Guide to installing the KasaDaka software on a Raspberry Pi (2)

*This process includes some compiling, which can take a long time. The use of tmux/screen/byobu is recommended.*

*The services installed in this guide are NOT set up securely. It is not recommended to connect the RPi to the internet unprotected (use iptables!).*

*For this tutorial, the raspbian default user, pi, is used. Remember to change the commands used to a different user when deploying on a different configuration.*

## 

## 

## Setting up Raspbian

Download the raspbian image, and ‘burn’ it to the sd card.

Boot the pi, and configure it, so it is safely reachable by ssh. If used, configure wifi (<https://www.raspberrypi.org/documentation/configuration/wireless/wireless-cli.md>)

#### Set password

Change the password of the *pi* user, as well as *root*.

|  |
| --- |
| passwd  sudo passwd |

#### Expand the filesystem to extend to the entire SD card.

|  |
| --- |
| sudo raspi-config |

Expand the filesystem, set boot options to console, set timezone and make other optional changes (change hostname, etc). Reboot after changing the settings.

#### Update the system packages

|  |
| --- |
| sudo apt-get update  sudo apt-get upgrade  sudo apt-get dist-upgrade |

#### Installing dependencies and other necessary programs

|  |
| --- |
| sudo apt-get install \  build-essential autoconf apache2 curl chrpath pkg-config \  ncurses-dev libreadline-dev libgmp-dev libssl-dev \  unixodbc-dev zlib1g-dev libarchive-dev libossp-uuid-dev \  libxext-dev libice-dev libjpeg-dev libxinerama-dev libxft-dev \  libxpm-dev libxt-dev \  libdb-dev openjdk-7-jdk junit git tmux libxml2 htop libsqlite3-dev \  php5 libapache2-mod-php5 flite graphviz libxml2-dev python python-dev \  libapache2-mod-wsgi python-pip |

## 

## 

## Downloading the KasaDaka files

|  |
| --- |
| cd ~  git clone <https://github.com/abaart/KasaDaka.git> |

## 

## 

## Installing swi-prolog

|  |
| --- |
| git clone https://github.com/SWI-Prolog/swipl-devel.git  cd swipl-devel  git submodule update --init  sudo ./prepare  cp -p build.templ build  sudo nano build |

Change the PREFIX to */usr/local*

|  |
| --- |
| su  ./build  exit |

## 

## 

## Installing Cliopatria

|  |
| --- |
| cd ~  git clone https://github.com/ClioPatria/ClioPatria.git  cd ClioPatria  ./configure  git submodule update --init web/yasqe web/yasr  ./run.pl |

Open the web interface, and create the administrator user. (ex: user:kasadaka pass:kasadaka)

[http://IP ADDRESS:3020/](http://localhost:3020/)

Stop the program by ctrl+c and e.

#### Installing Cliopatria as a service

The service will operate as the www-data user. Make the necessary files accessible for this user:

|  |
| --- |
| sudo mkdir storage log  sudo chmod -R 777 storage log RDF-store users.db  sudo chgrp www-data storage RDF-store users.db |

Copy the init.d script:

|  |
| --- |
| sudo cp /home/pi/KasaDaka/etc/init.d/cliopatria /etc/init.d/cliopatria |

When using a different user: check whether the DAEMON\_PATH is correct.

Then add the script to rc.d:

|  |
| --- |
| sudo chmod 755 /etc/init.d/cliopatria  sudo update-rc.d cliopatria defaults |

Test it:

|  |
| --- |
| sudo /etc/init.d/cliopatria start |

And after some time, Cliopatria should be reachable on [http://IP ADDRESS:3020/](http://localhost:3020/)

#### Filling the triple store with data

Log in to Cliopatria using your set user and password. Upload the needed turtle files for your application. Repository > Load local file.

For the default example, you can use the turtle files of the Radiomarche project. These can be found at: <https://github.com/abaart/radiomarche>

Download the files to your local computer: <https://github.com/abaart/radiomarche/archive/master.zip>

For the example, we need to load the following files:

* rdf/speakle.ttl
* rdf/market\_data.ttl

After loading the files, give the anonymous user rights to write to the database. This is unsafe, but needed for now in order to allow the KasaDaka to update the triple store. Admin > Users > anonymous > check read and write > Modify.

