RHEL 8: Rocky, Alma, CentOS

And maybe Fedora. Oracle can go screw.

- Setup
  - Apache & PHP
  - Install Composer
  - Amazon EFS
  - Setup SSL - It's Free

Setup

All guides here were written using Rocky Linux, but should work under any RHEL based distro.

Setup

Apache & PHP

Save yourself a lot of hassle and use Remi's RPM repository to ensure you have a well curated and up-to-date versions of Apache and PHP.

This is very easy. follow Remi's Configuration guide: https://rpms.remirepo.net/wizard/

> **Q:** Which EL version do I choose?
>
> **A:** Pick the EL version that matches your distro's version. For example if you are running Rocky Linux 8, choose EL 8.

Then just follow the commands provided by Remi to install the Repos and packages.

Install PHP

Once you've added Remi and followed all the commands, here's my recommended PHP install command to get all the most commonly needed PHP modules:

```
dnf install git libssh2 libssh2-devel ftp php php-json php-opcache php-gd php-intl php-mbstring php-mysqlnd php-pdo php-soap php-xml php-gmp php-sodium php-pear php-pecl-memcached php-pecl-mcrypt php-pecl-ssh2 php-pecl-zip php-pecl-mongodb memcached
```

This will Install:

- git - Everyone needs git
- libssh - For SSH connections
- ftp - Sometimes you need to ftp. You can remove this if you really don't want it
- memcached - Not all projects will need/use memcache, so this is optional
- php-mysqlnd, php-pdo, php-pecl-mongodb - Handles most of the DB connection types you will need
- Lots of other useful PHP modules for things like multi-byte support, xml, encryption, international, zip, and several others

You can always add or remove any PHP modules you do/don't want at any time.

Install Composer

Composer is an extremely commonly used PHP package manager. You will need/want it if you're working with PHP.

Run the following to install composer and make it available to the entire system. This requires root access. You can run these commands from any folder as they will clean up after themselves:

```
sudo su
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php composer-setup.php
mv composer.phar /usr/local/bin/composer
rm composer-setup.php
```

You can now access composer from anywhere!

Amazon EFS

If you are hosting on AWS EC2 and want to mount an Amazon EFS share to your server, follow these directions:

Install amazon-efs-utils

This is required for EFS support:

I recommend you run these commands in your home directory, but it can be done anywhere.

```
sudo dnf install git make rpm-build
git clone https://github.com/aws/efs-utils=
cd efs-utils
sudo make rpm
sudo dnf install ./build/amazon-efs-utils*rpm
```

Mount File Share

Here are a few placeholders you'll need to change to match your desired setup:

- **/mount_dir** - The directory you want the share to show up as
- **fs-XXX** - The filesystem ID of the EFS filesystem you're mounting
- **fsap-YYY** - The Access point ID of the filesystem. Found by selecting the filesystem and switching to the **Access Points** tab.

```
#First backup your fstab file in case you mess anything up:
sudo cp /etc/fstab /etc/fstab.bak

# Create the directory that you will be mounting the fileshare to
mkdir /mount_dir
```

```
# Open the fstab file
sudo vim /etc/fstab

# Add the following line to the bottom of the fstab, remembering to replace the fs-XXX,
mount_dir, and fsap-YYY values with your own
fs-XXX /mount_dir efs _netdev,tls,accesspoint=fsap-YYY 0 0

# Save and close Vim
:wq

# Verify it works
sudo mount -a
```

If the mount command is successful, you're done. The filesystem will automatically re-mount whenever the server reboots as well.

You can now **cd** into directory to read/write files.

Remember: when setting file permissions, Amazon EFS uses only the user and group IDs. This means if you are mounting the drive to multiple servers you need to take care to use the same user and group IDs on all servers. using the same names is not good enough.

You can specify the desired ID when setting up users and groups. For example:

```
groupadd -g 1003 apache
useradd -u 1502 -g 1502 myuser
```

In the above apache example, you'd need to do this before installing apache, otherwise the install process will assign it's own ID that you cannot control. It's not worth attempting to change IDs after they have been created. It's best to remove the user/group and start from scratch.

Setup SSL - It's Free

There's no reason to skip SSL anymore. These instructions will setup an auto-refreshing SSL certificate using the Let's Encrypt Certbot.

Install Certbot

Certbot requires snap (ugh). This will setup snap and then install Certbot.

```
sudo dnf install mod_ssl mod_fcgid snapd
sudo systemctl enable --now snapd.socket
sudo snap install core; sudo snap refresh core
sudo ln -s /var/lib/snapd/snap /snap
sudo snap install --classic certbot
sudo ln -s /snap/bin/certbot /usr/bin/certbot
```

If the **snap install** command fails, just run it again. It will sometimes fail on the first attempt.

Install SSL Cert

This assumes you have setup apache to serve your site over http first.

`sudo certbot --apache`

You can also add a specific list of domains to request certificates for. I prefer to do this:

`sudo certbot --apache -d www.mysite.com -d api.mysite.com`

Sometimes this step may fail the first time as well. Give it a a little time and then try again. It may fail because your DNS is still propagating.

Setup Automatic Certificate Renewal

Open up the root crontab entry:

`sudo vim /etc/crontab`

Enter the following entry which will check your SSL certs twice daily and also make sure not to flood Let's Encrypt at the same time as everyone else.

```
# Twice daily check of Lets Encrypt SSL Cert
0 0,12 * * * root python3 -c 'import random; import time; time.sleep(random.random() * 3600)'
&& certbot renew -q
```

If you don't have python3 installed, you can do so with:

`sudo dnf install python3`

General Notes

- Cloudflare can cause issues with the challenge so disable it (set it to passthrough on the DNS page) while issuing the cert if you run into issues.
- The certbot logs live in: /var/log/letsencrypt/letsencrypt.log

Areas for Improvement

This guide works but there are some areas for improvement in the future to lower the extra dependencies.

TODO:

- Use an alternate client instead of Certbot. For example: acme.sh

    - This is more lightweight and removes the dependency on snap (ugh).
- Use **$RANDOM** and **sleep** in crontab to remove python3 dependency