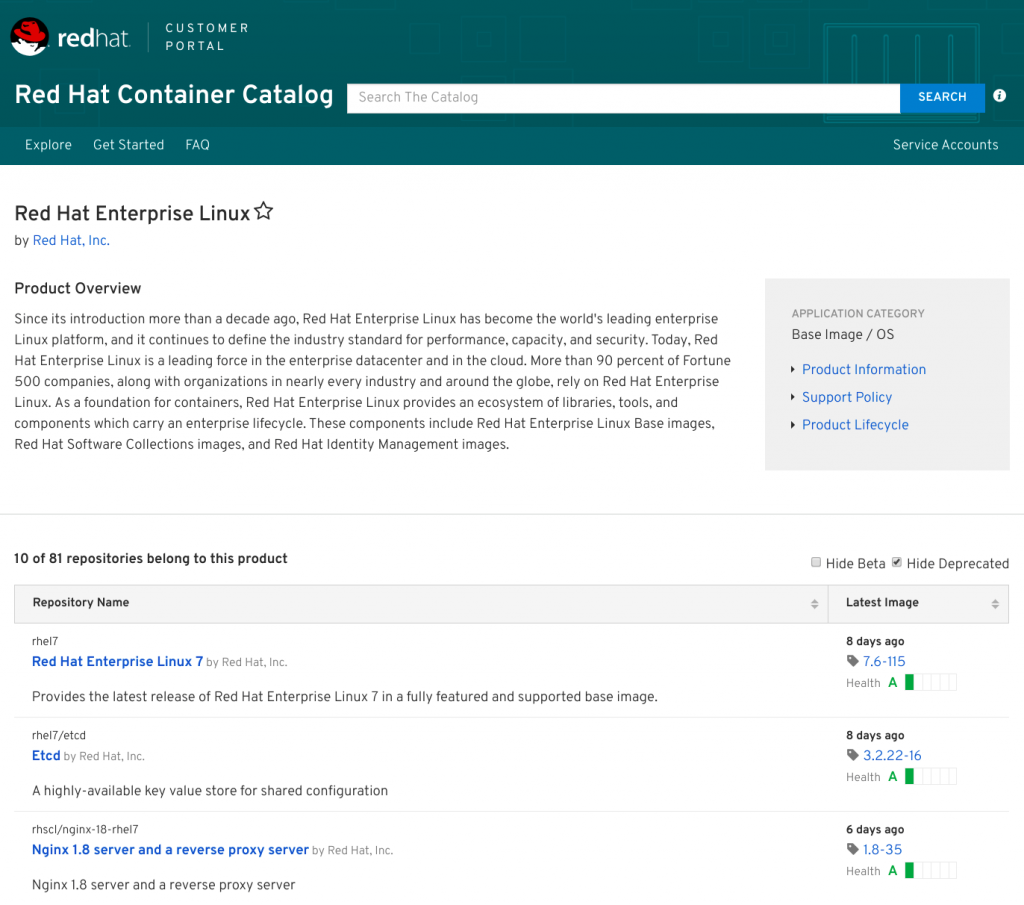
First of all, let’s explore the containers that are available for Red Hat Enterprise Linux through the [Red Hat Container Catalog (access.redhat.com/containers)](https://access.redhat.com/containers/):

[](https://developers.redhat.com/blog/wp-content/uploads/2018/11/FireShot-Capture-021-Container-Catalog-Red-Hat-Customer_-https___access.redhat.com_containers_-1-1024x613.png)

By clicking **Explore The Catalog**, we’ll have access to the full list of containers categories and products available in Red Hat Container Catalog.

[](https://developers.redhat.com/blog/wp-content/uploads/2018/11/FireShot-Capture-010-Container-Catalog-Red-Hat_-https___access.redhat.com_containers__explore.png)

Clicking **Red Hat Enterprise Linux** will bring us to the RHEL section, displaying all the available containers images for the system:

[](https://developers.redhat.com/blog/wp-content/uploads/2018/11/FireShot-Capture-009-Container-Catalog-Red-Hat-Customer-_-https___access.redhat.com_containe.png)

At the time of writing this article, in the RHEL category there were more than 70 containers images, ready to be installed and used on RHEL 7 systems.

So let’s choose some container images and try them on a Red Hat Enterprise Linux 7.6 system. For demo purposes, we’ll try to use Apache HTTPD + PHP  and the MariaDB database for a WordPress blog.

