# Notes, Dependencies & Install

This install is pretty quick and easy.  But there was a couple items I wanted to fix after my first run. Instead of trying to remember what in the world I did, I figured I should jot it down.  Honestly I really wanted to try and get the "Supervised" version installed so I had easy access to the addons and updates. But oh no! The HA team apparently just want to work with Debian OS's and decided to not give two craps about us RedHat driven folks.  So they ripped out all their scripts that would have worked, and just left a Debian package load.

This should also be a note to anyone reading this.  If you need support, you will be looking for answers in a collection of OLD information.  The HA team has no issues with just changing their product to bleeding edge code and components, and breaking it for everyone.  They also do not have issues with not responding to anyone.  This also has the drawback of making all fixes and documentation out their (including their own) obsolete.  So I say...  know your craft, get creative, and figure it out yourself.  If your building your system around someone else's well documented instructions.  Be sure to read it well, and make sure any links that they reference still work.  Most do not.  And also note that it is not the person that wrote the doc's fault.  Just like this write up, it may be broken tomorrow!  All I can hope for is that the basics still apply and we can fish our way through to a good install.

Ok.  Let me get this ball rolling.

## Install Rocky Linux

This part is on you.  I personally always start with a Minimal install, and remove any of the additional package groups that are selected.  The additional packages include a bunch of tools and one of them is Cockpit.  Cockpit is very cool don't get me wrong. But it's overkill for most of my needs. In addition it needs secured, being that it is an interface into your system.  So that means it will also need to install all the web, and security features that it requires.  Really nice, BUT... it's bloat.  These packages can all be installed quite easily later using dnf / yum.

A additional note I would like to make is that I am not going to get into locking down the system.  I use IPTables (verses Firewalld).  I know IPTables, and it's easier for me.  Also I will not be getting into SELinux.  SELinux is like Debians AppArmor.  It's a good idea.  But I have other means of locking my systems away from the outside (router with nat, firewall, reverse proxy, ipset, fail2ban, iptables, ssh keys, and some good wholesome paranoia), and I don't want the complexity of SELinux biting me when I'm home just screwing around on my machines or trying to figure

something out.  So, your going to just get the stuff I did to install this software.  If your going to allow access to your system outside your network, then I implore you to LOCK IT DOWN!

## Some Dependencies and Pre-Work

Home Assistant is deprecating versions of certain software that comes with Rocky Linux 8 / CentOS 8, and there is not a way to get to the versions they require by using the main repositories including epel.  I am sure that someone perhaps has created some packages and hosts a repo, but I have found that installing these couple pieces of software from source was the way to go, and is well documented.  I like keeping my systems rpm based, but sometimes I have to give.

In order to make things install and work smoothly we are going to install updated copies of **SQLite** and **Python** along side the existing software.  But we will keep those loads in /usr/local so they are easily upgraded or removed later if needed.

Here are some packages that I loaded initially into my new Rocky Linux build.

```
# Installing some basics and deps

dnf install yum-utils
dnf install dnf-plugins-core
dnf install epel-release
dnf config-manager --set-enabled powertools
dnf groupinstall "Development Tools"

dnf update

# Reboot if you got kernel updates

dnf install openssl-devel bzip2-devel libffi-devel \
attr make sqlite-devel systemd-devel tzdata unzip \
wget xz-devel zlib-devel

# NOTE: Some of these should be installed already by default.  But I found that if I don't just try
# and install again, I am shooting myself in the foot.  We will just call it... not being complacent.
```

### **SQLite 3.43.1** *(updated 9/21/2023)*

***NOTE: I used all the steps below and just changed out the version numbers with newer release, and it all still worked.***

We are going to start with SQLite.  We need to have the upgraded SQLite in place so that we can compile it into the new Python install.  You can do Python first, but I found that this method worked well for me, I didn't have to jump through hoops to make it work.

Here is the link to the SQLite download page:  https://www.sqlite.org/download.html

Home Assistant is moving to version 3.31 or better.  So installing a higher version now will hopefully save a headache later.  We want to grab the source autoconf tar.gz file.  Verify the file name (and version) your pulling down and update the commands below.

```
# Download, Build, Install

cd /opt

SQYEAR=2023
VERSION=3430100
wget https://www.sqlite.org/$SQYEAR/sqlite-autoconf-$VERSION.tar.gz
tar -xzf sqlite-autoconf-$VERSION.tar.gz
chown -R root.root sqlite-autoconf-$VERSION
cd sqlite-autoconf-$VERSION

unset SQYEAR
unset VERSION

./configure
make
make install

cd /opt
```

Now lets check the version on the new install.  It should also be part of the environment now too, so we will check both ways.

`sqlite3 --version`

**3.43.1**

`/usr/local/bin/sqlite3 --version`

**3.43.1**

At the end of the install it will have spit out some jargon in regards to where the library files were installed.  Note this for later.

`/usr/local/lib`

I am also going to add this to the systems library locations.

```
echo "/usr/local/lib" > /etc/ld.so.conf.d/sqlite3.conf
ldconfig
```

Ok.  That's done.  Next!

## Python 3.11.5 *(updated 9/21/2023)*

***NOTE: I used all the steps below and just changed out the version numbers with newer release, and it all still worked.***

**Update 3/3/22:** Validate that your OpenSSL is version 1.1.1 or better.  Otherwise the new Python versions will not compile it in.

**Update 6/13/22**: Be sure to check the other applications you may want to install, and make sure they will work with Python 3.10.  I ran into a snag with Fail2Ban.  Was not the end of the world, but was not expecting the extra work.

