Sage Server Setup

Martin Scholz

This Document is set with ConTEXt, using the MKIV branch
Many thanks to Hans Hagen for developing so hard on ConTEXt and
LuaTEX to providing such a great Document preparation system.

Kapitel

Contents

Kapitel 1

Foreword

As I have recently set up a Sage server I just wanted to share my experiences with the community. If you experience any problems, feel free to contact me under **info@scholz-net.org** or sage-support at **http://groups.google.com/group/sage-support**.

But first, what is Sage? Sage is a software application that covers many aspects of mathematics. The scope of the program includes algebra, combinatorics, numerical mathematics and calculus. The first version was released on February 24th, 2005 as open source under the the GNU GPL. Sage's aims are to create an free and open alternative to Magma, Maple, Mathematic and MATLAB.

Kapitel 2

Requirements

Sage offers you two versions for installing it,

1. Source Installation
2. Binary Installation

If you need to install from Source you first should make sure that you have the software fullfilled the requirements listed in **Section 2.2**.

You should use the Source installation if you want to use Sage on older hardware, but be warned, a build from source on a two Intel Pentium III tandem system could run from 12 up to 24 hours.

If you want to use the binary installation packages, you should have a modern Architecture as they need it. Unfortunately there is up to now no specification given. I only know it from personal experiences that you should have an architecture more modern than P III. If you wanna use that kind of architecture you need to take the time and compile the sources.

First determine what CPU architecture you are running. You can search for it with the following commands.

- On a Linux System you should run:

  # uname -a

  If the output contains a string like "x86_64", you are have a 64bit architecture and should load the 64-bit version, if the output contains a string like "x86", "i686", "i486" or "i386" you should load the 32-bit version. If you are using a 32-bit System on a 64-bit CPU you have to load the 32-bit version. A sample output of *uname -a* should look like on a 64-bit system:

  $ uname -a

  Linux localhost 2.6.32-15-server #1 SMP Mo Sep 13 22:17:05 UTC 2010 x86_64 GNU/Linux

  and on a 32-bit system you should get something like:

  $ uname -a

  Linux localhost 2.6.32-15-686 #1 SMP Mo Sep 13 22:30:54 UTC 2010 i686 GNU/Linux

- On Solaris systems execute the following command:

  # isainfo -v

  A sample output will look like:

  $ isainfo -v

64-bit sparcv9 applications

asi_blk_init vis2 vis

32-bit sparc applications

asi_blk_init vis2 vis v8plus div32 mul32

- We don't provide any builds for Windows. There are two solutions for you, you might find useful:

  1. You should use our VMware image and start it under your Windows installation in a virtual machine, but be warned, it would produce a heavy CPU load if you use it that way with multi calculations. Your Windows Host might get slow in responding.

  2. You should feel free to make a separate partition on your harddisk and install Linux on it.

- On Mac OS X you shouldn't have problems to use Sage. Sage is often tested on Mac OS X 10.5.x but should also run on Mac OS X 10.4.x and you should be able to run it without problems. The only thing you should know is your CPU type (weather it is an Intel or a PowerPC) and the architecture (if it is 32-bit or 64-bit). If you want to compile it yourself on a Mac OS X please make sure you have all the needed build-tools (see **Section 2.2**) including *Xcode*.

§ 2.1 - Installing the binary version

Sage provides at the moment prebuilded packages for Ubuntu, Mandriva, OpenSUSE, fedora and Debian. The installation of these packages is pretty simple, first load it from **http://sagemath.org/download.html** and extract the received *.tar.lzma*-file.

$ tar --lzma -xfv sage-*.tar.lzma

We recommend to load these lzma compressed archives, as they save you about 150MB of space for loading.

That was all you will receive a folder called *sage-x.y.z*. As you want to run a server we recommend to move that file to */opt/sage* by executing

mv ./sage-x.y.z /opt/sage

After adding */opt/sage* to your path and installing LaTeX with the packagemanager of your distribution, we have finished the binary installation. Unfortunately there are now distribution specific installations right now.

§ 2.2 - Software for installing form Source

If you are in the situation that the above installation failed in so far that you can't produce any graphics or sage isn't startable you need to do a source install. For a source installation you need to have the following tools installed

- gcc
- g++
- gfortran
- make
- m4
- perl
- ranlib
- tar
- readline and its development headers
- ssh-keygen – needed to run the notebook in secure mode.
- latex – highly recommended, though not strictly required

As the installation differs from distribution to distribution, you should consult your distribution specific documentation project for help.

§ 2.3 - Hardware

As Sage uses your CPU, RAM and Harddrive-space for calculating and saving you should at least have these requirements for running sage.

The used CPU is nearly not restricted, you should only keep in mind that the more users use your server setup simultanously you would have a lack of speed. So it would be a good starting point to have at least an Intel Pentium P III but a higher one would be much better.