**Everything you need to grow your career.**

With your free Red Hat Developer program membership, unlock our library of cheat sheets and ebooks on next-generation application development.

[**SIGN UP**](http://developers.redhat.com/register?intcmp=701f2000001ODrzAAG)

[](http://developers.redhat.com/register?intcmp=701f2000001ODrzAAG)

**Install a containerized service**

We’ll start by installing our first containerized service for setting up a MariaDB database that we’ll need for hosting the WordPress blog’s data.

As a prerequisite for installing containerized system services, we need to install the utility named Podman on our Red Hat Enterprise Linux 7 system:

**[root@localhost ~]# subscription-manager repos --enable rhel-7-server-rpms --enable rhel-7-server-extras-rpms**

**[root@localhost ~]# yum install podman**

As explained in my previous article, Podman complements Buildah and Skopeo by offering an experience similar to the Docker command line: allowing users to run standalone (non-orchestrated) containers. And Podman doesn’t require a daemon to run containers and pods, so we can easily say goodbye to big fat daemons.

By installing Podman, you’ll see that Docker is no longer a required dependency!

As suggested by the Red Hat Container Catalog’s MariaDB page, we can run the following commands to get the things done (we’ll replace, of course, docker with podman):

**[root@localhost ~]# podman pull registry.access.redhat.com/rhscl/mariadb-102-rhel7**

**Trying to pull registry.access.redhat.com/rhscl/mariadb-102-rhel7...Getting image source signatures**

**Copying blob sha256:9a1bea865f798d0e4f2359bd39ec69110369e3a1131aba6eb3cbf48707fdf92d**

**72.21 MB / 72.21 MB [======================================================] 9s**

**Copying blob sha256:602125c154e3e132db63d8e6479c5c93a64cbfd3a5ced509de73891ff7102643**

**1.21 KB / 1.21 KB [========================================================] 0s**

**Copying blob sha256:587a812f9444e67d0ca2750117dbff4c97dd83a07e6c8c0eb33b3b0b7487773f**

**6.47 MB / 6.47 MB [========================================================] 0s**

**Copying blob sha256:5756ac03faa5b5fb0ba7cc917cdb2db739922710f885916d32b2964223ce8268**

**58.82 MB / 58.82 MB [======================================================] 7s**

**Copying config sha256:346b261383972de6563d4140fb11e81c767e74ac529f4d734b7b35149a83a081**

**6.77 KB / 6.77 KB [========================================================] 0s**

**Writing manifest to image destination**

**Storing signatures**

**346b261383972de6563d4140fb11e81c767e74ac529f4d734b7b35149a83a081**

**[root@localhost ~]# podman images**

**REPOSITORY TAG IMAGE ID CREATED SIZE**

**registry.access.redhat.com/rhscl/mariadb-102-rhel7 latest 346b26138397 2 weeks ago 449MB**

After that, we can look at the [Red Hat Container Catalog page](https://access.redhat.com/containers/?tab=tech-details&platform=systemimages#/registry.access.redhat.com/rhscl/mariadb-102-rhel7) for details on the needed variables for starting the MariaDB container image.

Inspecting the previous page, we can see that under *Labels*, there is a label named *usage* containing an example string for running this container image:

**usage docker run -d -e MYSQL\_USER=user -e MYSQL\_PASSWORD=pass -e MYSQL\_DATABASE=db -p 3306:3306 rhscl/mariadb-102-rhel7**

After that we need some other information about our container image: the “*user ID running inside the container*” and the “*persistent volume location to attach*“:

**[root@localhost ~]# podman inspect registry.access.redhat.com/rhscl/mariadb-102-rhel7 | grep User**

**"User": "27",**

**[root@localhost ~]# podman inspect registry.access.redhat.com/rhscl/mariadb-102-rhel7 | grep -A1 Volume**

**"Volumes": {**

**"*/var/lib/mysql/data*": {}**

**[root@localhost ~]# podman inspect registry.access.redhat.com/rhscl/mariadb-102-rhel7 | grep -A1 ExposedPorts**

**"ExposedPorts": {**

**"*3306*/tcp": {}**

At this point, we have to create the directories that will handle the container’s data; remember that containers are ephemeral by default. Then we set also the right permissions:

