README

March 22, 2016

1 Jupyter Notebook Server on Raspberry PI 2 and 3

1.1 Intro

Sliderules are a thing of the past, decent calculators are hard to get by these days and spreadsheets are somewhat cumbersome, at times outright dangerous or just not the right tool for many tasks. Project Jupyter not only revolutionizes data-heavy research in all domains - it also boosts personal productivity for problems on a much smaller scale.

This repository documents my efforts to set up and configure a Jupyter Notebook Server on a Raspberry Pi 2 or 3 complete with Python 3.5.1, fully functioning nbconvert and a basic scientific stack with version 4.0 or later of all components making up the brilliant Jupyter interactive computing environment.

1.2 Requirements

- a Raspberry Pi 2 or 3 complete with 5V micro-usb power-supply
- a blank 16 GB micro SD card
- an ethernet cable to connect the Pi to your network *)
- a static IP address for the Raspberry Pi
- an internet connection
- a computer to carry out the installation connected to the same network as the Pi
- a fair amount of time *)

*) When I tested the setup on a Raspberry Pi 3, I used built-in WIFI to connect to my network. I encountered WIFI signal drops until I disabled WIFI power management by adding `iwconfig wlan0 power off` to `/etc/rc.local` before `exit(0)`.

1.3 Preparing the Raspbian Jessie Lite Image

Download the official Raspbian Jessie Lite image and transfer it to your SD card. Boot the Pi with the fresh image, log in (root password is raspbian and default user is pi) to set up timezone and locales and expand the filesystem using the raspi-config utility:

```
sudo raspi-config
```

Ensure that your installation is up to date and then use apt-get to install pandoc and git:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install -y pandoc
sudo apt-get install -y git
```

Set up new user jns (jns stands for **j**upyter **n**otebook **s**erver) and add the new user to the groups sudo and ssh as we are going to use this user to perform the installation and later to start the server.

```
adduser jns
usermod -aG sudo,ssh jns
```

1.4 Clone the GitHub Repository

Reboot and log in as user jns via ssh. From the terminal run:

```
cd /home/jns
git clone https://github.com/kleinee/jns.git
cd jns
chmod +x *.sh
```

This clones the github repository onto your Pi and makes the shell scripts executable. To complete preparation run:

```
sudo ./configure_disk_image.sh
```

This will set up a swap partition and improve memory management performance as suggested on https://github.com/debian-pi/raspbian-ua-netinst.git:

```
#!/bin/bash
# configure_disk_image.sh
# purpose: configure disk image
# last modified: 2016/01/02

if ! [ $(id -u) = 0 ]; then
   echo "to be run with sudo"
   exit 1
fi

# set up swap partition
target=/etc/fstab
if ! grep -Fxq /swap $target; then
    dd if=/dev/zero of=/swap bs=1M count=512
    mkswap /swap
    echo '/swap none swap sw 0 0' >> $target
else
    echo swap partition already configured
fi

# speed things up

#-------------------------------------------------------
apt-get install -y raspi-copies-and-fills
#-------------------------------------------------------
```

1.5 Server Installation

```
sudo ./install_jns.sh
```

This will create a directory notebooks in the home directory of user jns, clone this repository to get the installtion scripts, make the scripts executable and then run install_jns.sh which does the following:

- install Python
- install jupyter
- (pre)-configure the notebook server
- install TeX
- install scientific stack

The script is nothing spectacular - just convenience to save us some typing. The next section briefly describes the individual steps.