The minimum amount of harddrive space you need to 2 GB, but that is the minimum requirement if you use a binary installation, for a source based installation I would recommend to have 4 GB or more. As we would like to do a Server setup, I would be nice to have at least 4 GB for the binary installation and 6 GB for the source based.

The RAM depends on the expected amount of simultaneously users on the Sage Server. The following **table 2.1** will give you an overview of the amount of simultaneous users and RAM.

For the source based installation go to **http://sagemath.org/download.html** and select the Source download. Now get a nice cup of tea and have a bit of time, as you will load about 290 MB to your PC. When it is finished I recommend again to get the installation based in */opt/sage*.

```
cd /opt
tar xfvz /path/.../to/sage-x.y.z.tar.gz
```

RAM (GB)	Simultaneous Users
1	5
2	20
3	40
4	60
8	140
12	200

Table 2.1 Ram to Simultaneous Users

```
mv ./sage-x.y.z sage
```

```
cd sage
```

Now we like to give you the advice to take a look into the *README.txt* file, but it is not necesary for finishing the rest of our installation. If you read the README or decided to skip it execute

```
make
```

Now you get up and get yourself the ingredients for a cake, prepare it, eat a piece of it with a cup of tea, as the compilation would take a lot of time. As said by the readme it could take about an hour up to two weeks depending on the device you use to compile it on.

now that we like to use Sage in a multiuser-way we should start sage and make symlinks for executing it.

```
./sage
```

```
----------------------------------------------------------------------
| Sage Version 4.5.2, Release Date: 2010-08-05                       |
| Type notebook() for the GUI, and license() for information.        |
----------------------------------------------------------------------


Loading Sage library. Current Mercurial branch is: main
-rwxr-xr-x  1 root root   108 Aug 21 19:15 sage-push
-rw-r--r--  1 root root 34067 Aug 21 21:41 setup.py
-rwxr-xr-x  1 root root   883 Aug 21 19:13 spkg-delauto
-rwxr-xr-x  1 root root  1618 Aug 21 19:14 spkg-dist
-rwxr-xr-x  1 root root  3720 Aug 21 19:15 spkg-install
sage:
```

now type in the sage input command the following

```
install_scripts("/usr/local/bin/")
```

here you should change */usr/local/bin* to the path where you want to have Sage symlinked to, but that path should be all right.

Now we have all prerequierments fullfilled and could now move over to setup our server.

Kapitel 3

The Server Setup

Now we need to add some users to our system, here we should again have in mind how many simultaneous users we expect. Let us say for our howto here we expect 20 simultaneous users plus the educator, so create the users and a passwordless ssh key for granting the sageadm to login as one of our users. To do so execute the following commands:

```
alias sagesrv="echo 'Sourcing the profile' && sleep 1 && \

               source /etc/zsh/zprofile && echo 'Starting Sage Webserver' && \

               sleep 1 && nohup sage start_notebook.sage & "
```

Now we created 22 users (sageadm, nb0 - nb20), a passwordless ssh keyfile for sageadm and granted sageadm the login without a password. That was the main work for creating the system specific needs of a sageserver. For the next step we change to our sageadm user again by executing *su sageadm* and create a file called *start_notebook.sage* with the following content:

```
# A script for starting the Sage notebook with the options you like.

# Change the options to your needs, save this script and do "sage

# start_notebook.sage" to start the notebook with these options.

#

# by Dan Drake; see http://wiki.sagemath.org/DanDrake/JustEnoughSage-
Server


# we'll stuff everything into a dictionary and pass that on to note-
book()

nbopts = {}


# listen on all addresses

nbopts['interface'] = ''


# use https, not http

nbopts['secure'] = False


# don't open a viewer, I'll do that myself thankyouverymuch

nbopts['open_viewer'] = False
```

```
# use this directory for nb files; must be writable by the nb? users
nbopts['directory'] = '/home/sageadm/nbfiles'

# at most 500MB memory, 100MB files, 100 processes for nb? users
nbopts['ulimit'] = '-v 500000 -f 100000 -u 100'

# time out idle sessions after two hours
nbopts['timeout'] = 7200

# yes, can create new accounts
nbopts['accounts'] = True

# use these minions to do our bidding
nbopts['server_pool'] = ['nb%s@localhost' % n for n in [0..20]]

# Go!
notebook(**nbopts)
```

if you created more or less users edit the server_pool variable, the rest of the settings shouldn't be needed to change. Next is to create a nice alias to start our server nicely, unfortunately there isn't any startupscript for it now, hopefully there will be one someday.