**[root@localhost ~]# mkdir -p /opt/var/lib/mysql/data**

**[root@localhost ~]# chown 27:27 /opt/var/lib/mysql/data**

Then we can set up our systemd unit file for handling the database. We’ll use a unit file similar to the one prepared in the previous article:

**[root@localhost ~]# cat /etc/systemd/system/mariadb-service.service**

**[Unit]**

**Description=Custom MariaDB Podman Container**

**After=network.target**

**[Service]**

**Type=simple**

**TimeoutStartSec=5m**

**ExecStartPre=-/usr/bin/podman rm "mariadb-service"**

**ExecStart=/usr/bin/podman run --name mariadb-service -v /opt/var/lib/mysql/data:/var/lib/mysql/data:Z -e MYSQL\_USER=wordpress -e MYSQL\_PASSWORD=mysecret -e MYSQL\_DATABASE=wordpress --net host registry.access.redhat.com/rhscl/mariadb-102-rhel7**

**ExecReload=-/usr/bin/podman stop "mariadb-service"**

**ExecReload=-/usr/bin/podman rm "mariadb-service"**

**ExecStop=-/usr/bin/podman stop "mariadb-service"**

**Restart=always**

**RestartSec=30**

**[Install]**

**WantedBy=multi-user.target**

Let’s take apart our ExecStart command and analyze how it’s built:

* /usr/bin/podman run --name mariadb-service says we want to run a container that will be named mariadb-service.
* *–*v /opt/var/lib/mysql/data:/var/lib/mysql/data:Z  says we want to map the just-created data directory to the one inside the container. The Z option informs Podman to map correctly the SELinux context for avoiding permissions issues.
* *–*e MYSQL\_USER=wordpress -e MYSQL\_PASSWORD=mysecret -e MYSQL\_DATABASE=wordpress identifies the additional environment variables to use with our MariaDB container. We’re defining the username, the password, and the database name to use.
* --net host maps the container’s network to the RHEL host.
* registry.access.redhat.com/rhscl/mariadb-102-rhel7 specifies the container image to use.

We can now reload the systemd catalog and start the service:

**[root@localhost ~]# systemctl daemon-reload**

**[root@localhost ~]# systemctl start mariadb-service**

**[root@localhost ~]# systemctl status mariadb-service**

**mariadb-service.service - Custom MariaDB Podman Container**

**Loaded: loaded (/etc/systemd/system/mariadb-service.service; static; vendor preset: disabled)**

**Active: active (running) since Thu 2018-11-08 10:47:07 EST; 22s ago**

**Process: 16436 ExecStartPre=/usr/bin/podman rm mariadb-service ​(code=exited, status=0/SUCCESS)**

**Main PID: 16452 (podman)**

**CGroup: /system.slice/mariadb-service.service**

**└─16452 /usr/bin/podman run --name mariadb-service -v /opt/var/lib/mysql/data:/var/lib/mysql/data:Z -e MYSQL\_USER=wordpress -e MYSQL\_PASSWORD=mysecret -e MYSQL\_DATABASE=wordpress --net host regist...**

**Nov 08 10:47:14 localhost.localdomain podman[16452]: 2018-11-08 15:47:14 140276291061504 [Note] InnoDB: Buffer pool(s) load completed at 181108 15:47:14**

**Nov 08 10:47:14 localhost.localdomain podman[16452]: 2018-11-08 15:47:14 140277156538560 [Note] Plugin 'FEEDBACK' is disabled.**

**Nov 08 10:47:14 localhost.localdomain podman[16452]: 2018-11-08 15:47:14 140277156538560 [Note] Server socket created on IP: '::'.**

**Nov 08 10:47:14 localhost.localdomain podman[16452]: 2018-11-08 15:47:14 140277156538560 [Warning] 'user' entry 'root@b75779533f08' ignored in --skip-name-resolve mode.**

**Nov 08 10:47:14 localhost.localdomain podman[16452]: 2018-11-08 15:47:14 140277156538560 [Warning] 'user' entry '@b75779533f08' ignored in --skip-name-resolve mode.**

**Nov 08 10:47:14 localhost.localdomain podman[16452]: 2018-11-08 15:47:14 140277156538560 [Warning] 'proxies\_priv' entry '@% root@b75779533f08' ignored in --skip-name-resolve mode.**

**Nov 08 10:47:14 localhost.localdomain podman[16452]: 2018-11-08 15:47:14 140277156538560 [Note] Reading of all Master\_info entries succeded**