## 

## 

## Setting up the audio storage and VXML generator (apache2)

First remove the default apache web files, then create a symlink to the KasaDaka files:

|  |
| --- |
| sudo rm -R /var/www/html/  sudo ln -s /home/pi/KasaDaka/html /var/www/html  sudo ln -s /home/pi/KasaDaka/FlaskKasadaka /var/www/FlaskKasadaka |

When browsing to [http://IP\_ADDRESS/](http://ip_address/), you should now get: ‘KasaDaka apache2 server’. The static audio files are now reachable.

#### Set up the Flask-based VXML generator

Set up the virtual environment and install the Flask microframework.

|  |
| --- |
| sudo pip install virtualenv  cd /var/www/FlaskKasadaka/FlaskKasadaka  sudo virtualenv FlaskKasadakaVenv  source FlaskKasadakaVenv/bin/activate  sudo pip install Flask  sudo python \_\_init\_\_.py |

Check the Flask server, the vxml generator should now be reachable at: [http://IP\_ADDRESS:5000/](http://ip_address:5000/)

Shut down the server with ctrl+c.

Leave the virtual environment.

|  |
| --- |
| deactivate |

#### Configuring apache2

|  |
| --- |
| sudo a2enmod wsgi  sudo rm -R /etc/apache2/sites-available  sudo ln -s /home/pi/KasaDaka/etc/apache2/sites-available /etc/apache2/sites-available  sudo service apache2 restart |

You should now be able to reach the VXML generator at: [http://IP\_ADDRESS/FlaskKasadaka/](http://ip_address/FlaskKasadaka/)

## 

## 

## Installing Asterisk and VXI (VXML interpreter)

|  |
| --- |
| cd ~  wget http://downloads.asterisk.org/pub/telephony/asterisk/asterisk-11-current.tar.gz  tar -xvf asterisk-11-current.tar.gz  cd asterisk-11.21.0  sudo ./configure --disable-xmldoc  cd contrib/scripts  sudo ./install\_prereq install  cd ../..  sudo make menuselect |

Make sure in the compile options menu, that the channel driver for SIP is selected.

|  |
| --- |
| sudo make  sudo make install  sudo make config  sudo make install-logrotate  sudo /etc/init.d/asterisk start  cd ~ |

### Installing I6NET VXI

|  |
| --- |
| wget http://downloads.i6net.com/vxi/raspbian/vxml\_V11.0\_2014-12-20\_dev\_armv6\_debian-7.0.tar.gz  tar -xvf vxml\_V11.0\_2014-12-20\_dev\_armv6\_debian-7.0.tar.gz  cd vxml\_V11.0\_2014-12-20\_dev\_armv6\_debian-7.0  sudo ./install.sh  sudo /etc/init.d/openvxi stop  sudo /etc/init.d/asterisk stop |

Asterisk and the VXML interpreter should be working now.

### Link the configuration files to the files from the git repository

Add the *asterisk* user to the *pi* group so it can access the git files. This way, it is easy to change the config in git and push it to (many) RPi’s.

|  |
| --- |
| sudo nano /etc/group |

And add asterisk to the pi group: *pi:x:1000:asterisk*

The next step is to remove the default config files. You can also choose to use mv to give them a different name, so the original files are backupped.

|  |
| --- |
| sudo rm -rf /etc/asterisk/  sudo rm -rf /etc/openvxi/ |

We will create symlinks to the config files pulled from git

|  |
| --- |
| sudo ln -s /home/pi/KasaDaka/etc/asterisk/ /etc/asterisk  sudo ln -s /home/pi/KasaDaka/etc/openvxi/ /etc/openvxi |

We are now ready to start the services again. As asterisk is dependant on openvxi, it is important to start both services in the correct order:

|  |
| --- |
| sudo /etc/init.d/openvxi start  sudo /etc/init.d/asterisk start |

Asterisk should now be reachable. This can be tested by using a software SIP phone (such as linphone), and calling the extension helloworld. ex: helloworld@IP\_ADDRESS. You should hear a hello world.

*NOTE: when using linphone for the first time, make sure to change the dtmf setting! Make sure ‘Send DTMFs as SIP info’ is checked. Otherwise, pressing a number will not be forwarded to asterisk.*

## 

## 

## Accessing the KasaDaka service

Now, everything is set-up. You can call the KasaDaka with your SIP phone, using:

kasadaka@IP\_ADDRESS.