Open your *.bashrc* or *.zshrc* for the sageadm user and add the follwoing content to it:

```
alias sagesrv="echo 'Sourcing the profile' && sleep 1 && \
               source /etc/zsh/zprofile && echo 'Starting Sage Web-
server' && \
               sleep 1 && nohup sage start_notebook.sage & "
```

here you should replace the source command's path with the path to your profile-file and the Command should work like that, if you experience any problems, delete the \and write the command in one line.

after sourching now our profile-file we are able to start the sageserver as user sageadm, you need to start the server manually after every restart/reboot of your Sage Server. after executing

```
sagesrv
exit
```

you are able to access your Sage Server already under *http://hostname:8000*, but that looks not really nice, and for an access from the outer world it is not the wished way. So we first go to **http://no-ip.com** and create a free account for our Sage Server. The no-ip.com-client is included in the most distributions, install it with your package manager and configure it, or feel free to use any other DynDNS service.

Now we need to have an Apache2 server installed, install it with you prefered package manager of your distribution and look for the *proxy* and *proxy_http* modules support enabled for Apache2. For funtoo/gentoo just set these useflags and for Debian based systems execute:

```
a2enmod proxy
```

```
a2enmod proxy_http
```

Now create a file called *sageserver-vhost.conf* with the following content:

```
NameVirtualHost Yourhost.no-ip.TLD:80


<VirtualHost Yourhost.no-ip.TLD:80>


    ServerName Yourhost.no-ip.org:80
    PoxyRequests Off


    <Proxy *>
        Order deny,allow
        Allow from all
    </Proxy>


    ProxyPass / http://localhost:8000/
    ProxyPassReverse / http://localhost:8000/


    <Location />
        DefaultType text/html
    </Location>


</VirtualHost>
```

and if you need to enable it do it. Under fundoo/gentoo this file is saved as */etc/apache2/vhosts.d/10_Yourhost.no-ip.TLD.conf* and under Debian based systems in */etc/apache2/sites-available* and you need to make it available with

```
a2ensite sagenotebook
```

```
/etc/init.d/apache2 restart
```

For the final usage we just install some fonts for our Sage Server by executing

```
sage -i jsmath_image_fonts-1.4.p3
```

and we are done.

Ath the end wou could only add a little improvement, if we did all like the above, we see that it is hard to start the sage-server here is a little improvement if you are able to run start-stop-runscripts...

Just save the following runscript under */etc/init.d/sage-notebook-server*:

```
#!/sbin/runscript

extra_commands="setup"
description_reset=""

depend() {
        need net
}

start() {
        if [ ! -e ~sageadm/.sage/sage_notebook.sagenb/users.pickle ];
then
                eerror "Sage's Notebook was not properly set up. Run
setup first."
                return 1
        fi

        ebegin "Starting Sage's Notebook server"
        start-stop-daemon \
                --start \
                --background \
                --stdout ~sageadm/.sagelog/sagenb.log \
                --stderr ~sageadm/.sagelog/sagenb.err \
                --pidfile ~sageadm/.sagelog/sagenb.pid \
                --user sageadm \
                --exec /usr/local/bin/sage \
                -- -notebook \
```

```
                interface=${interface-''} \
                secure=${secure-False} \
                open_viewer=False \
                directory='/home/sageadm/nbfiles' \
                ulimit='-v 500000 -f 100000 -u 100' \
                timeout=${timout-0} \
                accounts=${accounts-True} \
                server_pool=['nb%s@localhost' % n for n in [0..20]]
        eend $? "Failed to start Sage's Notebook server"
}

stop() {
        ebegin "Stopping Sage's Notebook server"
        start-stop-daemon --stop --pidfile ~sageadm/.sagelog/sagenb.pid
--user sageadm
        eend $? "Failed to stop Sage's Notebook server"
}
```

make it executable by running

```
chmod 744 /etc/init.d/sage-notebook-server
```

then run the following:

```
/etc/init.d/sage-notebook-server setup
```

and execute in SAGE the following

```
notebook()
```

it will now creat some directorys for you and ask for an admin password, remember it good and quit thesetup environment. Now just create one directory *.sagelog* in the homedir of *sageadm* and that is it... you are ready to use the notebook server as a runlevelscript. Under funtoo you can do so with the following procedure that also adds the notebook-server to the default-runlevel under funtoo/gentoo:

```
su sageadm
mkdir ~/.sagelog
exit
rc-update add sage-notebook-server default
/etc/init.d/sage-notebook-server start
```

and it should all be set up and working now.

Kapitel A

Why an own Sage Server

Most of you would now ask themself why they should set up their own Sage Server. The ones who used **http://sagenb.org** know that the server is sometimes really slow, and if you are working whole group on the server you would feel it even more.

So you wouldn't feel these slowdowns if you provide your students an own Sage Server. The next reason is that the **http://sagenb.org** only has a way to make worksheets publicly available and some students might not want it, so you have the opportunity to make the server only available from the intranet by firewalling it. And if you provide them a home access there are several other ways of doing it, keep for that a talk with your IT-Department.