**Nov 08 10:47:14 localhost.localdomain podman[16452]: 2018-11-08 15:47:14 140277156538560 [Note] Added new Master\_info '' to hash table**

**Nov 08 10:47:14 localhost.localdomain podman[16452]: 2018-11-08 15:47:14 140277156538560 [Note] /opt/rh/rh-mariadb102/root/usr/libexec/mysqld: ready for connections.**

**Nov 08 10:47:14 localhost.localdomain podman[16452]: Version: '10.2.8-MariaDB' socket: '/var/lib/mysql/mysql.sock' port: 3306 MariaDB Server**

Perfect! MariaDB is running, so we can now start working on the Apache HTTPD + PHP container for our WordPress service.

First of all, let’s pull the right container from Red Hat Container Catalog:

**[root@localhost ~]# podman pull registry.access.redhat.com/rhscl/php-71-rhel7**

**Trying to pull registry.access.redhat.com/rhscl/php-71-rhel7...Getting image source signatures**

**Skipping fetch of repeat blob sha256:9a1bea865f798d0e4f2359bd39ec69110369e3a1131aba6eb3cbf48707fdf92d**

**Skipping fetch of repeat blob sha256:602125c154e3e132db63d8e6479c5c93a64cbfd3a5ced509de73891ff7102643**

**Skipping fetch of repeat blob sha256:587a812f9444e67d0ca2750117dbff4c97dd83a07e6c8c0eb33b3b0b7487773f**

**Copying blob sha256:12829a4d5978f41e39c006c78f2ecfcd91011f55d7d8c9db223f9459db817e48**

**82.37 MB / 82.37 MB [=====================================================] 36s**

**Copying blob sha256:14726f0abe4534facebbfd6e3008e1405238e096b6f5ffd97b25f7574f472b0a**

**43.48 MB / 43.48 MB [======================================================] 5s**

**Copying config sha256:b3deb14c8f29008f6266a2754d04cea5892ccbe5ff77bdca07f285cd24e6e91b**

**9.11 KB / 9.11 KB [========================================================] 0s**

**Writing manifest to image destination**

**Storing signatures**

**b3deb14c8f29008f6266a2754d04cea5892ccbe5ff77bdca07f285cd24e6e91b**

We can now look through this container image to get some details:

**[root@localhost ~]# podman inspect registry.access.redhat.com/rhscl/php-71-rhel7 | grep User**

**"User": "1001",**

**"User": "1001"**

**[root@localhost ~]# podman inspect registry.access.redhat.com/rhscl/php-71-rhel7 | grep -A1 Volume**

**[root@localhost ~]# podman inspect registry.access.redhat.com/rhscl/php-71-rhel7 | grep -A1 ExposedPorts**

**"ExposedPorts": {**

**"8080/tcp": {},**

As you can see from the previous commands, we got no volume from the container details. Are you asking why? It’s because this container image, even if it’s part of RHSCL (formerly known as Red Hat Software Collections), has been prepared for working with the Source-to-Image (S2I) builder. For more info on the S2I builder, please take a look at its [GitHub project page](https://github.com/openshift/source-to-image).

Unfortunately, at this moment, the S2I utility is strictly dependent on Docker, but for demo purposes, we would like to avoid it..!

So moving back to our issue, what can we do for guessing the right folder to mount on our PHP container? We can easily guess the right location by looking at all the environment variables for the container image, where we will find APP\_DATA=/opt/app-root/src.

So let’s create this directory with the right permissions; we’ll also download the latest package for our WordPress service:

**[root@localhost ~]# mkdir -p /opt/app-root/src/**

**[root@localhost ~]# curl -o latest.tar.gz https://wordpress.org/latest.tar.gz**

**[root@localhost ~]# tar -vxf latest.tar.gz**

**[root@localhost ~]# mv wordpress/\* /opt/app-root/src/**

**[root@localhost ~]# chown 1001 -R /opt/app-root/src**

We’re now ready for creating our Apache http + PHP systemd unit file:

**[root@localhost ~]# cat /etc/systemd/system/httpdphp-service.service**

**[Unit]**

**Description=Custom httpd + php Podman Container**

**After=mariadb-service.service**

**[Service]**

**Type=simple**

**TimeoutStartSec=30s**

**ExecStartPre=-/usr/bin/podman rm "httpdphp-service"**

**ExecStart=/usr/bin/podman run --name httpdphp-service -p 8080:8080 -v /opt/app-root/src:/opt/app-root/src:Z registry.access.redhat.com/rhscl/php-71-rhel7 /bin/sh -c /usr/libexec/s2i/run**