```
#!/bin/bash
# script name:      install_jns.sh
# last modified:    2015/09/30
# sudo:             yes

if ! [ $(id -u) = 0 ]; then
   echo "to be run with sudo"
   exit 1
fi

# run scripts
./install_python.sh
./install_jupyter.sh
sudo -u jns ./configure_jupyter.sh
./install_tex.sh
./install_stack.sh
```

If everything goes to plan you end up with a fully functional Jupyter Notebook server!!! To start the server just run:

```
jupyter notebook
```

You should now be able to access the system from any browser on your network via the IP address of the Raspberry Pi on port 9090. The **notebook server password*** set during installation is **jns**. This can be changed if requirerd.

1.6 Step by Step Installation + Configuration

If you prefer a setp by step installation, execute the respective shell scripts in the order given below:

- To install Python 3.5.1 run `install_python.sh`
- To install TeX run "'install_tex.sh"'
- To install Jupyter run `install_jupyter.sh`
- To configure Jupyter run `configure_jupyter.sh`
- To install scientific stack run `install_stack.sh`

1.6.1 Python 3.5.1 Installation

Instructions for building Python from source can be found here. I adjusted them to suit installtion of Python 3.5.1 and turned the instructions into a script:

```
#!/bin/bash
# script name:      install_python.sh
# last modified:    2015/09/30
# sudo:             yes
#
# see: http://sowingseasons.com/blog/building-python-3-4-on-raspberry-pi-2.html

if ! [ $(id -u) = 0 ]; then
   echo "to be run with sudo"
   exit 1
fi
```

```
#Python 3 version to install
version="3.5.1"

#-------------------------------------------------------
apt-get install -y build-essential libncursesw5-dev
apt-get install -y libgdbm-dev libc6-dev
apt-get install -y zlib1g-dev libsqlite3-dev tk-dev
apt-get install -y libssl-dev openssl
#-------------------------------------------------------

wget "https://www.python.org/ftp/python/$version/Python-$version.tgz"
tar zxvf "Python-$version.tgz"
cd "Python-$version"
./configure
make
make install
pip3 install pip --upgrade

# soft link to make pip3 default

ln -s /usr/local/bin/pip3 /usr/local/bin/pip

# clean up

cd ..

rm -rf "./Python-$version"
rm "./Python-$version.tgz"
```

1.6.2 TeX Installation

We need TeX for notebook conversion to PDF format with nbconvert / pandoc.

```
#!/bin/bash
# script name:     install_tex.sh
# last modified:   2015/09/22
# sudo:            yes

if ! [ $(id -u) = 0 ]; then
   echo "to be run with sudo"
   exit 1
fi

#-------------------------------------------------------
apt-get install -y texlive
apt-get install -y texlive-latex-extra
apt-get install -y dvipng
#-------------------------------------------------------
```

1.6.3 Jupyter Installation

The developers made this step amazingly simple. The only minor issue that I came across was that IPython complained about missing readline upon first start. We adress this here by installing readline. We also install ipyparallel as it is not installed by default.

```
#!/bin/bash
# script name:      install_jupyter.sh
# last modified:    2015/09/22
# sudo:             yes

if ! [ $(id -u) = 0 ]; then
   echo "to be run with sudo"
   exit 1
fi

pip install jupyter

#-------------------------------------------------------
apt-get -y install libncurses5-dev
#-------------------------------------------------------

pip install readline
pip install ipyparallel
```

1.6.4 Jupyter Configuration

We generate a jupyter notebook configuration directory and in it a file called jupyter_notebook_config.py that holds the configuration settings for our notebook server. We also create a folder notebooks in the home directory of user jns as the notebook_dir for our server. In the notebook configuration file, we apply the following changes:

- we tell jupyter not to sart a browser upon start - we access the server from a remote machine
- we set the IP address to the current IP address of the Raspberry Pi assuming it is the satic IP we require
- we set the port for the notbook server to listen on to 9090
- we enable mathjax for rendering math in notebooks
- we set the notebook_dir to ~/notebooks, the directory we created
- we use the password hash for the default server password jns
- We tell jupyter that we installed ipyparallel

To change settings upon installation, just edit ./jupyter/jupyter_notebook_config.py to suit your needs.