Now we will install Python and link it to the SQLite library above.  Go get the gzipped source tarball from Python.  Again check the file name (and version) you are downloading and update the commands below.

https://www.python.org/downloads/source/

And here we go...

```
# Download, Build, Install

cd /opt

VERSION=3.11.5
wget https://www.python.org/ftp/python/$VERSION/Python-$VERSION.tgz
tar xf Python-$VERSION.tgz
chown -R root.root Python-$VERSION
cd Python-$VERSION
```

```
unset VERSION

# This may take a little while because of the "optimizations" being enabled.  But it's worth it (they say).
# Might be able to use "make -j xx" and add some more processing power.
LD_RUN_PATH=/usr/local/lib ./configure --with-ensurepip=install --enable-optimizations
LD_RUN_PATH=/usr/local/lib make
LD_RUN_PATH=/usr/local/lib make altinstall

# Set the new install to be default or just call it using 'Python3.11'

# Method 1)  - Remove any old or preset python and pip alternatives, I also set the priority to 3
alternatives --remove-all python
alternatives --remove-all pip
update-alternatives --install /usr/bin/python python /usr/local/bin/python3.11 3
update-alternatives --install /usr/bin/pip pip /usr/local/bin/pip3.11 3

# Method 2)
cd /usr/local/bin
ln -s python3.11 python3
ln -s python3.11 python
ln -s pip3.11 pip3
ln -s pip3.11 pip

# Upgrade pip (should already be the latest)
/usr/local/bin/python3.11 -m pip install --upgrade pip

cd ~
```

Now check the SQLite version that was compiled in.

`python -c "import sqlite3; print(sqlite3.sqlite_version)"`
**3.43.1**

`python3.11 -c "import sqlite3; print(sqlite3.sqlite_version)"`
**3.43.1**

# Installing Home Assistant Core

HA works inside its own Python virtual environment.  So in the sections below we will first create a new user account just for HA and then build out the environment.

```
# build the user account and using the -r indicates that this should be a system account
useradd -rm ha --home-dir /opt/ha

# add the user to the "dialout" group, so that it has access to USB devices
# i.e zWave devices
usermod -aG dialout ha

# create a configuration directory that does not live inside the virtual environment
mkdir /etc/ha

# create a media folder so camera footage and the like can be stored there
# Home Assitant will look for the media folder in the default configuration directory
# but you could put this folder elsewhere and configure it the configuration.yaml
# https://www.home-assistant.io/more-info/local-media/add-media
mkdir /etc/ha/media

# apply the new user permissions to the new directories
chown -R ha.ha /opt/ha /etc/ha /etc/ha/media
```

Now we will change to the new user account and build the virtual environment.  We will add a few dependencies to the virtual environment as well, so that it all goes smoothly.  I have cobbled together various dependencies and "nice to haves" from the main Home Assistant documentation and from various other how-to's out there.  So please note that what I am using is not necessarily the minimum basics or required.  But it works for me.

```
# switch to the new user
su - ha
# can also switch users using the home assistant documented method
# either way we need to be using the new users environment and not
# bring the root users environment with us
sudo -u ha -H -s

# change to ha's home direcotry if you used the second method
cd /opt/ha

# create the virtual environment
python3 -m venv .
```

```
# now activate the environment
# you should always activate this environment when doing work in here
. ./bin/activate

# now we need a couple dependencies built in this environment so that
# they are available to the home assistant install
pip3 install wheel
pip3 install -U git+git://github.com/lilohuang/PyTurboJPEG.git
pip3 install colorlog flask-sqlalchemy

# install the home assistant core stuff with zwave functions
pip3 install homeassistant home-assistant-frontend homeassistant-pyozw

# exit the activated environment and return to the root user
exit
```

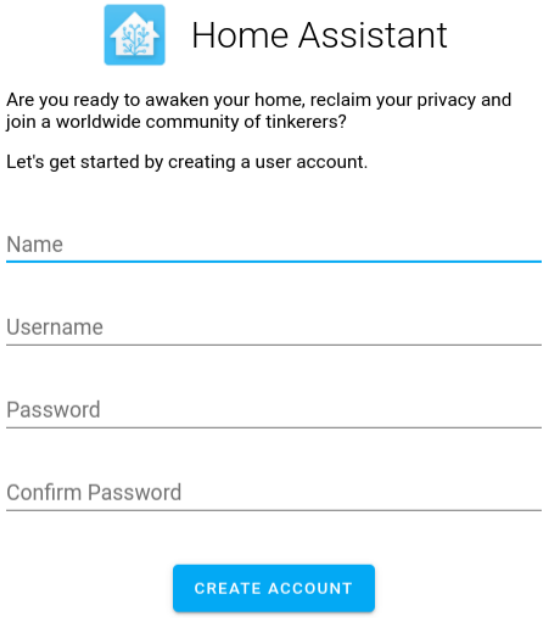
The last things we need to do is build a startup service file, and start up Home Assistant.

```
# create the startup service file
cat << EOF > /usr/lib/systemd/system/ha.service
[Unit]
Description=Home Assistant
After=network-online.target

[Service]
Type=simple
User=ha
ExecStart=/opt/ha/bin/hass -c "/etc/ha"
WorkingDirectory=/opt/ha
PrivateTmp=true
Restart=on-failure

[Install]
WantedBy=multi-user.target

EOF
```

```
# add the new file to the system (inform it of a change)
systemctl daemon-reload
```

```
# enable the service so it starts up on boot
systemctl enable ha

# and start it up
systemctl start ha
```

We should now be able to reach the Home Assistant web interface: http://<your ip or dns name>:8123
(If your running a firewall on the server you will need to open and allow port **8123**)




# Success!

Hope this helped a bit.  Take care!

---

Revision #25
Created 17 January 2022 16:42:54 by Phatlix
Updated 21 September 2023 17:08:46 by Phatlix