**ExecReload=-/usr/bin/podman stop "httpdphp-service"**

**ExecReload=-/usr/bin/podman rm "httpdphp-service"**

**ExecStop=-/usr/bin/podman stop "httpdphp-service"**

**Restart=always**

**RestartSec=30**

**[Install]**

**WantedBy=multi-user.target**

We need then to reload the systemd unit files and start our latest service:

**[root@localhost ~]# systemctl daemon-reload**

**[root@localhost ~]# systemctl start httpdphp-service**

**[root@localhost ~]# systemctl status httpdphp-service**

**httpdphp-service.service - Custom httpd + php Podman Container**

**Loaded: loaded (/etc/systemd/system/httpdphp-service.service; static; vendor preset: disabled)**

**Active: active (running) since Thu 2018-11-08 12:14:19 EST; 4s ago**

**Process: 18897 ExecStartPre=/usr/bin/podman rm httpdphp-service (code=exited, status=125)**

**Main PID: 18913 (podman)**

**CGroup: /system.slice/httpdphp-service.service**

**└─18913 /usr/bin/podman run --name httpdphp-service -p 8080:8080 -v /opt/app-root/src:/opt/app-root/src:Z registry.access.redhat.com/rhscl/php-71-rhel7 /bin/sh -c /usr/libexec/s2i/run**

**Nov 08 12:14:20 localhost.localdomain podman[18913]: => sourcing 50-mpm-tuning.conf ...**

**Nov 08 12:14:20 localhost.localdomain podman[18913]: => sourcing 40-ssl-certs.sh ...**

**Nov 08 12:14:20 localhost.localdomain podman[18913]: AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 10.88.0.12. Set the 'ServerName' directive globall... this message**

**Nov 08 12:14:20 localhost.localdomain podman[18913]: [Thu Nov 08 17:14:20.925637 2018] [ssl:warn] [pid 1] AH01909: 10.88.0.12:8443:0 server certificate does NOT include an ID which matches the server name**

**Nov 08 12:14:20 localhost.localdomain podman[18913]: AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 10.88.0.12. Set the 'ServerName' directive globall... this message**

**Nov 08 12:14:21 localhost.localdomain podman[18913]: [Thu Nov 08 17:14:21.017164 2018] [ssl:warn] [pid 1] AH01909: 10.88.0.12:8443:0 server certificate does NOT include an ID which matches the server name**

**Nov 08 12:14:21 localhost.localdomain podman[18913]: [Thu Nov 08 17:14:21.017380 2018] [http2:warn] [pid 1] AH10034: The mpm module (prefork.c) is not supported by mod\_http2. The mpm determines how things are ...**

**Nov 08 12:14:21 localhost.localdomain podman[18913]: [Thu Nov 08 17:14:21.018506 2018] [lbmethod\_heartbeat:notice] [pid 1] AH02282: No slotmem from mod\_heartmonitor**

**Nov 08 12:14:21 localhost.localdomain podman[18913]: [Thu Nov 08 17:14:21.101823 2018] [mpm\_prefork:notice] [pid 1] AH00163: Apache/2.4.27 (Red Hat) OpenSSL/1.0.1e-fips configured -- resuming normal operations**

**Nov 08 12:14:21 localhost.localdomain podman[18913]: [Thu Nov 08 17:14:21.101849 2018] [core:notice] [pid 1] AH00094: Command line: 'httpd -D FOREGROUND'**