```
#!/bin/bash
# script name:      configure_jupyter.sh
# last modified:    2015/09/30
# sudo:             no

if [ $(id -u) = 0 ]
then
   echo "to be run as jns"
   exit 1
fi

# generate config and create notebook directory
jupyter notebook --generate-config
cd $home
mkdir notebooks

target=~/.jupyter/jupyter_notebook_config.py
```

```
# get current ip address - we assume it is static
ip=$(echo $(hostname -I))

# set up dictionary of changes for jupyter_config.py
declare -A arr
app='c.NotebookApp'
arr+=(["$app.open_browser"]="$app.open_browser = False")
arr+=(["$app.ip"]="$app.ip ='$ip'")
arr+=(["$app.port"]="$app.port = 9090")
arr+=(["$app.enable_mathjax"]="$app.enable_mathjax = True")
arr+=(["$app.notebook_dir"]="$app.notebook_dir = '/home/jns/notebooks'")
arr+=(["$app.password"]="$app.password =\
'sha1:5815fb7ca805:f09ed218dfcc908acb3e29c3b697079fea37486a'")
arr+=(["$app.server_extensions.append"] =\
"$app.server_extensions.append('ipyparallel.nbextension')")

# apply changes to jupyter_notebook_config.py

# change or append
for key in ${!arr[@]};do
    if grep -qF $key ${target}; then
        # key found -> replace line
        sed -i "/${key}/c ${arr[${key}]}" $target
    else
        # key not found -> append line
        echo "${arr[${key}]}" >> $target
    fi
done
```

1.6.5 Installation of Scientific Stack

The list of packages istalled here is just a suggestion. Feel free to adjust as needed.

```
#!/bin/bash
# script name:      install_stack.sh
# last modified:    2015/11/21
# sudo:             yes

if ! [ $(id -u) = 0 ]; then
   echo "to be run with sudo"
   exit 1
fi

pip install numpy
pip install matplotlib
pip install sympy
pip install pandas
pip install numexpr
pip install bottleneck
pip install SQLAlchemy
pip install openpyxl
pip install xlrd
pip install xlwt
pip install XlsxWriter
pip install beautifulsoup4
```

```
pip install html5lib

#-----------------------------------------------------
apt-get -y install libxml2-dev libxslt-dev
#-----------------------------------------------------

pip install lxml
pip install requests
pip install networkx
pip install plotly

#-----------------------------------------------------
apt-get -y install libblas-dev liblapack-dev
apt-get -y install libatlas-base-dev gfortran
#-----------------------------------------------------

pip install scipy
```

1.7 Keeping Your Installation up-to-date

Occasionally you may want to check for software updates for both the operating system and the python python packages we installed.

1.7.1 Operating System

```
sudo apt-get update
sudo apt-get upgrade
```

1.7.2 Python Packages

List outdated packages and if there are any, update them individually. Here we assume that package xyz is to be updated after the check:

```
pip list --outdated
sudo pip install xyz --upgrade
```

The script below automates the process: It genereates a list of outdated (pip installed) packages and subsequently processes the list to conduct upgrades.

```
#!/bin/bash
# script name:     upgrade_jns.sh
# last modified:   2015/11/22
# sudo:            yes

if [ $(whoami) != 'root' ]; then
        echo "Must be root to run $0"
        exit 1;
fi

# generate list of outdated packages
echo ">>> CHECKING INSTALLATION FOR OUTDATED PACKAGES..."
lst=(`pip list --outdated |grep -o '^\S*'`)

# process list of outdated packages
if [ ${#lst[@]} -eq 0 ]; then
```

```
    echo ">>> INSTALLATION UP TO DATE"
    exit 1;
else
    echo ">>> UPGRADING PACKAGES"
    for i in ${lst[@]}; do
        pip install ${i} --upgrade
    done
fi
```

1.7.3 OpenSSH Host Keys

To regenerate host keys, delete the old keys and reconfigure openssh-server. It is safe to run the commands over remote ssh based session. Your existing session shouldn't be interrupted:

```
sudo rm /etc/ssh/ssh_host*
sudo dpkg-reconfigure openssh-server
```