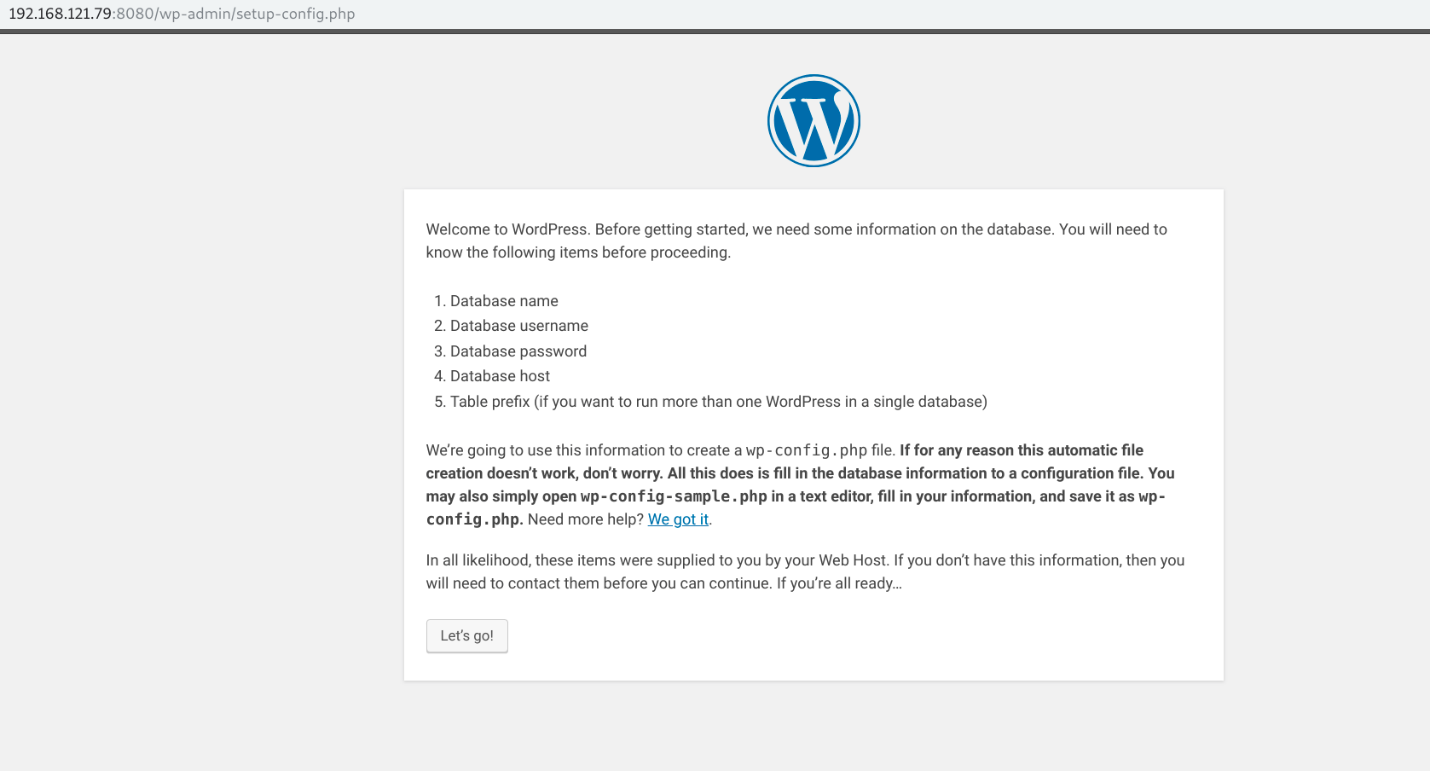
**Hint: Some lines were ellipsized, use -l to show in full.**

Let’s open the 8080 port on our system’s firewall for connecting to our brand new WordPress service:

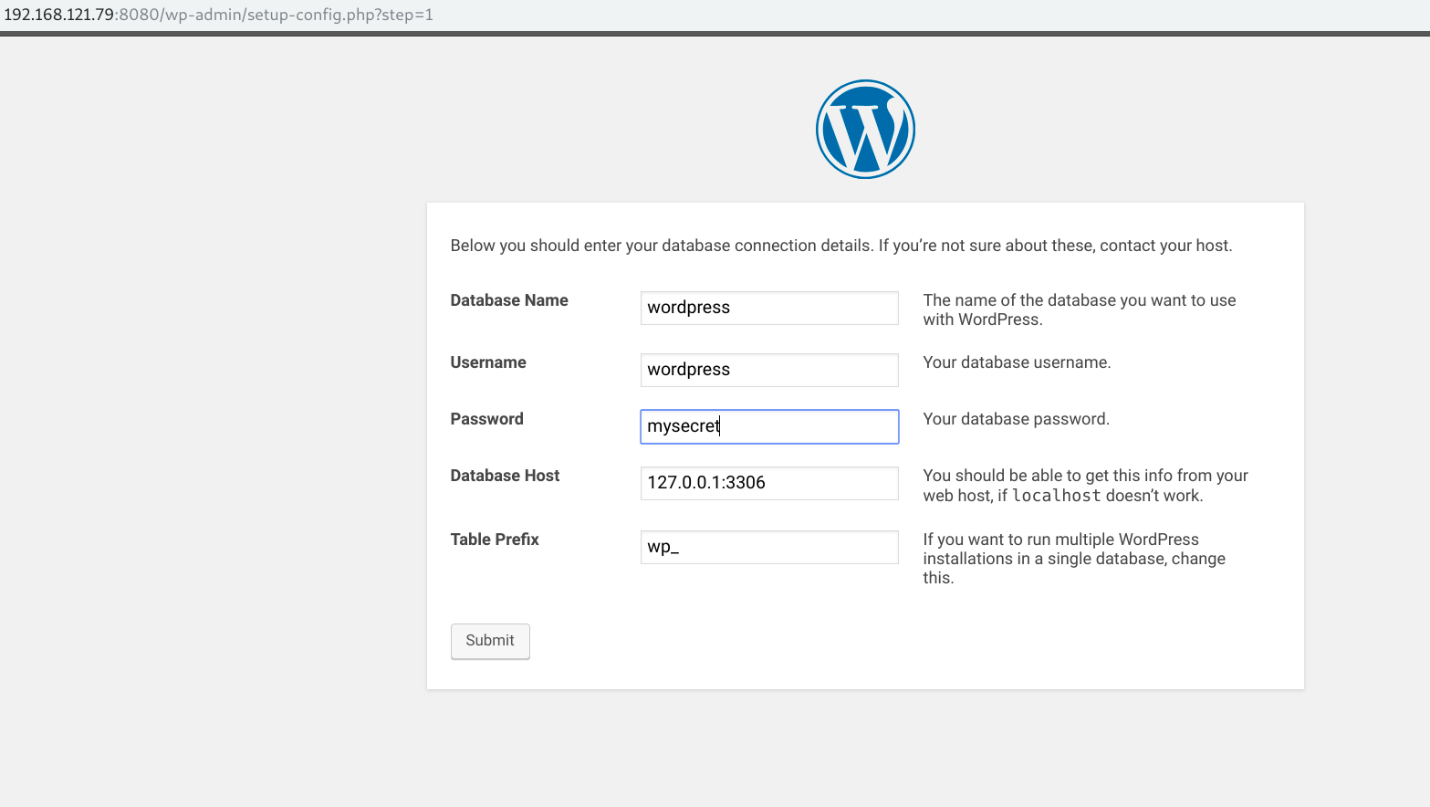
**[root@localhost ~]# firewall-cmd --permanent --add-port=8080/tcp**

**[root@localhost ~]# firewall-cmd --add-port=8080/tcp**

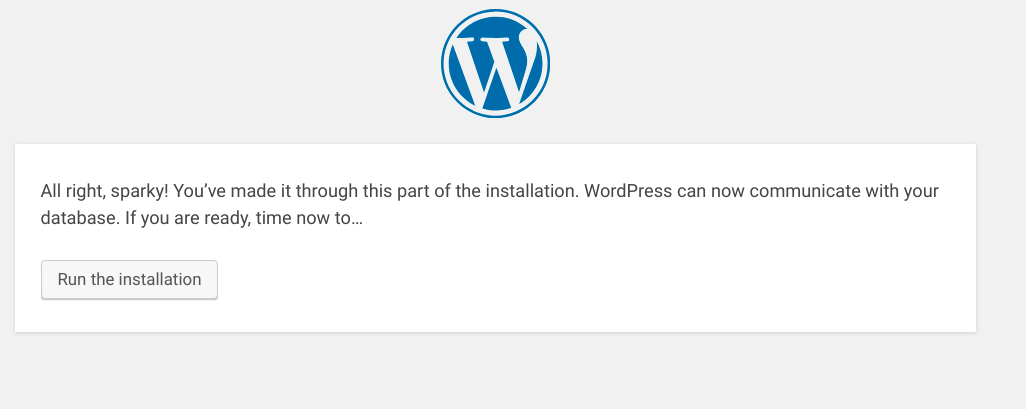
We can surf to our Apache web server:

[](https://developers.redhat.com/blog/wp-content/uploads/2018/11/Screenshot-from-2018-11-08-18-16-06.png)

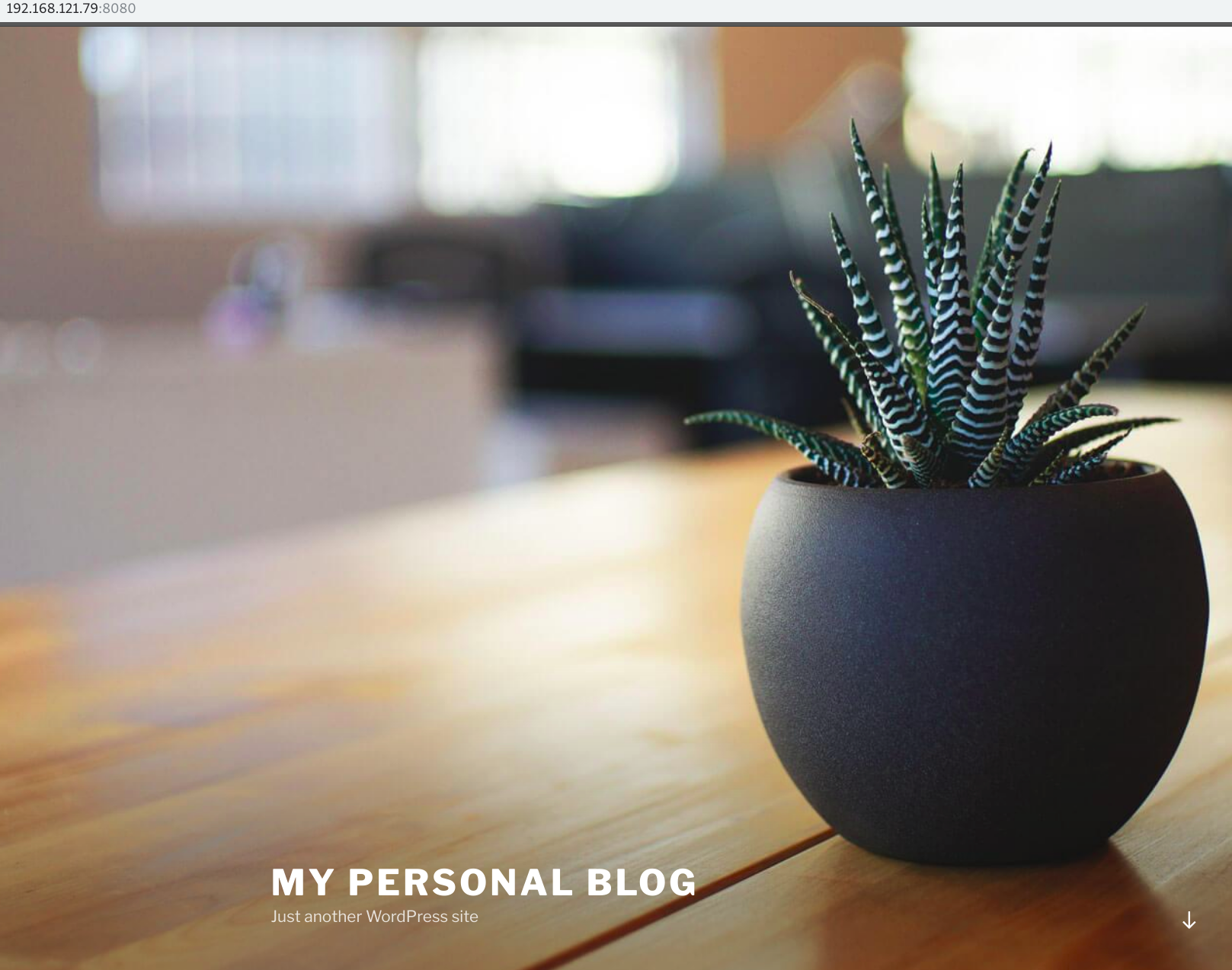
Start the installation process, and define all the needed details:

[](https://developers.redhat.com/blog/wp-content/uploads/2018/11/Screenshot-from-2018-11-08-19-04-46.png)

And finally, run the installation!

[](https://developers.redhat.com/blog/wp-content/uploads/2018/11/Screenshot-from-2018-11-08-19-07-07.png)

At the end, we should reach out our brand new blog, running on Apache httpd + PHP backed by a great MariaDB database!

[](https://developers.redhat.com/blog/wp-content/uploads/2018/11/Screenshot-from-2018-11-08-19-08-39-1024x806.png)

That’s all folks; may containers be with you!